JYVÄSKYLÄ STUDIES IN COMPUTING

52

Mauri Leppänen

An Ontological Framework and a Methodical Skeleton for Method Engineering

A Contextual Approach

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 52

Mauri Leppänen

An Ontological Framework and a Methodical Skeleton for Method Engineering

A Contextual Approach

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksessa (Ag Aud. 3)
elokuun 22. päivänä 2005 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in the Building Agora, (Ag Aud. 3), on August 22, 2005 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2005

An Ontological Framework and a Methodical Skeleton for Method Engineering

A Contextual Approach

JYVÄSKYLÄ STUDIES IN COMPUTING 52

Mauri Leppänen

An Ontological Framework and a Methodical Skeleton for Method Engineering

A Contextual Approach

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2005

Editors
Seppo Puuronen
Department of Computer Science and Information Systems, University of Jyväskylä
Pekka Olsbo, Marja-Leena Tynkkynen
Publishing Unit, University Library of Jyväskylä

URN:ISBN:9513921867
ISBN 951-39-2186-7 (PDF)

ISBN 951-39-2133-6 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä

ABSTRACT

Leppänen, Mauri
An Ontological Framework and a Methodical Skeleton for Method Engineering – A Contextual Approach
Jyväskylä: University of Jyväskylä, 2005, 702 p.
(Jyväskylä Studies in Computing
ISSN 1456-5390; 52)
ISBN 951-39-2186-7
Finnish summary
Diss.

Method engineering (ME) is commonly carried out in an intuitive and improvising fashion. This is largely due to the lack of explicit methodical support. The literature does suggest an array of ME strategies, ME approaches, ME techniques, and ME procedures, but the assistance they provide is not satisfying. It can be argued that the ME field has not advanced far from its "pre-methodical" stage. An ISD method is an abstract and multifaceted notion, and there exist quite diverging views on its nature, structure, content, and significance. For this reason, to construct a feasible methodical support to method engineering, it is necessary to anchor it upon a theoretically sound and uniform conceptual foundation. The objective of this thesis is to construct an ontological framework for conceiving, understanding, structuring, and representing the phenomena related to an ISD method and its engineering, and to construct a methodical skeleton to support the ME process. The thesis crafts two design artifacts, OntoFrame and MEMES. OntoFrame is an ontological framework, comprising a number of component ontologies with a multi-dimensional structure. These components range from highly generic ontologies to ME-specific ones. Resulting from the application of a contextual approach in ontology engineering, the ontologies highlight, in a multi-faceted manner, contextual features of reality. The framework was derived from multiple theories and existing ME artifacts with deductive and inductive principles, and it is represented in UML-based meta models. MEMES is a normative prescription for ME, structuring and guiding the accomplishment of ME work. It consists of three ME workflows: ISD method requirements engineering, ISD method analysis, and ISD method evaluation. For each of the ME workflows, ME approaches, principles, and steps are suggested in MEMES. Both of the artifacts are evaluated extensively in a number of comparative analyses of existing artifacts. MEMES is also evaluated with empirical methods. The results of the thesis can be utilized in future research to analyze and compare existing artifacts and to construct new ones for the use of ISD and ME. MEMES can be applied in practice to support the engineering of generic and domain-specific ISD methods. The research follows the design theory paradigm and applies conceptual and empirical research methods.

Keywords: information systems development method, method engineering, metamodeling, ontology engineering, contextual approach, abstraction, design theory

ACM Computing Review Categories

D.2.1	Software Engineering: Requirements/Specifications *Languages, Methodologies, Tools*
D.2.2	Software Engineering: Design Tools and Techniques *Object-oriented design methods*
D.2.10	Software Engineering: Design *Methodologies, Representation*
H.1.0	Models and Principles: General
H.1.2	Models and Principles: User/Machine Systems *Human information processing*
I.6.5	Simulation and Modeling: Model Development *Modeling methodologies*

Author's address Mauri Leppänen
University of Jyväskylä
Department of Computer Science and
Information Systems
P.O. Box 35 (Agora)
FIN-40014 Jyväskylä, Finland
Email: mauri@cs.jyu.fi

Supervisor Professor Kalle Lyytinen
Department of Information Systems
The Weatherhead School of Management
Case Western Reserve University
Cleveland, USA

Reviewers Professor Juhani Iivari
Department of Information Processing Science
University of Oulu
Oulu, Finland

Professor John Krogstie
The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway

Opponent Professor Richard Welke
Georgia State University
Atlanta, USA

ACKNOWLEDGEMENTS

The road to this doctoral dissertation has been exceptionally long, partly due to the challenges caused by the broad scope of the thesis and partly due to my other obligations. Nevertheless, now the thesis is ready, and it is time to thank all of those who have made this possible.

First of all I would like to thank my supervisor, Professor Kalle Lyytinen for his encouragement, comments and constructive conversations. Not even the vast Atlantic Ocean between us could hinder his advice reach me and help finish the thesis.

I would also like to record my gratitude to the external reviewers of the thesis: Professor Juhani Iivari from University of Oulu and Professor John Krogstie from The Norwegian University of Science and Technology (NTNU). Their constructive comments and suggestions helped me improve my work.

In addition, I would like to express my gratitude to Professor Eero Peltola, who was the supervisor of my licentiate thesis. He was the one from whom I learned how to make conceptual research in a strict and thorough fashion.

Many of my colleagues have also contributed to my work. I participated in many research projects with Professor Vesa Savolainen, and we coauthored several articles. Tennis games with Vesa acted as a good counterbalance to research work. With Esa Auramäki I investigated the essence of exceptions and their modeling in information systems. In the last couple of years I have had most interesting discussions, often about philosophical issues, with Veikko Halttunen who has the same inquisitiveness into the fundamental subject matters as I do. I want also to thank Steve Legrand for checking the language.

This research was partly funded by COMAS Graduate School and the INFWEST.IT program. I express my gratitude to people in these organizations.

Finally, I would like to express my special thanks to my wife Leena for her continuous understanding and support throughout the long research process. Moreover, without her skills and patience I would have lost the battle with Word. And as a notice to Reija and Olli, perhaps now I have time for those other "projects".

Jyväskylä
May 2005

Mauri Leppänen

FIGURES

TABLES

CONTENTS

1 INTRODUCTION

In this chapter we will first describe the background of and motivation for the thesis. Second, we will define our research domain and decompose it into five sub-domains. We will also give an overview of those disciplines upon which this thesis has been built. Third, we will define the main research problem, research questions, and research objectives. In addition, we will introduce the main contributions, an ontological framework and a methodical skeleton for method engineering, and specify concrete objectives for them. Fourth, we will describe our research framework based on the design-science paradigm. Fifth, we will represent the cyclic and multi-domain research process and outline the research methods used in the thesis. Sixth, we will assess our research in terms of problem relevance, research contributions, design evaluation, and limitations. Finally, we will present the structure of the dissertation.

1.1 Background and Motivation

Information System Development Methods

Organizations have become highly dependent on advanced information technology (IT). High volumes of business transactions, huge amounts of data in databases and data warehouses, exceedingly complex calculation and inference rules, and needs for communication with 'instantaneous' concurrency among thousands of sites around the world cannot be coped with without the large-scale use of IT and high quality information systems.

Information systems are considered investments that have, besides operational importance, also strategic significance. This entails high demands for information systems development (ISD). A development process should be accomplished in an efficient and productive way. It should also yield an information system that satisfies organizational requirements (e.g. security, robustness, extendibility, maintainability, cost-effectiveness) as well as user requirements (e.g. functionality, acceptability, accuracy, timeliness, user-friendliness, reliability, usability, personalizability).

To support efficient and productive development of information systems, hundreds or even thousands of methods have been engineered during the last four decades. An *ISD method* is a prescription used in ISD to make organizational and technical changes in an IS possible or more productive. More specifically, an ISD method is a composition of paradigmatic assumptions, approaches, concepts, notations, models and guidelines. It conveys collective knowledge and experience of ISD process, application domain, IC technology, and human and social issues (cf. Iivari *et al.* 2001), externalized in the form of text books, manuals, or pro forma documents, and disseminated on paper, CD-rom, World-Wide-Web, etc.

Numerous empirical studies report on how methods can benefit ISD. The use of methods improves productivity (Fitzgerald 1998a, 318; Rahim *et al.* 1998, 975; Grant *et al.* 2003; Hardy *et al.* 1995) and communication (Palvia *et al.* 1993; Rahim *et al.* 1998, 975; Wastell 1996), and increases user involvement and fulfillment of user requirements (Schönström *et al.* 2003; Rahim *et al.* 1998, 975; Hardy *et al.* 1995). The methods enable the use of skills more effectively through skill specialization and division of labor (Fitzgerald 1998a, 318). They are seen as organizational memories (Stolterman 1994; Fitzgerald 1998a) that form useful vehicles of organizational and individual learning (Iivari *et al.* 2001, 183), advance process standardization (Roberts *et al.* 1998, 64; Avison *et al.* 1995a, 423) and improve project management (Avison *et al.* 1995a; Fitzgerald 1998a; Chatzoglou 1997).

However, there are a many empirical studies that indicate severe problems in the use of ISD methods. First, there are studies that show that method use is not as frequent as believed (Hardy *et al.* 1995; Russo *et al.* 1996, Chatzoglou 1997; Fitzgerald 1998a; Iivari *et al.* 1998b). Second, a large variety of explanations have been presented for having problems in the method use. Methods are perceived to be prescriptive, burdensome and difficult to apply (Middleton 1994, 474) and often too massive and complex to be easily adopted and adapted to a specific situation (Hidding 1997, 104). Rahim *et al.* (1998, 957) and Kautz *et al.* (1994) found out that the difficulty to learn the method was the most pressing problem. There exist also disappointments in productivity (Avison *et al.* 2003). The methods are seen to be monolithic (Hidding 1997) or one-dimensional (Avison *et al.* 2003, 81), advocating a single path or approach, which is often conceived as one-size-fit-all. Methods are not contingent on the type or size of a project, nor upon the technology environment and organizational context (Avison *et al.* 2003). Sometimes they are seen too detailed to efficiently support the planning of a project (Hidding 1997). Many methods are documented only on paper, making their use awkward and their customization difficult. Tools advocated by method proponents can be costly and they may require highly technical skills. Other reported problems include the inability of the method to cover the whole life cycle of software projects and the failure of the method to reduce project completion time.

There are also problems in a way methods are applied. According to Wastell (1996), methods may be applied in a ritualistic way, which inhibits creative thinking. Developers proceed in slavish and blind adherence to methods and lose sight of the fact that development of an actual system is the real objective (Fitzgerald 1994; Fitzgerald 1996b; Wastell 1996).

Part of the problems in method use can be traced back to human, organizational or technical settings. For example, without proper training a method can be totally ignored or only partly utilized. Incompatibility of the approach of a method with organizational culture and traditions may also cause unsolved problems. Further, the use of a method may be experienced as a nuisance and a waste of time, if there is no CASE tool supporting ISD work. Excluding all the problems due to these environmental issues, there still remain many problems that result from unsatisfactory features of existing methods.

More challenges to ISD and ISD methods are brought about by continuous evolution (a) in business and its environment, (b) among application areas, and (c) in approaches and technologies of development environments. Business processes are changing in various dimensions (e.g. flexibility, interconnectivity, coordination style, autonomy) due to market conditions, organizational models, and usage scenarios of information systems. They are required to act more effectively in shorter time-frames (Fitzgerald 1997). At the same time, business processes are getting more complex and difficult to manage. Businesses are increasingly moving towards extensive automation of their private and public processes. Increasing domestic and global competition and changing economics pressure to deliver information systems "yesterday" to exploit business opportunities (Wynekoop *et al.* 1997).

Resulting from evolution in business and its environment as well as from advancements in IT, novel application areas have emerged. Examples of the new areas are: e-commerce, m-commerce, www-information systems, multimedia information systems, trustworthy systems, bioinformatics, and ubiquitous systems. Typical for new areas is that they amalgamate organizational, conceptual and technical issues from several research fields, in the way bioinformatics does from biological data management, genomic information retrieval and bio data mining.

Rapid progress of technology has resulted in new architectural frameworks and platforms for information systems, e.g. J2EE, Visual Studio .NET, XML-based technology, service-oriented architectures, peer-to-peer technology, model-driven architecture (MDA), and grid computing technology. This has led to the birth and diffusion of new computing and development approaches and paradigms, e.g. object-oriented approach, agent-based approach, fuzzy approach, anywhere/any time/any means paradigm, generative programming approach, aspect-oriented approach, ontology & service oriented (OSO) programming approach, soft computing approach, peer-to-peer computing paradigm, etc. Especially, the component-based approach with reusable components has established a firm foothold in ISD. Companies rely far less on in-house development of systems, and buy software packages or outsource ISD instead (Bansler *et al.* 1994). This might be referred to as the industrialization of ISD.

Parts of the systems are less likely to require large-scale, long-term development projects, and more likely to be smaller, short term, incremental projects (Baskerville *et al.* 2001; Fitzgerald *et al.* 2002; Baskerville *et al.* 2004). With the emergence of light web-based applications, new birth of ”quick and dirty” approaches, currently called agile approaches or short cycle time systems development, are getting popular (Agile Alliance 2002; Cockburn 2001; Astels *et al.* 2002; Baskerville *et al.* 2004). These emphasize e.g. individuals and interactions over processes and tools, working software over comprehensive documentation, and customer collaboration over contract negotiation. Also, emergent organizations require new practices for ISD, such as continuous analysis of IS applications, dynamic requirements negotiations, and continuous redevelopment (Truex *et al.* 1999).

New technologies also enable new ways of working in ISD projects. Besides advanced CASE (Computer-Aided Systems Engineering) tools and environments (e.g. Rational Rose, Select Enterprise, Silverrun, Prosa, etc.[1]), there are tools, called CAME (Computer-aided Method Engineering) (e.g. RAMATIC (Bergsten *et al.* 1989), ConceptBase (Jarke 1992), Navigator (Ernst & Young 1995), MetaEdit+ (Kelly *et al.* 1996)), with which the methods and environments can be tailored according to the needs of projects. Many kinds of tools to further cooperation and coordination of ISD have also been taken into use.

To summarize, ISD methods appear to be useful both to ISD processes and their outcomes. However, there are several problems in methods and method use that should be overcome. In addition, although numerous methods of different kinds already exist, more methods with new features are still desired. This process of engineering new methods and customizing existing ones is propelled by the everlasting changes in organisational and technological environments of ISD.

Method Engineering

Method engineering (ME) means all those actions by which an ISD method is developed, and later customized and configured to fit the needs of an organization and/or an ISD project. The contents of, and relationships between, ME actions vary depending on the target, strategy and organizational context of ME. First, the ME may aim to produce a generic method, a domain-specific method, an organization-specific method, or a project-specific method. Second, the strategy of the ME may be “from scratch”, or adapting and/or integrating (parts of) existing methods. Third, the ME or parts thereof can be accomplished outside or inside the organization for which the method is to be engineered. Fourth, ME actions can be variously scheduled in relation to corresponding ISD

[1] See Index of CASE tools: http://www.cs.queensu.ca/Software-Engineering/tools.html .

actions. Due to this heterogeneous nature, the process of ME is difficult to piece together and structure, and in particular to manage.

ME has appeared to be problematic in many ways. One set of problems arises from a variety of conceptions about the nature and role, which the method is seen to have in ISD (e.g. Wastell 1996; Baskerville *et al.* 1992; Baskerville 1996). For example, ISD can be seen as a transformation process (Wand 1988a; Tracz *et al.* 1993), a problem solving process (Sol 1992), a decision making process (Iivari 1982; Jarke *et al.* 1990; Grosz *et al.* 1997), or a learning process (Iivari *et al.* 1987; Lyytinen *et al.* 1999; Ramesh *et al.* 1994). It is very challenging to engineer a method that can successfully serve in roles of such a variety. Another set of problems stems from the contents and structure of the methods. Next, we discuss these problems in relation to ME actions in which they are encountered. We use the following taxonomy of ME actions: (a) analysis of current methods, (b) characterization of a target ISD context, and (c) adaptation and integration.

In practice the ME never starts with ”an empty table”. Although ME would not end up to heavily utilizing existing methods, reviewing them is an integral part of every ME effort. Current methods largely differ from each other in terms of their assumptions, approaches, concepts, notations, coverage, flexibility, formality, etc. Also, ways of describing methods vary from narrative texts used in text-books (e.g. Yourdon 1989; Skidmore *et al.* 1992) to semi-formal specifications of the syntax and semantics of notations (cf. UML, Booch *et al.* 1999). Often it is difficult to gain a clear conception of what approaches and terms in a method really mean. Yet more difficult it is to make precise and neutral comparisons between the methods. Although there is a large set of literature on feature lists, taxonomies, and frameworks for comparative reviews (e.g. Olle *et al.* 1983; Olle *et al.* 1986), the support they provide to the analysis is inadequate in many respects. In addition, to make reliable assessments on the methods, some knowledge of their proved applicability is needed. However, experiences from ISD efforts executed in organizations are not structured and recorded in a way that enables their effortless utilization. In contrast, knowledge of former ISD efforts is usually unstructured in the heads of ISD analysts and designers who have, in the worst case, left the organization. To summarize, there should be some uniform framework that could be used in analyzing current methods and contexts in which the methods have been used.

ME commonly aims to produce an ISD method for a specific project. To be able to select among existing methods and customize the one for the use of the project, the target ISD context should be characterized properly. There is a large set of contingency frameworks composed of factors to be used for the characterization (e.g. Naumann *et al.* 1980; Davis 1982; Burns *et al.* 1985; Louadi *et al.* 1991; van Swede *et al.* 1993; van Slooten *et al.* 1996; Punter *et al.* 1996; Harmsen 1997; Roberts *et al.* 1998; Yadav *et al.* 2001). There are, however, many problems related to the use of such frameworks (cf. Kumar *et al.* 1992; Avison 1996; Tolvanen 1998; Zhu 2002). What is needed here is a conceptual foundation on the basis of which upcoming ISD contexts could be described in a way that is comparative to the descriptions of the application areas of the ISD methods and accomplished ISD efforts.

Method engineering is, to a large extent, conceived as an intuitive and creative effort that is hard to systematize. An ISD method seems to be quite abstract to perceive and difficult to deal with. This has often resulted in ad hoc –like decisions and actions in ME. For instance, one may "rush" for a new method, mainly inspired by advertisements or recommendations of consultants, without properly contemplating whether the features of the method and the needs of the project really match. Second, neglects in ME may lead to adherence to the current method, though there is strong evidence, got from previous projects, that improvements into the method are urgent. Third, it may be decided to exclude some parts of the current method without considering consequences for the usability of the rest of the method. The same kinds of concerns pertain to regardless extensions of the current method with components taken from other method(s).

When done carefully ME needs time and resources, which are less and less available in contemporary organizations living in the "hectic" world. The selection, customization and configuration of a new method for an organization can be organized as a separate ME project, which pilots the method in some ISD project(s). This kind of ME is in a better position in ensuring resources. In contrast, an ME effort that is carried out in parallel with an on-going ISD project is experienced as extra work that unnecessarily burdens the budget of the ISD project. For this reason, adaptation should be able to be accomplished as systematically and efficiently as possible.

To make ME more efficient, computer-aided engineering environments (CAME) and MetaCase tools have been developed. From those dating back to the 1980's (PLS/PSA (Teichroew *et al.* 1977; Teichroew *et al.* 1980), MetaView (Sorenson *et al.* 1988), RAMATIC (Bergsten *et al.* 1989)), tools and environments have advanced (Maestro II (Merbeth 1991), ConceptBase (Jarke 1992), Navigator (Ernst & Young 1995), MetaEdit+ (Kelly *et al.* 1996), Decamerone (Harmsen 1997), and MERU (Gupta *et al.* 2001), but still they have severe shortcomings in functionalities, user-friendliness, flexibility, process support, etc.

To summarize, an ISD method is conceived to be an abstract artifact that is very difficult to engineer. But because methods have been, are, and will be significant to the success of ISD, it is extremely important to support their development, adaptation, integration, customization, and configuration. There are many kinds of problems, which complicate method engineering e.g. in analysis and comparison of methods, characterizing prior and target ISD contexts, and in integrating and adapting existing methods or parts thereof. The ME literature suggests various ME strategies and ME approaches (e.g. Kumar *et al.* 1992; Oei 1995; Harmsen 1997; Tolvanen 1998; Saeki 1998; Leppänen 2000; Ralyte *et al.* 2003), metamodeling languages (e.g. Chen 1976; Nijssen *et al.* 1989; Smolander 1991; Heym *et al.* 1992a; Jarke *et al.* 1995; Kelly *et al.* 1996; Harmsen 1997; Bandinelli *et al.* 1993; Deiters *et al.* 1994; Christie 1993), ME techniques (e.g. van Slooten *et al.* 1993; Kinnunen *et al.* 1996; Punter *et al.* 1996; Saeki 2003) and

ME procedures (e.g. Vlasblom *et al.* 1995; Nuseibeh *et al.* 1996; Song 1997; Harmsen 1997; Tolvanen 1998; Gupta *et al.* 2001). Suggested ME strategies, ME approaches, ME techniques and ME procedures constitute an unrelated and incompatible set of artifacts that does not nullify the fact that the current methodical support to ME is hopelessly inadequate to overcome prevailing problems in ME.

Conclusions

We have above discussed information systems development and experience from the use of methods in ISD. Despite disputable problems in method use and temptations to amethodical approaches (Truex *et al.* 2000), the ISD methods are undoubtedly needed and used in the future. Rapid and pervasive changes in business, application and technological environments increase pressure to renew current methods, as well as to develop new kinds of methods.

Method engineering is often seen to be an "unnecessary nuisance" to be accomplished with minimum efforts. It is, however, so complicated and multifaceted array of intentions, actors, actions, deliverables and tools that it has to be taken seriously. Otherwise, problems in methods and method use may disperse negative impacts, via failures in ISD, to information systems and to business processes in the organization(s) as well. Although ME is highly interrelated to ISD endeavors, it has to be appreciated as an "independent" effort that is entitled to have proper methodical support. But to develop methodical support that goes beyond the present state of the art, we also need a uniform and consistent conceptual foundation, which provides appropriate concepts and conceptual "building blocks" from which profitable support to ME can be constructed.

In conclusion, we argue that *there is the need for (1) a conceptual foundation that provides concepts and constructs to conceive, understand, structure and represent phenomena related to an ISD method and its engineering, and for (2) a methodical support with which the process of method engineering can be accomplished in a more structured and productive manner.*

1.2 Research Domain

A *research domain* is the subject matter under study in a research effort (Nunamaker *et al.* 1991, 91). In this section we describe the research domain of our study, starting from the identification of the main subject of the study and then describing sub-domains related to it. We also discuss the theoretical foundations and the research fields underlying our research domain.

The main research subject of this thesis is an ISD method. We are interested in what kind of artifact an ISD method is, what the conceptual contents, structure and representation of an ISD method are, and in particular, how an ISD method can be or should be engineered. Hence, *our research domain is composed of all those sub-domains that are related to an ISD method.*

An ISD method is the main deliverable of ME actions. To understand the underlying intentions of, and assumptions behind, an ISD method, it is necessary to know, how the method has been engineered, by whom, why, where, and when. This part of the research domain is called the *ME domain.* Second, an ISD method describes / prescribes contexts in which the ISD method is to be used, that is to say ISD contexts. To make sense of the essence of the ISD context, we need a conceptual foundation, which enables us to recognize, structure, represent, manage, and assess phenomena related to the information systems development. We call this sub-domain the *ISD domain.* Third, through describing deliverables of an ISD, here called the IS models, an ISD method also semantically structures contexts in which the IS models are implemented and utilized. Hence, also IS users with their intentions, IS processes with their logical and temporal relations, and IS deliverables with their contents and meanings are relevant for our study. This sub-domain is called the *IS domain.*

Hence, our research domain constitutes a multi-layered structure that comprises, at least, the IS domain, the ISD domain, and the ME domain. But there are still more sub-domains. First, we need concepts and constructs with which the nature, structure and representation of an *ISD method* itself can be conceived. Second, to engineer methodical support for ME, we need concepts and constructs with which the nature, structure and representation of an *ME method* can be understood. With these complements we have achieved the structure of sub-domains, which covers four processing layers (Figure 1). The topmost layer corresponds to this research work (RW). The other layers are method engineering (ME), information systems development (ISD) and information system (IS). In Figure 1 the main deliverables are represented by rectangles on each layer. Ellipses stand for processes, which produce the deliverables. The layers are related to each other through the describes/prescribes relationships. The research work produces RW deliverables that embrace an ontological framework and a methodical skeleton for ME. Next, we discuss the nature of these two deliverables and the research fields underlying them, first for the ontological framework and then for the methodical skeleton.

Ontological Framework (OntoFrame)

In order to understand, analyze, compare and engineer ISD methods, we need a consistent and coherent set of concepts and constructs. Information systems science is a rather young discipline. For this reason, views and concepts in the field greatly differ between schools and approaches. Even for key notions such as 'information system', 'object system' and 'ISD method' there are dozens of

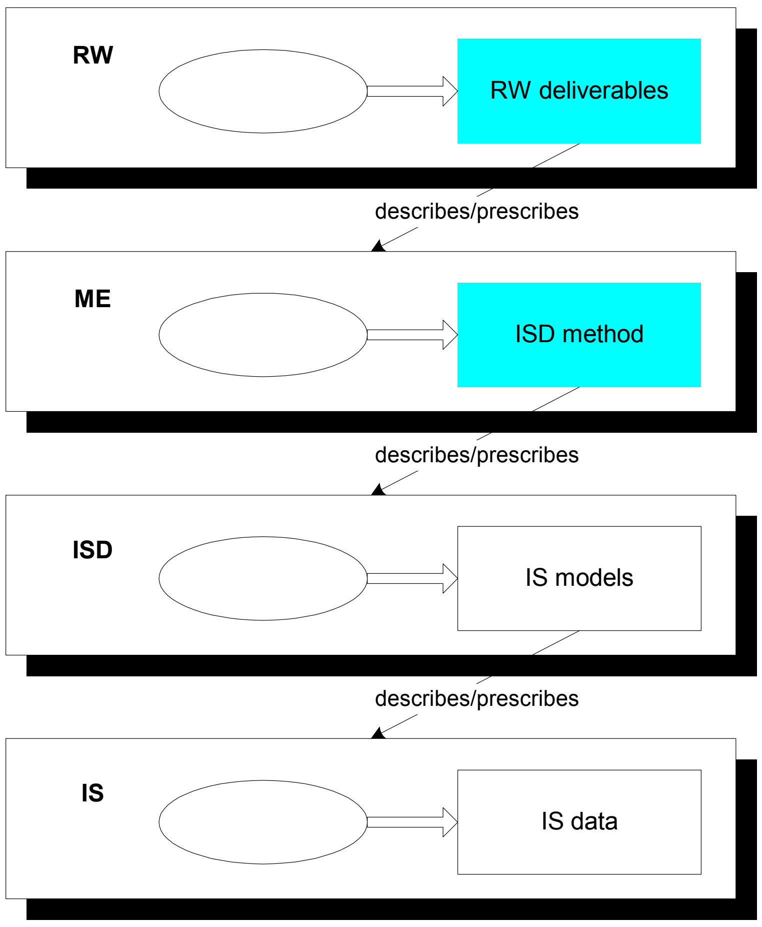


FIGURE 1 Multi-layered structure of the research domain

different interpretations (Ein-Dor *et al.* 1993; Mentzas 1994; Carvalho 1999; Barron *et al.* 1999; Avison *et al.* 1996; Avison *et al.* 1995a). This is partly due to the fact that theories and concepts in them have been established by scientists, who have come from different disciplines. Difficulties in finding a common "language" result in frequent misunderstandings. This necessitates conceptual and terminological preciseness and clarity. These are subject matters in particular in two disciplines. The disciplines are: (a) conceptual modeling, and especially metamodeling, and (b) ontology engineering.

Conceptual modeling means describing some aspects of the physical or social world around us for the purposes of understanding, explanation, prediction, reasoning, and communication (cf. Kangassalo 2002, VI). A conceptual model is intended to provide an accurate, complete representation of someone's or some group's perceptions of the semantics underlying a domain or some part of a domain (Bodart *et al.* 2001). Typically, models of the same domains share, on a general level, some concepts and constructs. These similarities can be modeled resulting in models of models, or meta models. A discipline studying meta models, languages to represent meta models, and procedures to produce meta models, is called *metamodeling*. Metamodeling is also the name of a process, which takes place on one level of abstraction and logic higher than modeling process (cf. Tolvanen 1998, 82).

Ontology is the study of existence, of all kinds of things – abstract or concrete - that make up the world (Sowa 2000, 51). It concerns "what is out there" (Quine 1953). *Ontology* is an explicit specification of a conceptualization of some part of reality that is of interest (cf. Gruber 1993, 199). A specification can be presented in the form of a vocabulary, a taxonomy, a thesaurus, a framework, or a theory (Sugumaran *et al.* 2002, 253). An ontology provides "glasses" through which one can perceive, conceive, and structure the world - in our case the research domain. *Ontology engineering* is a discipline, which studies ontologies, ontology representation languages and procedures for engineering ontologies.

Hence, both metamodeling and ontology engineering can be used in building a well-defined conceptual foundation. Both of them can be performed as a structured process to yield explicitly defined concepts, terminology and rules to represent consensual knowledge about relevant domains. A meta model as well as an ontology can be presented on a level that is general enough to abstract away specificities of a single application area, an ISD context or an ME context. Metamodeling and ontology engineering support one another in many respects. A meta model facilitates communication about an ontology, reveals inconsistencies and anomalies within an ontology, streamlines the comparison of ontologies, enables ontology engineering and supports ontology-based method engineering (Davies *et al.* 2003). For these reasons, we build our conceptual foundation upon these disciplines. To indicate our adherence to ontology engineering we name our conceptual foundation the ontological framework, shortly OntoFrame. *OntoFrame* is an ontological framework, comprising a number of component ontologies with a multi-dimensional structure. These components range from highly generic ontologies to ME-specific ones.

Methodical Skeleton (MEMES)

Next, we turn our discussion on the nature and the underlying research fields of our second RW deliverable that is a methodical skeleton for ME (see Figure 1). As mentioned above, the ME literature suggests only some ME strategies, ME approaches, metamodeling languages, ME techniques and ME procedures. To have an overall picture of, and to manage the whole process of method engineering, we need a more comprehensive support for ME. With that aim we suggest the ME methodical skeleton, called MEMES (**M**ethod **E**ngineering **ME**thodical **S**keleton). The *ME methodical skeleton* is a normative prescription of the ME context, structuring and guiding the ME process. To elaborate the notion of the methodical skeleton and to position it among the other artifacts in the literature describing / prescribing an ME effort, we compare it with the notions of an ME conceptual framework and an ME methodical framework. An *ME conceptual framework* provides generic concepts and constructs for conceiving and structuring ME contexts, or parts thereof. Similar frameworks are used at the ISD layer, for instance, to evaluate and compare ISD methods (e.g. Iivari *et al.* 1983; Essink 1986; Iivari 1994; Jayaratna 1994; Tudor *et al.* 1995).

An *ME methodical framework* is built up from meta models. In a simple form, the framework is composed of ISD meta models that are used to semi-formally describe the abstract syntax of the ISD models. Meta models provide the concepts and constructs used in an ME effort but only for the part that concerns phenomena in the ISD domain. In a more comprehensive form, the ME methodical framework also includes ME meta models that describe ME process models on a general level. Corresponding frameworks at the ISD layer are suggested by e.g. Henderson-Sellers *et al.* (1999c) and Hruby (2000b)[2].

Both of the ME artifacts introduced above are descriptive and contain no normative ingredients. The ME methodical skeleton provides all that have been mentioned above, and in addition major constructs for a skeleton-like structure of an ME process. This structure integrates and gives normative meanings for the meta models mentioned above. The methodical skeleton is not, however, a complete ME method. Instead, an ME method can be derived from MEMES.

MEMES has been firmly grounded on OntoFrame, as seen in Figure 2. In the figure the left side describes MEMES in its intentional and functional environment. MEMES is to be applied in an ME context to engineer an ISD method, which in turn is to be deployed in an ISD context to develop an IS. The right side in the figure describes the structure of OntoFrame from the viewpoint of the ME method ontology[3]. The ME method ontology includes the ME ontology, the ISD method ontology, the ISD ontology, and the IS ontology. The arrows denote how OntoFrame has been deployed to engineer the components of MEMES, an ISD method and IS models. We can see that the structure of MEMES has been adapted from the ME method ontology. The main components of MEMES are ME models, ISD meta models and IS meta models[4]. The ME models have been specialized and instantiated from the ME ontology. They describe/prescribe what is to be done, with which and why in the ME context. The ISD meta models and the IS meta models have been selected and adapted from the ISD ontology and the IS ontology, correspondingly. Likewise, the structure of an ISD method is to be adapted from the ISD method ontology. The ISD models are specialized and instantiated from the ISD ontology. The concepts and constructs of the IS meta models are selected and adapted from those belonging to the IS ontology. The IS models are specialized and instantiated from the IS ontology.

The research field studying issues relevant to the ME methodical skeleton, and more generally the nature, structure, contents and engineering of methods, is known as *method engineering* (ME)[5]. Method engineering is also regarded as an approach to, or a process of, analysis, design, implementation and

---

[2] The term 'methodical framework' is sometimes (cf. Krogstie *et al.* 1996) used to mean a conceptual framework in terms of our taxonomy.

[3] The ontologies within OntoFrame will be defined later in this thesis.

[4] In Chapter 11 we will give a more detailed view of the structure of MEMES.

[5] Kumar and Welke (1992), the "godfathers" of ME, used the term 'methodology engineering' to refer to "a meta-methodology for designing and implementing ISD methodologies" (ibid p. 257).

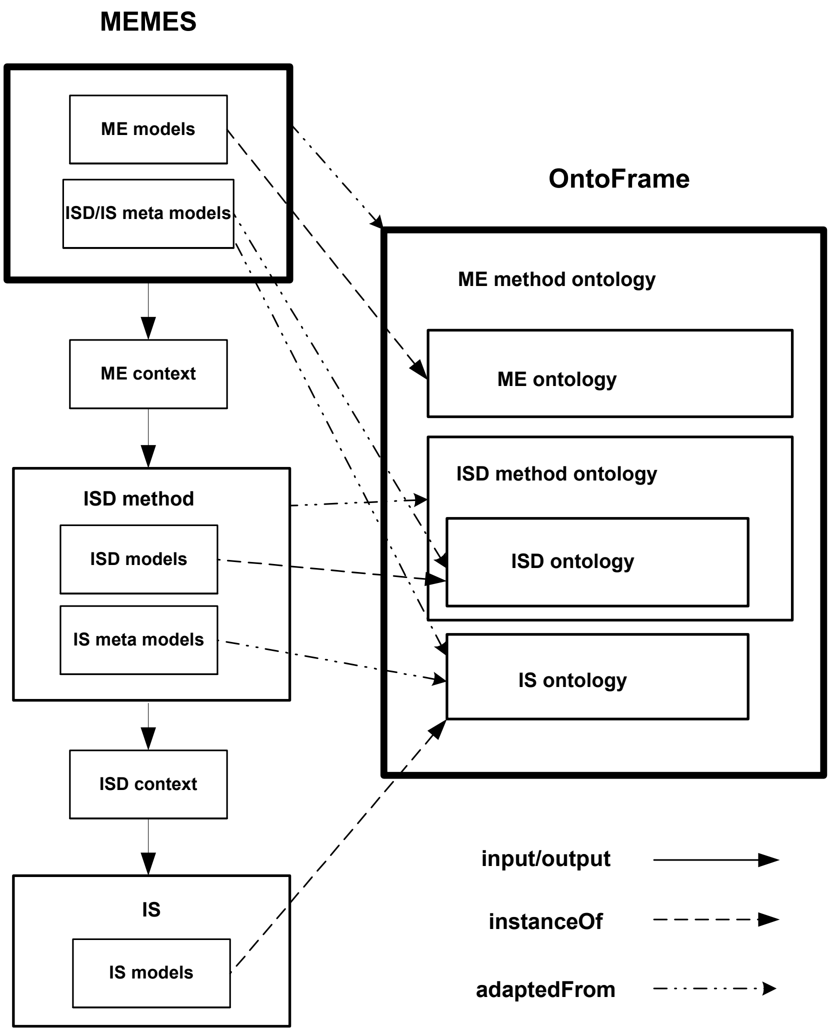


FIGURE 2 MEMES and OntoFrame

evaluation of ISD methods (cf. Brinkkemper *et al.* 1999, 209; ter Hofstede *et al.* 1997, 401; Tolvanen *et al.* 1996, 296; Harmsen 1997, 25). Since an ISD method is often modeled in an early phase of ME, metamodeling is an essential part of method engineering. It provides the concepts and notations for IS meta models and ISD meta models, as well as rules for using them.

1.3 Research Questions and Objectives

The *main research problem* of the thesis is: *"How to conceive and methodically support the engineering of an ISD method?"* This problem can be decomposed into the following *research questions:*

- What is the conceptual foundation with which phenomena in the research sub-domains can be conceived, understood, structured and presented?
- What are the nature, contents and structure of an ISD method?

- How to structure and support the process of method engineering?

As seen from the above, the most essential single subject matter in this study is a method, considered from two viewpoints. First, a method appears as the main target of method engineering. Second, a method is needed to support and guide the process of method engineering. In the former case, we are interested in the nature, contents, structure, and models of an ISD method. In the latter case, we are challenged with constructing a methodical artifact, which contains the essentials of an ME method. But before we can pursue a unified and sound view of these issues, we need a comprehensive, consistent and coherent conceptual foundation that covers the relevant research sub-domains and reveals, in particular, the meanings of those issues.

To be able to conceive meanings of phenomena in the sub-domains we apply a contextual approach, based on the notion of a context. A context is a suitable notion for many reasons. First, it is highly universal, known and applied in a number of disciplines, including formal logic (e.g. Costa 1999), knowledge representation and reasoning (e.g. Brezillon *et al.* 1998; Sowa 2000), machine learning (e.g. Matwin *et al.* 1996), pragmatics (e.g. Levinson 1983), computational linguistics (e.g. Berthouzoz 1999), sociological linguistic (e.g. Halliday 1978), problem solving (e.g. Motschnig-Pitrik *et al.* 2001), organizational theory (e.g Weick 1995), cognitive psychology (e.g. Kokinov 1999), and information systems (Kyng *et al.* 1997). Second, it is a common term also in the ordinary speech. Third, the most common aim of the use of context in various disciplines is to consider a focal thing or an event of interest as a part of the environment (i.e. context) in order to understand its nature and meaning (Duranti *et al.* 1992). That is precisely what we wish to achieve with OntoFrame.

From the aforementioned research problems we can infer the *main objectives* of this study: *(1) to construct an ontological framework for conceiving, understanding, structuring, and representing phenomena in an IS, an ISD, an ISD method, an ME, and an ME method with contextual concepts, and (2) to construct a methodical skeleton to support the ME context.*

The research area is very large. Therefore, we are forced to exclude, completely or partly, many interesting research issues. For instance, as we have adopted a methodical view on ME, we exclude subject matters that are related to IS/ISD technology and technical support to method engineering. Second, we mainly focus on "operational" processes of method engineering, not on managerial issues. Although we recognize the importance of project management and identify generic structures of it, we are neither aiming to specialize nor instantiate them into the use of the ME context.

Next, we define more concrete objectives for the ontological framework (OntoFrame) and the methodical skeleton (MEMES).

OntoFrame

OntoFrame is a result from the application of approaches and principles in two disciplines: conceptual modeling and ontology engineering. Thereby it has also inherited quality measures from them. In conceptual modeling the quality of a model is measured in terms of adequacy or completeness (e.g. Amberg 1996; Bajaj *et al.* 1999; Moynihan 1996; Chaves *et al.* 1996; Moody 2003b), readability or legibility (e.g Hardgrave *et al.* 1995; Chaves *et al.* 1996), and easy-to-use or efficiency (e.g. Kramer *et al.* 1991; Bock *et al.* 1993; Kim *et al.* 1995; Bajaj 2001; Gemino *et al.* 2002). In addition to the single criteria mentioned above, research in conceptual modeling has produced comprehensive frameworks of quality criteria (e.g. Lindland *et al.* 1994; Krogstie 1995; Krogstie *et al.* 2000).

In ontology engineering, there are no such well-established quality criteria (Weinberger *et al.* 2003). Gruber (1995), for instance, proposes clarity, coherence, extendibility, minimal coding bias, and minimal ontological commitment. Fox (1998) defines functional completeness, generality, efficiency, perspicuity, precision/granularity, and minimality. Uschold *et al.* (1996) apply the qualities of clarity, coherence, and extendibility. Weinberger *et al.* (2003) distinguish between conceptual coverage, utility and usability. Ruiz *et al.* (2004) require that ontology is clear, precise, coherent and consistent.

In specifying objectives for the ontological framework we should also learn from qualities specified for conceptual frameworks (e.g. Iivari *et al.* 1983; Bodart *et al.* 1983; Brodie *et al.* 1983; Essink 1986; van Swede *et al.* 1993; Falkenberg *et al.* 1998). Brodie *et al.* (1983), for instance, brings out the following requirements for an ISD method framework: general, precise, comprehensive, balanced, flexible, and unbiased. The Frisco[6] framework (Falkenberg *et al.* 1998) has been built by five principles (ibid p. 10): global consistency, generality, simple is beautiful, anchoring information system concepts in related fields, and a conceptual foundation to be build upon.

Based on the works in conceptual modeling and ontology engineering, as well as taking into account qualities related to conceptual frameworks, we specify the following objectives for OntoFrame:

1. *Comprehensiveness*
   OntoFrame should cover all the sub-domains mentioned above (i.e. IS, ISD, ISD method, ME, and ME method). Due to their extensiveness, all the phenomena contained in them cannot, of course, be addressed in the framework.
2. *Contextuality*
   OntoFrame should enable to contextualize phenomena of reality, that is to treat them as contexts, or as parts of a context.
3. *Consistency*
   In OntoFrame there should be no contradictions between the definitions of concepts and constructs.
4. *Coherence*
   In OntoFrame each concept should be related, directly or indirectly, to every other concept in a well-established way.

[6] Frisco = A FRamework of Information System COncepts.

5. *Generality*
   OntoFrame is meant to be as general as possible, in order to be shared by various communities and to help find the right level of generality at which modeling approaches can be related to each other and to which new emerging modeling approaches can be attached. This objective cannot be fully achieved due to great discrepancies between views of schools and approaches.
6. *Clarity*
   OntoFrame should effectively communicate the intended distinctions between the concepts. This means that ambiguity should be minimized in the definitions.
7. *Naturalness*
   OntoFrame should be based on mental structures that are inherently typical for human conceptions and abstractions.
8. *Generativeness*
   OntoFrame should be established in a way that allows one to derive specific concepts of the framework from more general concepts by specialization.
9. *Extensibility*
   OntoFrame should be extendable with new and more specialized concepts. Extensions should be possible without the revision of existing definitions. In this sense the purpose of the framework is to serve as a kind of "crystallization kernel" (cf. Falkenberg *et al.* 1998).
10. *Modularity*
    OntoFrame should be composed of well-defined modules, which together constitute an integrated whole.
11. *Theory bases*
    OntoFrame should be anchored on relevant and sound theories, such as semiotics, semantics, pragmatics, and systems theory.
12. *Applicability*
    OntoFrame should be applicable for three kinds of intentions, descriptive, analytical and constructive intentions. In the descriptive sense OntoFrame should offer concepts and a terminology for conceiving, understanding, structuring and presenting phenomena in the concerned sub-domains. In the analytical sense OntoFrame should provide the key concepts and constructs for the analysis and comparison of existing artifacts. An artifact here means an ontology, a meta-model, a framework, a frame of reference, a method, etc. of the concerned sub-domains. In the constructive sense OntoFrame should support the construction of other artifacts, in particular ISD methods and ME methods, by providing a conceptual foundation for these artifacts.

OntoFrame is composed of ontologies at various levels. Therefore, we next consider the aforementioned objectives from the viewpoint of ontology engineering[7]. The main purpose of an ontology is to advance communication between people. To fulfill this aim, the concepts and constructs in each part of OntoFrame should be natural and clear (Gruber 1995; Uschold *et al.* 1996). The view provided by an ontology about a slice of reality should be comprehensive, consistent and coherent (Gruber 1995; Uschold *et al.* 1996; Fox 1998; Ruiz *et al.* 2004). These objectives can be furthered by building an ontology upon relevant and sound theories. Some of these theories should lay the foundation for viewing phenomena of reality as contexts or parts of a context. Resulting from the contextual approach applied in this thesis, the comprehensiveness is not interpreted as a quantitative measure, evaluated in the numbers of concepts and constructs, but in terms of how faithfully an ontology, or a set of ontologies, is able to reflect contextual features in reality. To be applicable in more than one specific situation, an ontology should capture general semantics of reality (Fox 1998). To balance between specificity and generality demanded by the sub-domains, OntoFrame should be composed of ontologies at several levels of generality. No single ontology, neither any ontological framework, can ever become complete (Gruber 1995; Uschold *et al.* 1996). To ease making extensions and still maintaining the coherence and consistence of the structure, OntoFrame should be generative and modular. And last but not least, OntoFrame should be applicable. An ontology that fulfils all the other goals but fails to provide support for the intended purposes is useless. That is why we want to emphasize this goal in particular

The objectives are interrelated in many ways. On one hand, there are objectives that have positive influence on the other objectives. For instance, the derivation of concepts from more generic concepts helps establish and maintain the framework consistent and coherent. Rooting the concepts on the proper theories contributes to the naturalness of the framework. Furthermore, extensibility can be advanced by the modular structure and the generativeness of the framework. On the other hand, there are objectives that are, at least to some degree, contradictory to the other objectives. For instance, the more comprehensive the framework becomes, the more difficult it is to fulfil the objectives of clarity, consistence and coherence.

To build a comprehensive and uniform conceptual foundation for our research domain is very challenging. This was experienced, although on a much smaller scale, by the Frisco group (Falkenberg *et al.* 1998), when it was building a conceptual foundation for the IS domain. Difficulties are multiplied here, as we extend the scope of the conceptual foundation far beyond the IS domain. We agree with Harmsen (1997, 144) that no universal foundation can be established, partly due to the relative immaturity of the information system science. Therefore, we pursue a foundation that is a coherent and consistent representation of the research domain and that can also be used to relate varying views. We strive for these objectives with the following means. First, we carefully select those views that are in harmony with one another. By this

[7] OntoFrame could be, respectively, considered from the viewpoint of conceptual modeling as well.

we try to ensure that among the concepts based on the views there are no inherent inconsistencies. Second, we apply the integration strategy to couple aggregates of view-based concepts together. Whenever bare integration is not enough, we adapt concepts to get them properly connected to our framework.

The ontological framework is presented in two forms, informally and semi-formally. Every concept and construct within the framework is defined in natural language. The definitions are embedded in the text and also included in the vocabulary in Appendix 1. We use meta models to give an overview of the concepts and constructs in each part of the sub-domains. This form is also used to specify constraints imposing the relationships between the concepts. Meta models are presented in a UML based language. For some core parts of OntoFrame we also use the first order predicate calculus to define axioms.

MEMES

While the ontological framework provides concepts and constructs for the sub-domains, MEMES aims at providing methodical "building blocks" and guidelines to carry out an ME effort. MEMES is not intended to be a method in a strict sense. To build a comprehensive method would require the specification of ingredients, such as models with notations, techniques with a large variety, quality metrics, procedures for ME project management, etc (cf. Graham *et al.* 1997, 2). To reach such a level of concreteness and comprehensiveness would be impossible in this study.

The ISD literature suggests artifacts that come close to the notion of a methodical skeleton. In the Unified Process (Jacobson *et al.* 1999) the notion of a process is defined to refer to "a concept that works as a template that can be reused by creating instances of it" (ibid p. 24). It is compared to a class form, which can be used to create instances. The OPEN framework (Graham *et al.* 1997) is "a framework for third-generation OO software development methods" (ibid p. 4). It is "a methodical framework", which can be instantiated to have specific methodological processes (Henderson-Sellers *et al.* (1999b, 40)[8]. Besides instantiable meta models, the Unified Process and the OPEN framework provide descriptions of ISD actions and ISD deliverables on a rather detailed level.

Both the Unified Process and the OPEN framework concern ISD, whereas MEMES is to support ME. Regardless of this dissimilarity, we compose, as is done in the Unified Process and the OPEN framework, MEMES from meta models. We also include prescriptive models in MEMES (see Figure 2). Resulting from anchoring MEMES upon the conceptual basis provided by OntoFrame, we pursue the categorizations and structures of views, actions, and deliverables of ME, which makes MEMES much more modular and flexible for adaptations and instantiations, as compared to the OPEN framework and the Unified Process.

[8] The OPEN framework is also called "a methodological metamodel of process" and "architecture of a process metamodel" (Henderson-Sellers 1999, 63).

There are a large variety of criteria suggested for the evaluation and comparison of ISD methods in the ISD literature (e.g. CRIS[9] conferences, EMMSAD[10] workshops, UML Conferences). Some of them are common in the frameworks mentioned above. Because MEMES is not a method, we have selected only a small set of criteria to be used as the objectives of MEMES. The objectives of MEMES are as follows:

1. *MEMES should be based on a solid and sound view on the sub-domains.*
   The methodical skeleton should be built upon a conceptual foundation that is anchored on sound theories. Satisfying this objective is furthered by the use of the concepts and constructs in OntoFrame for conceiving and structuring the target and the process of method engineering with contextual concepts.
2. *MEMES should be modular and flexible.*
   MEMES should be composed of structural and functional components that facilitate the elaboration, customization and configuration of the skeleton into a specific ME method. Despite the modular structure, MEMES should constitute a uniform, consistent and coherent totality.
3. *MEMES should be applicable.*
   MEMES should be applicable for framing, constructive and analytical intentions. The framing intension means that MEMES should provide concepts and constructs to help make sense of and structure phenomena in ME in reality. The constructive intention means that MEMES should support the engineering of an ISD method, or parts thereof. The analytical intention means that MEMES should provide concepts and constructs for the analysis and comparison of existing ME artifacts. ME artifacts here mean ME strategies, ME approaches, ISD meta models, ME techniques, and ME procedures.

MEMES is to be presented in natural language, supported with diagrams illustrating structural and functional features of the skeleton.

1.4 Research Framework

Up till now, we have described the background and the motivation of this study, outlined the research domain and defined the research questions and objectives. In this section we position and characterize our study with the research framework of Hevner *et al.* (2004).

According to Hevner *et al.* (2004), there are two paradigms within the research in the information systems discipline. These paradigms are the behavioral-science paradigm and the design-science paradigm. The *behavioral-*

[9] CRIS = Comparative Review of Information Systems (e.g. Olle *et al.* 1983; Olle *et al.* 1986; Olle *et al.* 1988b).

[10] EMMSAD = Evaluation of Modeling Methods in Systems Analysis and Design.

science paradigm tries "to develop and verify theories that explain or predict human or organisational behaviour" (ibid p. 75). The *design-science paradigm* seeks "to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" (ibid p. 75). It is fundamentally a problem solving paradigm (Simon 1996), in which knowledge and understanding of a problem domain and its solution are pursued by the building and application of the designed artifacts. During the last decade the design-science paradigm has matured as a respectable and supported paradigm (e.g. Walls *et al.* 1992; March *et al.* 1995; Markus *et al.* 2002; Gregor 2002; Iivari 2003; Gregor *et al.* 2003; Hevner *et al.* 2004).

Our research clearly belongs to the design-science paradigm. We are pursuing the ontological framework (OntoFrame) and the methodical skeleton (MEMES), which both are design artifacts intended for the use of ME. To make the conception of the paradigm more concrete, we consider the research framework (Hevner *et al.* 2004) in Figure 3. We describe the main parts of the framework and characterize our study in the light of it.

The *research framework* is composed of three related main parts: environment, IS research and knowledge base. *Environment* "defines the problem space in which reside the phenomena of interest"(ibid p. 79). In our case the environment means purposes, actors, actions, deliverables, and tools involved in method engineering. From this environment arise the needs for the engineering of specific methodical support.

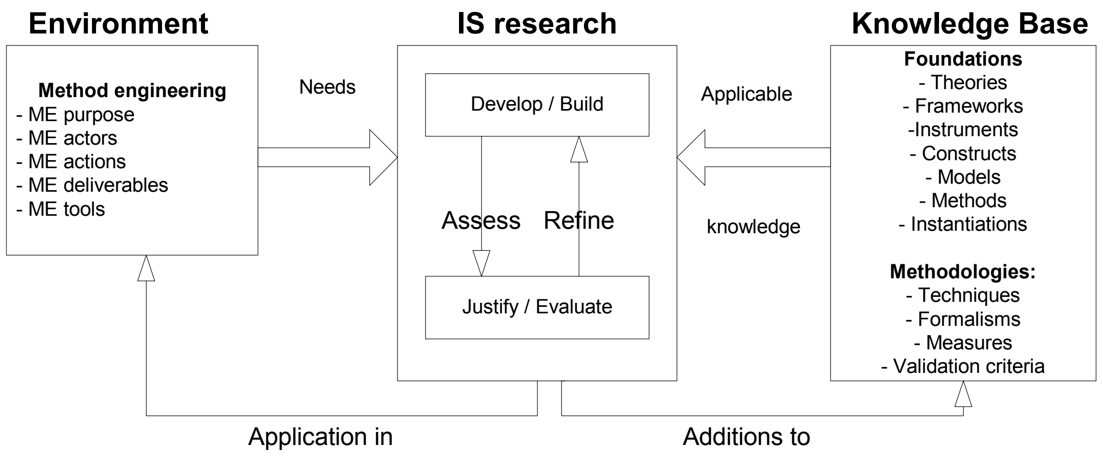


FIGURE 3 Research framework (cf. Hevner *et al.* 2004, 80)

Knowledge base "provides the raw materials from which and through which research is accomplished" (ibid p. 80). It is composed of foundations and methodologies. Foundations contain theories, frameworks, instruments, constructs, models, methods, and instantiations. For our study the foundations contain theories (e.g. semiotics, semantics, pragmatics, systems theory), frameworks for IS and ISD, meta models and meta meta models, modeling languages, ontologies, and so on. Methodologies provide guidelines to justify and evaluate the designed artifacts. These embrace methods for conceptual and empirical research, measures, and validation criteria.

IS research means the process by which a research effort is accomplished. It is composed of two complementary parts: development/build and justify/evaluate. The former means designing and, in some cases, implementing artifacts to be applied in the environment. The latter comprises actions with which designed artifacts are evaluated and justified.

Three main parts are related to one another, as denoted by arrows in Figure 3. The environment defines needs or "problem" for a research. The knowledge base provides applicable knowledge to be used, advanced and accumulated in the research. The IS research contributes design artifacts to be applied in the environment, thus benefiting it. The research also contributes to the knowledge base with new or improved frameworks, instruments, constructs, models, methods, etc.

March *et al.* (1995) and Hevner *et al.* (2004) distinguish between four kinds of design artifacts: constructs, models, methods, and instantiations. *Constructs* provide "the vocabulary and symbols used to define problems and solutions", thus impacting on the way in which tasks and problems are conceived (Hevner *et al.* 2004, 83). They constitute a conceptualization of a domain (March *et al.* 1995). *Models* use constructs to represent a real world situation. Models are composed of "propositions or statements expressing relationships among constructs" (March *et al.* 1995)[11]. *Methods* define processes and provide guidelines to perform tasks (Hevner *et al.* 2004; March *et al.* 1995). *Instantiations* "show that constructs, models, or methods can be implemented in a working system" (Hevner *et al.* 2004, 79).

Our research contributes two kinds of design artifacts: a construct and a method. OntoFrame is a comprehensive construct composed of ontologies, which provide vocabularies to conceive, understand, structure and represent phenomena in the sub-domains. MEMES is a method – according to the terminology of the design-science - containing prescriptions for the accomplishment of an ME effort.

Next, we apply three parts of the research framework to characterize our study. The environment of the research and needs for the study were discussed in Section 1.1 (Background and Motivation). We concluded that there is a need for a conceptual foundation and a methodical support. These needs were targeted to and "packaged" into the forms of the ontological framework and the methodical skeleton in Section 1.2. Concrete objectives of the artifacts were detailed in Section 1.3.

As the research domain of this study is very large, our knowledge base includes a large array of fundamentals and methodologies. We have made numerous surveys of the relevant literature, analyzed them for their applicability for our purposes, and compared them to our artifacts. The results from these surveys and analyses are reported in appropriate sections in the

11 Note that the meaning given by March *et al.* (1995) to the notion of a model differs from which we later associate to the notion in this study.

thesis. For example, to establish the contextual approach in Chapter 4, we made a comprehensive search for theories addressing the notion of a context, and built, based on them, a foundation for distinguishing between seven contextual domains.

The research process itself has been very complicated and iterative. We will describe the process and research methods applied during the process in Section 1.5. Hevner *et al.* (2004) suggest seven guidelines for making and reporting on research based on the design-science paradigm. We will apply guidelines to consider problem relevancy, research contributions, design evaluation, and limitations as the key research qualities of our study in Section 1.6.

1.5 Research Methodology

A *research methodology* consists of the combination of the process, methods, and tools that are used in conducting research in a research domain (Nunamaker *et al.* 1991, 91). In this section we first discuss the process by which our research has been conducted and then describe research methods that we have followed.

1.5.1 Research Process

To illustrate the process of this study, we use a simple grid of four subfields (Figure 4). The subfields are: ME practice, evaluation, ME method engineering, and ontology engineering. Our research process has progressed across the subfields in a cyclic manner. The process started with engineering ontological constructs and continued with putting them in practice and evaluating experience from that. In certain stage, we entered the sub-field of ME method engineering, reengineered ontologies, tested them in practice, and so on. The figure is not meant to portray an exact registration of all the stages of the process but rather a symbolic presentation of iterations between different kinds of research approaches and fields. In the following we first discuss our research process in each subfield and then summarize the description.

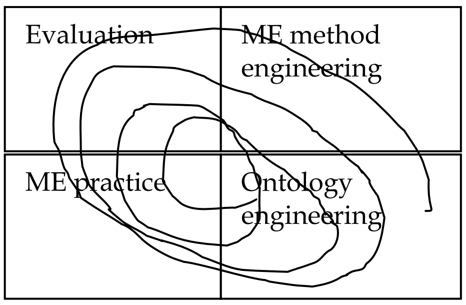


FIGURE 4 Research process

ME practice stands for all those efforts in which the researcher has been involved in developing an ISD method in practice. There are four projects that belong to this subfield. The first attempt was made, when we developed a language and a "design model" for conceptual data base design based on the linguistic approach (Leppänen 1984a). In the language essential pragmatic (i.e. a sender, a receiver, a sending time, a receiving time, a mode), semantic (i.e. entity type, relationship type, attribute, property type, time, constraints) and syntactical elements were distinguished. The design model was built from generic constructs (e.g. workflow structure, system decomposition structure, abstraction structures, problem solving structure). The model was composed of two layers: a frame layer consisting of generic concepts and constructs, and a core layer standing for instantiated concepts and constructs.

The second attempt was related to a large international Esprit[12] project, called OSSAD (Office Support Systems Analysis and Design) into which the research group (Vesa Savolainen, Mauri Leppänen) from the University of Jyväskylä was accepted in 1986. The other project partners were from France (D. Conrath CETME Aix-en-Provence, P. Dumas CETMA Toulon, G. Charbonnel CETMA Toulon), Italy (V. de Antonellis University of Milan, C. Simone University of Milan, G. de Petra IPACRI Rome, C. de Santis IPACRI, Rome), and Germany (S. Sorg IOT Munchen, E. Beslmuller IOT Munchen). The purpose of the project (1985-1989) was to develop a method for the analysis and design of office support systems. The researcher's role in the project was to comment on, and ideate, the conceptual foundation of the method, to contribute to some specific parts of the method, as well as to field test the method in a case organization (Leppänen *et al.* 1988; Leppänen *et al.* 1989a). The project engineered the comprehensive method that was published in the manual (Conrath *et al.* 1989) and in numerous articles (e.g. Beslmuller *et al.* 1986; Beslmuller *et al.* 1987; Conrath *et al.* 1988; Charbonnel *et al.* 1991; Conrath *et al.* 1992; Vincent *et al.* 1992; Conrath *et al.* 1999; Savolainen 1999).

The third attempt dates back to 1988/1989 when the researcher participated in a large consulting project to plan an information technology and service strategy for the city of Jyväskylä. For the project, a method, called SPITS (Strategic Planning of Information Technology and Services) (Leppänen *et al.* 1991), was first engineered. In the method, the process of strategic planning is decomposed into three parts: analyzing the service strategy, planning the IS strategy, and planning the implementation of the IS strategy. For each part, several design techniques were selected from existing methods and customized to fit the needs of the project. Researchers acted as teachers, mentors and partly as designers in the project. Experiences of the applicability of the method were collected by interviews, although they were not reported in public.

The fourth attempt to method engineering in practice concerned engineering a method for database application design for the purposes of teaching. The first version of the method was developed during 1985-1993 (Leppänen 1993). The method was based on a view of the centralized

[12] Esprit = European Strategic Program for Research in Information Technology.

architecture and the use of the Ingres database management system. The second version was engineered for web-based database application design with Oracle 8i (Leppänen 2001). Both of the methods were constructed by selecting and customizing models and techniques from existing methods. The methods were used by students in their designing and implementing small-scale database applications in project groups of 2 – 4 members. The course has been lectured seven times so far, and about 100 projects have been accomplished with the methods.

Ontology engineering means establishing a conceptual foundation for conceiving, understanding, structuring and representing phenomena in the domains of IS, ISD, ISD method, ME, and ME method. The first concepts and constructs were defined for the core part of OntoFrame in Leppänen (1984a). The work continued with elaborating definitions of core concepts, abstraction structures and abstraction strategies (Leppänen 1984b). Later the ontology was extended and refined by establishing a contextual approach, which recognizes concepts within seven contextual domains. This work laid the ground for the IS ontology. In the last stages, the ontological framework was extended to cover the ISD domain and the ME domain as well. During the cyclic process, definitions and categorizations of new and old concepts and constructs have been modified and elaborated several times.

Evaluation comprises two kinds of actions: (1) making analyses and comparative reviews of definitions, classifications, models, frameworks, and methods presented in the literature, and (2) reflecting from the practice of method engineering. The purpose in the first case has been to learn existing knowledge in the research sub-domains and compare it to artifacts produced in each stage of this study. Examples of the early work on these issues are a classification of research methods (Leppänen *et al.* 1988), an analysis of the OSSAD method (Leppänen 1989a), and a conceptual analysis of socio-technical methods (Leppänen 1989b). The OSSAD method was analyzed on the basis of its core concepts and abstraction constructs. Socio-technical methods were analyzed with the concepts of seven contextual domains. Later, several comparative analyses have been conducted related to the IS ontology, the ISD ontology, ISD method ontology, and the ME method ontology. Also suggestions for ME strategies, ME approaches, ME techniques, and ME procedures have been evaluated. The results of these analyses are reported in this study. The evaluation of experience from the application of the artifacts of this work has been made with two empirical methods. First, we applied retrospective analysis (cf. Fitzgerald 1991) to investigate documents and notes from the OSSAD project and to elicit findings and lessons to be used in elaborating the artifacts. Second, we applied the reflection-in-action approach (Schön 1983) to constructing MEMES with applying MEMES itself. Results from these analyses are reported in this thesis.

The first serious attempts to contribute to ME method engineering were made in 1994, when a technique for describing, analyzing and refining conceptual relationships of ISD techniques and ISD models within an ISD method was developed (Kinnunen *et al.* 1994; Kinnunen *et al.* 1996). This technique was applied to test the interoperability between techniques of the SPITS method (Leppänen *et al.* 1991). Next, activities, benefits and pitfalls of metamodeling were distinguished based on an early version of the ontological framework (Leppänen 1994). Later, a model integration technique, analogous to a view integration technique (Batini *et al.* 1992), was developed to ensure the consistency in integrating an ISD method from method components (Leppänen 2000). Also a set of ME approaches were defined. In the last few years, the research was extended to cover more comprehensive methodical and procedural structures of ME. This work was based on the established OntoFrame and experiences got from ME practice.

Engineering a methodical skeleton itself was, in fact, an instance of ME effort. This effort did not aim to build an ISD method but an artifact that is a kind of pre-stage of an ME method. Nevertheless, the engineering process of MEMES should apply exactly the same structural and dynamic building blocks that are included in MEMES itself. This observation made us to decide to systematically follow MEMES while the incremental and cyclic process went along. Instant feedbacks substantially contributed to the theoretical part of method engineering.

To summarize, our research process has been highly iterative, crossing four sub-fields. The work has been interplay between theoretical considerations, analysis of literature, empirical work and its evaluation. The process reflects a learning cycle (Checkland 1981, 254) in which each deliverable (i.e. an ontology, an ME technique, and a methodical skeleton) is created, applied, learned from, and refined.

1.5.2 Research Methods

To support the research process we have applied two kinds of research methods. The first method is conceptual. By this method we have defined OntoFrame and MEMES and conducted a large set of comparative analyses of the existing literature to show how our framework differs from other artifacts (i.e. ontologies, meta-models, frameworks, and frames of reference). Second, we have applied several empirical methods. During the OSSAD project, a field test was carried out to collect experiences on applying the OSSAD method in practice (Leppänen *et al.* 1988). The application of the SPITS method in a large municipal organization was carried out as a case study. Experiences from users were collected here too. The use of the database application design method by students corresponds to a laboratory experiment in a sense that all the project groups (ca. 100 groups in seven years) applied the same method (in two versions) in application design. However, no specific research setting was defined neither used in these situations. Nevertheless, the researcher got a very "rich" picture of the applicability of the method during the guidance hours that were provided for groups in each of 5 – 6 phases, as well as through exhaustive documentations produced by each project group.

We do not explicitly report on the results from the aforementioned investigations in this study. Instead, we have selected two efforts to be analyzed here in more depth. They are the OSSAD project and the MEMES effort. We apply the retrospective analysis (cf. Fitzgerald 1991) to make sense of the process and outcomes of the OSSAD project. In the analysis we use MEMES as a frame to find out essential features and approaches of the OSSAD project, to discover possible problems, and speculate how these could have been avoided with a methodical support such as MEMES. Second, we apply the retrospective analysis to describe and evaluate, how MEMES performs as a prescription for the engineering process of MEMES itself. The process is conducted according to reflection-in-action approach (Schön 1983). The major characteristics of this approach are a fluid and reflective process model and the use of a generative metaphor in situation framing (Heiskanen 1995).

1.6 Key Research Qualities

Hevner *et al.* (2004) suggest seven guidelines for the design science to be determined "when, where, and how to apply each of them in a specific research project" (ibid p. 82). Guidelines are based on the view according to which design science is inherently a problem solving science. These guidelines concern design artifacts, problem domain, contributions, design evaluation, research contributions, design as a search process, and communication of research. In this section we apply the guidelines to consider our study from the viewpoint of design science. We discuss problem relevancy, research contributions, design evaluation, and limitations of this study.

1.6.1 Problem Relevancy

According to Hevner *et al.* (2004), the objective of research should be "to acquire knowledge and understanding that enable the development and implementation of technology-based solutions to heretofore unsolved and important business problems" (ibid p. 84). The most essential words in the statement are 'unsolved' and 'important' problems.

We described the background and the motivation of our study in Section 1.1. We arrived at the following conclusions: *There are severe problems in methods and method use in practice, and more methods with new approaches and features are required to provide methodical support for emerging application areas. On the other hand, method engineering in practice suffers from serious problems, which complicate and hamper the analysis, design, and implementation of ISD methods. The current ME literature provides a large array of ME artifacts, but they are inadequate to overcome the problems. No single artifact among them can be seen to come even close to the ME method as the notion is generally understood.* Concluded from the above, we argue that there are clearly unsolved problems, which we are addressing in our study.

The ISD community has divergent conceptions and opinions of the nature, role and significance of a method in ISD (Baskerville *et al.* 1992; Wastell 1996; Pfleeger *et al.* 1997; Truex *et al.* 2000; Fitzgerald *et al.* 2002; Chang *et al.* 2002). On the basis of the large review made in this study among the ISD literature, we argue that methods, in one form or another, are needed in practice. Considering a huge number of information systems, their diffusion into a vast spectrum of application fields, and their significance to business, we can conclude that getting some of the problems solved in method engineering is clearly of considerable importance. Later in this study we show the problem areas to which we have in particular contributed.

1.6.2 Research Contributions

Hevner *et al.* (2004) argue that design-science research "must provide contributions in the areas of the design artifact, design construction knowledge (i.e. foundations) and/or design evaluation knowledge (i.e. methodologies)" (ibid p. 87). This study contributes to foundations (see Figure 3). We suggest the extensive ontological framework (OntoFrame) and the methodical skeleton (MEMES) to be used in ME. Contributions can be evaluated in terms of their novelty, generality, and significance (Hevner *et al.* 2004). Here we consider our contributions with the first two criteria. The third criterion is relevant for considerations at the end of this study, after the design artifacts have been described in more detail.

We will show in the following chapters that at present there is no unified, coherent and theory-based framework that would support the understanding and structuring of phenomena in the sub-domains. Furthermore, there is a large set of aspects in individual sub-domains that are poorly understood and/or covered in artifacts. Our response to this situation is to apply the principles of ontology engineering in building a coherent and unified framework. Furthermore, we anchor the framework upon generic and relevant theories (e.g. semiotics, semantics, pragmatics, systems theory) to ensure the comprehensiveness of the framework. The main "yardstick" used in the evaluation of comprehensiveness is, resulted from our contextual approach, how well the framework covers contextual features of the subject matters. These approaches make our ontological framework (OntoFrame) quite novel and different from all other suggestions.

MEMES stands out from the others, first due to the approach deployed to engineer it, and second due to its structure and contents. Most of the ME artifacts at present have been engineered by externalizing ME conventions in practice. This is the inductive approach to the ME research. The ME practice is, however, still in its infancy, and so many of the established conventions are undeveloped and not worth of writing into prescriptions in ME artifacts. We have adopted another approach. Still taking into account the experience and lessons from ME practice, we have started to engineer the main functional and modeling structures of MEMES on the grounds of the ontological framework. Hence, we apply the deductive approach to the ME research. This way we are able to look at ME efforts with "new eyes" and find solutions that have not been discovered earlier. As mentioned above, the ME literature provides inadequate support to ME efforts. Although MEMES is not a complete method, it nevertheless covers the most essential parts of ME. For these parts it provides both generic specifications in meta models and specific guidelines for the accomplishment of ME actions. Many of these suggestions are novel.

OntoFrame and MEMES are intended for the general use. OntoFrame is composed of a large number of ontologies, each of which can be used to conceive, understand, structure, and represent phenomena in an appropriate sub-domain. Also in the selections of approaches and views underlying the ontologies we have aimed for generality. MEMES is a methodical skeleton that provides generic support for ME, whatever ME strategy (i.e. "from scratch", integration, adaptation) is followed. Guidelines and steps contained in MEMES are described in the way, which enables their deployment in a large variety of situations.

1.6.3 Design Evaluation

Hevner *et al.* (2004) state that "the utility, quality, efficacy of a design artifact must by rigorously demonstrated via well-executed evaluation methods" (ibid p. 85). In this section we consider the evaluation of OntoFrame and MEMES in terms of verification and validation.

The outcomes of this research are conceptual and highly abstract. Verification and especially validation of these kinds of artifacts have been recognized to be problematic, both in ontology engineering (e.g. Gomez-Perez 1995; Gruninger *et al.* 1995; Guarino 1997; Shank *et al.* 2003) and in method engineering (e.g. Fitzgerald 1991; Schipper *et al.* 1996). Therefore, we first discuss the notions of validation and verification in general and then describe how our outcomes are verified and validated.

In the ME field, validation has been discussed in relation to e.g. models and techniques. According to Flon Arnesen *et al.* (2002), the validity of a model "means that all statements made in the model are correct relative to the domain" (ibid p. 68). Schipper *et al.* (1996) state that validation of a modeling technique "is a means to determine whether the technique serves its intended purpose" (p. 1). According to Fitzgerald (1991) the term 'validation' is used "to mean the justification of the technique in terms of its power, effectiveness and practicality in relation to its purpose and objective" (p. 659). He emphasizes in particular well-foundedness, soundness, and applicability of a technique to the case or circumstances.

In this study, we distinguish between two dimensions to the evaluation of the artifact: validation and verification (Weinberger *et al.* 2003). *Validation* means a process of seeking evidence of the applicability of the artifact to the predefined intentions of use (cf. Schipper *et al.* 1996, 1). The features of the artifact affecting the applicability are called external features. *Verification* means a process of acquiring evidence that the artifact fulfills the objectives related to internal features of the artifact. Examples of internal features are consistence and coherence of the concepts (Weinberger *et al.* 2003). Hence, in both cases, it is a question about the evaluation of the features of the artifact based on the predefined objectives. In the following we first use the term 'evaluation' when discussing the assessment of the properties of the artifact in general. After that, we show in which case evaluation means verification and in which case it is a question about validation of the artifact in our work.

Evaluation should start with describing the researcher's intentions for the use of the artifact (Schipper *et al.* 1996, 7). Next, objectives in terms of internal and external features should be derived. That is exactly what we have reported on in Section 1.3. There are many ways to derive and express the features (cf. Sol 1983, 4). In the simplest case, features are presented in a list without any clear connection to intentions. There are many examples of this kind of lists in the literature (see e.g. Olle *et al.* 1983; Olle *et al.* 1986). A better way is to prioritize the features based on the intentions or other explicated grounds. The third way is to use an "idealized" artifact as a "frame of reference" of assessments (Sol 1983, 4). An idealized artifact is usually developed by deriving it from features of existing artifacts, e.g. including "best properties" of the existing artifacts in the artifact. Instead of, or besides, the existing artifacts, derivation can be based on some underlying theory (cf. the socio-cybernetic framework in Iivari *et al.* (1983)).

Evaluation can be done conceptually or empirically. Conceptual evaluation can be conducted in two ways: (a) focusing on the artifact only, or (b) comparing it to some other artifact(s). In the first case evaluation concerns the artifact and/or the process of engineering. In a comparative analysis, the artifact is evaluated by comparing it to one or more existing and comparable artifacts (e.g. Henderson-Sellers *et al.* 1999a; Kabeli *et al.* 2002; Kavakli *et al.* 2003). This way one tries to prove that the artifact is "better" than the others in some relevant terms (e.g. comprehensiveness). Conceptual evaluation can be supported by formalization. For instance, consistence and coherence of the artifact can be more easily proven through formal definitions and axioms (ter Hofstede *et al.* 1998, 521). Besides enforcing the defining of concepts into a more rigorous direction, a successful completion of formalization itself gives an evidence of 'formalizability' of the ideas underlying the artifact (ter Hofstede *et al.* 1992; ter Hofstede *et al.* 1998). Yet stronger evidence can be got implementing the artifact into e.g. a software tool (e.g. Harmsen 1997; Gupta *et al.* 2001). Successful implementation provides evidence of the implementability of the ideas included in the artifact. However, it is important to point out that even a successful implementation of the artifact into a software tool does not prove that the artifact is useful. To evidence that requires empirical tests of the tool.

Empirical evaluation is based on experiences got from using the artifact in one or more contexts. Contexts can differ from each other in four ways. First, the context can be hypothetic or real. A hypothetic context may concern hypothetic and often simplified problems to be solved in a laboratory or the like. A real context is a working situation solving real problems in an organization. Second, users of the artifact from whom experiences are collected can be developers of the artifact (e.g. in Fitzgerald 1991), students, or real workers in an organization. The strongest evidence can be got from real workers (e.g. in Moynihan (1996) and Grant *et al.* (2003)) but due to problems in involving them, it is common to recruit students into the work (e.g. Shoval *et al.* 1997; Chaves *et al.* 1996). Third, the artifact can be deployed as a whole or for some part only in a context. If the artifact is large, like a method, it is common to concentrate on features related to some part (e.g. Moody 2003a). Fourth, evaluation in a context can be carried out as one time action or as a sequence of assessments. In the latter case, evaluation can be included as an inherent part of evolutionary engineering of the artifact (e.g. Tolvanen 1998).

Next, we consider, based on the taxonomies and definitions given above, how the features of OntoFrame and MEMES are evaluated. We apply two kinds of research methods in this work. The first method is conceptual and it concerns verification of the qualities. The second method is empirical and it is used in validation. In what follows we discuss verification and validation of the framework and skeleton with relation to the objectives stated in Section 1.3.

To support the achievement and evaluation of clarity, consistency and coherency of the concepts and constructs within OntoFrame, we use semi-formal meta models in presenting the ontologies. In addition, we make cross-checking among the definitions of the concepts. Generality of the framework is supported by the stratified structure of the framework in which specialized concepts are derived from the fundamental concepts. We demonstrate the suitability of OntoFrame to relating concepts of different approaches by numerous comparisons. Generativeness and modularity result from the process by which we establish the framework. Extensibility is furthered by generativeness and modularity. To obtain strong evidence of the naturalness of the framework would require tests in real contexts, with real workers, and we do not do it here. Instead, we argue for the naturalness of OntoFrame by referring to the nature of theories underlying the framework. The fact that the framework is theory-based becomes evident from the use of theories.

All the evaluation means discussed above are examples of verification of the internal features of OntoFrame. In contrast, the evaluation of comprehensiveness and applicability presumes validation. Comprehensiveness is evaluated in two ways: (a) founding on and comparing to the relevant theories, and (b) making comparative analyses of existing artifacts. OntoFrame is aimed to be applicable for three kinds of intentions. As to the analytical intentions, we refer to the results from the comparative analyses to claim for applicability. To gain evidence from the applicability of OntoFrame in the constructive sense, OntoFrame is applied to construct MEMES. To validate the applicability of OntoFrame in the descriptive sense would require empirical tests. These are not made in this study.

For MEMES it has been stated that it has to be based on a solid and sound view on the domain area. Reaching this objective is aided by anchoring MEMES to the theory-based ontological framework. The fact that MEMES is component-based becomes evident by viewing its structure. Validation of the applicability of MEMES is carried out conceptually and empirically. Conceptual validation is carried out with a comparative analysis of existing ME artifacts (cf. the analytical intention of use). Empirical validation is accomplished in two ways: (b) by the retrospective analysis of the process and outcome of the OSSAD project (cf. the framing intension of use), and (b) by the retrospective analysis of the process of applying MEMES in the engineering of the skeleton itself according to the reflection-in-action approach (Schön 1983) (cf. the constructive intention of use). In addition, experience from all those efforts in which the researcher has been involved has been utilized in engineering MEMES.

1.6.4 Limitations

This thesis is quite large and addresses a number of sub-domains. Even so there are some limitations that we have made during the study. First, both OntoFrame and MEMES could have been refined further. We have consciously excluded some less essential parts from the lower levels of OntoFrame in order to keep the size of the framework reasonable. Formalization of the framework could have been augmented by more axioms to specify rules and constraints more precisely. The scope of MEMES has been limited to cover three ME workflows (i.e. the ISD method (ISDM) requirements engineering, the ISDM analysis, and the ISDM evaluation) out of five. Although this is enough to have an overview of method engineering and to cope with decisions on methodical structures on a general level, the skeleton could have been extended to deal with the other workflows (i.e. the ISDM design and the ISDM implementation) with equal weight. Also more procedures and guidelines, for instance, for alternative ways of engineering, could have been included in MEMES.

Second, there are some limitations in our research methodology. One of the most severe limitations in this study is that the method skeleton is not properly tested in a real context. We do have heavily utilized experience on ME practices in which we have been involved, as described above. We have also applied MEMES in a step-by-step process of engineering of the skeleton itself. But these contexts do not fully satisfy requirements presumed from the empirical validation of MEMES in terms of applicability and usefulness. For this reason, the skeleton should be first elaborated into a specific ME method, and then this ME method should be used in an ME effort to engineer a specific ISD method. This way we could obtain a more reliable conception of how MEMES performs. In fact, to have a strong evidence of the applicability and usefulness of MEMES, validation should also address the use of the ISD method engineered in the development of some specific information system. Due to the largeness of this study, it was not possible to include these kinds of case studies in the thesis. They will be subjects of future research.

1.7 Structure of the Dissertation

The thesis is organized into thirteen chapters. The structure of the chapters 2 – 13 is illustrated in Figure 5. The arrows between the boxes express how the subject matters dealt with in the chapters are based on the considerations in preceding chapters. In Chapter 2 we will present an overview of OntoFrame. After motivating needs for the framework and describing its theoretical foundations, we will outline the modular structure of the framework. For each part the intention and the domain are described. After that we will portray the approach and the process applied in ontology engineering. Finally, we present a comparative review of the most relevant artifacts in the IS literature.

Chapter 3 defines the fundamental concepts of the research domain as the core ontology. The core ontology is composed of seven sub-ontologies: the generic ontology, the semiotic ontology, the intention/extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology. After reviewing the relevant works and describing the overall structure, we will present meta-models with concept definitions for each of the sub-ontologies. Concepts are widely compared to those presented in the literature. In addition, we will present a comparative analysis of two most-well known artifacts, namely the Frisco framework and the BWW model. In the analysis we compare the objectives, ontological positions, basic structures, coverage, and emphases of the artifacts with the core ontology.

Chapter 4 presents the context ontology. First, we will define the contextual approach and characterize its application domain and objectives. Next we will review the literature to establish a theoretical basis for the contextual approach. Deriving from the selected theories (e.g. semiotics, semantics, pragmatics and some theories of human and social action), we will elaborate the notions of a context and a contextual domain. For each contextual domain, we will present a meta-model and define their contextual concepts and constructs. In addition, we will define inter-domain and implicit relationships.

Chapter 5 specifies the layer ontology. First, we will define the basic concepts related to information and information processing. We will also distinguish between the information system, the object system, the utilizing system, and the controlled system. Next we will recognize the primary actions and the development actions in information processing. Deriving from this dichotomy, we will define the system of four processing layers: information system, information system development, method engineering, and research work. Each layer will be characterized from the teleological, functional and structural viewpoints. The contents of and relationships between the contexts positioned at the layers will be discussed, and the notions of the utilizing system and object system will be specialized to concern each of the processing layers.

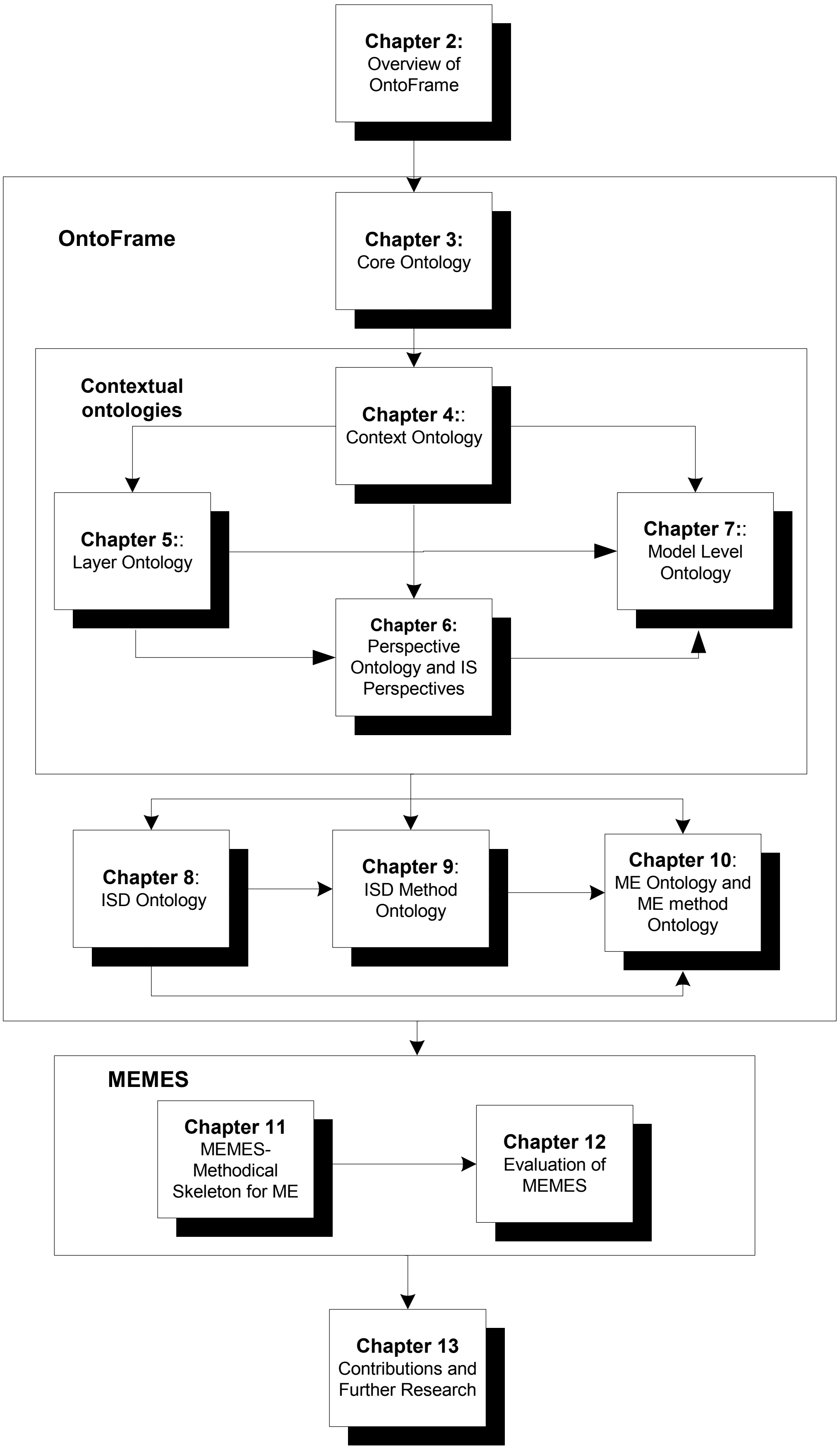

Chapter 2: Overview of OntoFrame
OntoFrame
Chapter 3: Core Ontology
Contextual ontologies
Chapter 4:: Context Ontology
Chapter 5:: Layer Ontology
Chapter 7:: Model Level Ontology
Chapter 6: Perspective Ontology and IS Perspectives
Chapter 8: ISD Ontology
Chapter 9: ISD Method Ontology
Chapter 10: ME Ontology and ME method Ontology
MEMES
Chapter 11 MEMES-Methodical Skeleton for ME
Chapter 12 Evaluation of MEMES
Chapter 13 Contributions and Further Research

FIGURE 5 Structure of the thesis

Chapter 6 addresses the perspective ontology. We will start with defining the perspective ontology by establishing the system of perspectives along particular dimensions and by defining the contents of five perspectives (i.e. systelogical, infological, conceptual, datalogical, and physical perspectives). We will consider how the perspectives are applied at the four processing layers. Next we will define the IS perspectives and relationships between them. Finally, we will present a comparative analysis of IS perspectives suggested in the IS/ISD literature. The analysis is composed of three parts, giving an overview of and revealing the conceptual contents and detailed concepts of the perspectives.

Chapter 7 presents the model level ontology. Here we will first define the notions of model and modeling and present the main classifications of the models. Second, we will extend our considerations to concern models at meta levels. We will also derive classifications of models and meta models based on the contextual domains and the perspectives. Third, we will present a comparative analysis of conceptions about the meta levels in the ISD literature. Fourth we will examine in more detail how the contextual ontologies are related to one another.

Chapter 8 addresses the ISD ontology. We will start with discussing and classifying the ISD paradigms and the ISD approaches and give a comprehensive definition of ISD. Second, we specify the first part of the ISD ontology, which is composed of meta models and concept definitions within four ISD domains (i.e. ISD purpose domain, ISD actor domain, ISD action domain, and ISD object domain). Also an overview of the inter-domain relationships will be provided. Third, we will present the second main part of the ISD ontology, which consists of four ISD perspectives. The perspectives are the ISD systelogical perspective, the ISD infological perspective, the ISD conceptual perspective, and the ISD datalogical perspective. Also the inter-perspective relationships will be discussed. Fourth we will make a comparative analysis of artifacts (i.e. frameworks, meta models and the like) presented in the literature. The purpose of the analysis is to obtain an overview of the artifacts, to find out how they differ from one another, and compare them with the ISD ontology in terms of comprehensiveness and focus.

Chapter 9 presents the ISD method ontology. First we will consider why the ISD methods are actually needed and used in practice by reviewing empirical studies. Second, we will discuss the difference between the terms 'methodology' and 'method'. Third, we will delineate the concept of the ISD method as a 'carrier' of ISD knowledge and specify basic classifications of ISD methods. Fourth, we will define seven fundamental views from which the ISD method can be conceived, and present an integrative definition of the ISD method that highlights the essential features of the method from all seven views. Fifth, we will establish the ISD ontology that is composed of parts corresponding to the seven views. Sixth, we will apply the ISD method ontology to consider, from a larger perspective, a range of methodical support and specify a taxonomy of methodical artifacts. We will also discuss the criteria for acknowledging an artifact as an ISD method. Seventh, we will make a comparative analysis of frameworks and categorizations of the ISD methods proposed in the literature. Eighth, we will consider the notion of method component in more detail. We will define the notion, present a classification scheme and specify a contextual interface of the method component. We will also discuss the integration of method components, illustrate the discussion with examples, and make a comparative analysis of the conceptions of the method component in the literature.

Chapter 10 defines the ME ontology and the ME method ontology. First, we will describe needs for method engineering and reasons behind them. Based on a short literature survey we will present basic classifications of ME strategies and ME processes and derive a fundamental categorization of ME contexts from them. We will also provide a definition of the ME context that integrates various contextual aspects of method engineering. Next we will present the first main part of the ME ontology addressing four ME domains. For each ME domain the concepts and constructs will be defined and described in ME meta models. After that we will provide the second main part of the ME ontology including four ME perspectives. Finally we will define the notion of ME method and derive the ME method ontology from the ISD method ontology established in Chapter 9.

Chapter 11 presents the methodical skeleton for ME. First, we will argue for the needs for methodical support to ME and define MEMES in terms of its intention, basis and contents. Also relations between MEMES and OntoFrame will be demonstrated. Second, we will derive the views and structure of describing MEMES in this work. Third, we will describe the background and application area of MEMES and state the goals for MEMES. Fourth, we will present the overall structure of MEMES in terms of ME workflows. Fifth, we will describe the approaches and steps of three ME workflows (i.e. the ISD method requirements engineering, the ISD method analysis, and the ISD method evaluation).

Chapter 12 provides evaluations of MEMES. First, we will apply MEMES itself to make plans for the evaluation. Second, we will report on the retrospective analysis and reflection-in-action approach applied to evaluate MEMES in two ME efforts (i.e. the OSSAD project and the MEMES effort). The OSSAD project engineered a method for office support systems analysis and design (Conrath *et al.* 1989). The MEMES effort means the process of engineering MEMES itself. Third, we will use MEMES as an analytical framework to make a comparative analysis of those artifacts, which are aimed to provide methodical support to ME.

In Chapter 13 we will summarize the results of the thesis, discuss their implications to research and practice, and describe directions for future research.

2 OVERVIEW OF ONTOFRAME

Method engineering needs a comprehensive and uniform foundation, a kind of conceptual "platform", upon which conceptions and representations of an ISD method and a process of engineering an ISD method can be established. The purpose of this chapter is to present an overview of such a foundation. We call it the ontological framework, and shortly OntoFrame.

This chapter is organized as follows. In Section 2.1 we discuss needs for an ontological framework. Next, we consider the suitability of metamodeling and ontology engineering for theoretical grounds of OntoFrame. In Section 2.3 we describe the domain and overall structure of Ontoframe. In addition, we discuss and select forms in which OntoFrame will be presented. In Section 2.4 we delineate the approach and the process by which OntoFrame has been engineered. In Section 2.5 we make a comparative review of the relevant literature in order to give an overview of the state of the art and to show how our framework differs from the others. The chapter ends up with a summary.

2.1 Needs for an Ontological Framework

Becoming aware of phenomena in reality and understanding their meaning necessitates that a human being has a mental structure that gives a means to focus, structure, and organize perceptions (cf. Koskinen 2000, 77). The more abstract thinking a work requires, the more necessary it is to hold and utilize well-defined concepts in doing the work. Concepts are not separate but constitute webs of associations in which each concept has its specific roles and constraints. These webs are called conceptual frameworks. A *conceptual framework* means a structure of named and defined concepts, views, etc. by means of which an individual or group conceives and communicates ideas (cf. Webster 1989). It is a kind of intellectual structure, which determines which phenomena are meaningful to us and which are not.

Our ontological framework is a conceptual framework. Its purpose is to provide a unified conceptual foundation for conceiving, understanding, structuring, and representing phenomena in ME. But what is actually ME? What are those features it holds, which we should become aware of and which we should be able to communicate about? That is what we try to figure out in this section, in order to grasp the baseline and premises for the ontological framework.

The most essential individual thing in ME is an ISD method. It is the main target of engineering. An ISD method is a highly abstract and multifaceted notion about which there are quite different conceptions. It is 'externalized' in the form of a manual, a textbook, or a Power Point slice show, and marketed as written documentation, consultant services, part of a CASE tool, etc. This concrete side of an ISD method is easy to understand and agree on. But what are the nature and contents of an ISD method? Which kinds of phenomena do texts and diagrams in manuals, textbooks, and slices refer to? What actually happens in ISD and what is the role of an ISD method in this process? These are examples of questions that should be asked when engineering an ISD method, and not only asked but also pursued to obtain "right" answers. As said above, in the literature there are quite different views (answers) on these issues. In what follows, we give examples of these views to demonstrate the variety of conceptions and to motivate our search for a unified conceptual foundation.

An ISD method is commonly considered to contain collective knowledge and experience of ISD (Tolvanen 1998; Fitzgerald *et al.* 2002; Schönström *et al.* 2003). There are, however, varying conceptions on a degree to which an ISD method as a "knowledge base" can convey and provide knowledge for ISD developers (Wastell 1996; Wordsworth 1999; Truex *et al.* 2000). Those that are the most optimistic think that all the ISD knowledge can be thought to reside outside the ISD practice – in books and in learned institutions, whereas some others see the most relevant knowledge reside in ISD practice itself (cf. method-in-action). Some part of the community appreciates formal methods (e.g. Jones 1986; Pfleeger *et al.* 1997; Wordsworth 1999), whereas others stand on the side of informal and even amethodological approaches (Wastell 1996; Ciborra 1999; Truex *et al.* 2000). This confusion is not much relieved by results from empirical studies of the nature and significance of ISD methods to ISD work (cf. Stolterman 1992; Hardy *et al.* 1995; Fitzgerald 1996a; Chatzoglou 1997; Hidding 1997; Rahim *et al.* 1998; Avison *et al.* 2003), as they are partly contradictory.

An ISD method is defined to be an approach (e.g. Brinkkemper 1996; Truex *et al.* 2000; Russo *et al.* 1996), a way of accomplishing (Kruchten 2000), a description of a technique (Hirschheim *et al.* 1995), a procedure (Kitchenham *et al.* 1999), and a system (Jones *et al.* 1988; Krogstie 1995). From the structural viewpoint, an ISD method is seen as "a collection of procedures, techniques, tools, and documentation aids" (Avison *et al.* 1995a, 10), as "a collection of procedures and heuristics" (Nuseibeh *et al.* 1996, 269), as "an organized collection of concepts, methods, beliefs, values and normative principles" (Iivari *et al.* 1998a, 165), and as "a system of rules, techniques, and tools" (Krogstie

1995, 13). Implied from these samples, there seem to be quite different conceptions about the structure of an ISD method. Which of the aforementioned parts are mandatory to acknowledging an artifact as an ISD method is also an open question.

Having considered an ISD method from the structural viewpoint, we next turn to examine its contents. The view of what an ISD method contains depends on what kind of effort we think ISD is, and that, in turn, is affected by adopted paradigmatic assumptions (Hirschheim *et al.* 1989; Iivari 1991; Hirschheim *et al.* 1992a), schools of though (Iivari 1991; Iivari *et al.* 1998a), and ISD approaches (Wood-Harper *et al.* 1982; Lyytinen 1986; Hirschheim *et al.* 1995). ISD can be seen, for instance, as a transformation process (e.g. Lundeberg *et al.* 1981; Lehman 1984; Moynihan 1993; Tracz *et al.* 1993; Jacobson *et al.* 1999), as a decision making process (e.g. Jarke *et al.* 1990; Wild *et al.* 1991; Grosz *et al.* 1997), as a problem solving process (e.g. Bodart *et al.* 1983; Sol 1992; Blum 1994; Jayaratna 1994), as a learning process (e.g. Iivari *et al.* 1987; Ramesh *et al.* 1994; Lyytinen *et al.* 1999), or as a cooperative process containing negotiations, bargaining, power and social interactions (e.g. Keen 1981; Markus 1983; Newman *et al.* 1990; Chang *et al.* 2002). There are also ISD approaches, which differ from one another in how they emphasize certain parts in ISD or how they structure the ISD process (cf. Wood-Harper *et al.* 1982; Bracchi *et al.* 1984; Barbic *et al.* 1985; Graham 1989; Vessey *et al.* 1994). The "world views" reflected by these approaches seem to be so far from one another that it is difficult to distinguish what is common to them. Perhaps one common denominator is a conception about ISD as a context where people carry out specific actions, with the support of tools, to produce IS models and their implementations. If this is the case, we have still a problem: what is a context?

Context plays an important role in many disciplines, such as in formal logic, knowledge representation and reasoning, machine learning, computational linguistics, organizational theory, sociology, neurology, etc. Also in the IS field there are several approaches that consider the notion of a context important (e.g. Searle 1979; Auramäki *et al.* 1988; Sowa 1984; Engeström 1999; Kuutti 1991; Motschnig-Pitrik 1995; Motschnig-Pitrik 1999; Rolland *et al.* 1995). Which view of these disciplines should we prefer, and what are the essential ingredients of a context according to such a view?

Developing information processing in an organizational context is a complex endeavour. To manage the complexity, different viewpoints, levels of abstraction, and perspectives have been specified and deployed (e.g. Welke 1977; Olive 1983; Essink 1986; Olle *et al.* 1988a; Iivari 1989a; Avison *et al.* 1990; Sol 1992; Sowa *et al.* 1992; van Swede *et al.* 1993; Freeman *et al.* 1994; ISO 1996; Booch *et al.* 1999). With a set of well-established perspectives it is easier to determine scopes within which development issues are taken into consideration in a step-by-step fashion. But which of the defined viewpoints and perspectives are beneficial and suitable for the premises and selections we have made before?

ISD work produces and deploys a large variety of IS models. What is actually a model? How is it produced and represented? What kinds of models exist and for which kinds of purposes they are suitable? Different kinds of conceptions about these issues have been suggested (c.f. Minsky 1965; Checkland 1981; Gigch 1991; Wijers 1991; Krogstie 1995; Hirschheim *et al.* 1995; Falkenberg *et al.* 1998; Ramesh *et al.* 2001; Rosemann *et al.* 2002). Models appear at several meta levels, and surprisingly, divergent conceptions exists also about sets of meta levels (e.g. Bergheim *et al.* 1989; ISO 1990; Brinkkemper 1990; Gigch 1991; Wijers 1991; Heym *et al.* 1992a; Jarke 1992; Harmsen 1997; Falkenberg *et al.* 1998; OMG 2002).

We have raised, up till now, a number of questions to which answers should be sought – and found - in order to establish a coherent and consistent conceptual foundation. To succeed in this task, still more fundamental issues must, however, be considered first. These issues are related to views of and conceptions about reality in general: How do we conceive structural and behavioral features of reality? How do we present our conceptions, and how do we use abstraction mechanisms to derive generic conceptions from instances and details? About these fundamentals also there are divergent assumptions and suggestions in the literature (cf. Aristotles, Kant, Husserl, Peirce, Ogden & Richards, Lyons, Morris). Commitment to certain assumptions and views has a substantial effect on what kinds of approaches and viewpoints are applicable in considering more IS-specific and ISD-specific issues.

In conclusion, method engineering may seem to be a well-structured effort with a confined scope. In practice, the situation is far from that. Method engineering involves, through the ISD method under construction, a huge collection of issues, ranging from the nature, structure and contents of an ISD method to the fundamentals of reality. With the short review of the literature presented above we have shown that there exists a large variety of assumptions and viewpoints about almost every single issue. To ensure a unified, coherent and consistent conceptual foundation for ME, it is necessary that the questions and issues raised above are carefully contemplated and answered, and "compatible" conceptual constructs are established and integrated.

That is what we are going to do in this work. The conceptual foundation is presented as an ontological framework, called OntoFrame. In Section 1.3 we defined the goals for OntoFrame, requiring that it should be comprehensive, contextual, consistent, coherent, and general. Further, we presume that OntoFrame is clear, natural, generative, extensible, modular and based on sound theories. Finally, we require that OntoFrame has to be applicable for three intentions. First, it should offer concepts and constructs for conceiving, understanding, structuring and representing phenomena in the concerned sub-domains (the descriptive intention). Second, it should serve as a conceptual foundation for the analysis and comparison of existing artifacts (the analytical intention). An artifact here means an ontology, a meta-model, a framework, a frame of reference, a technique, a method, etc. in the sub-domains. Third, OntoFrame should support the construction of other artifacts, in particular ISD methods and ME methods, by providing a conceptual foundation for the artifacts.

The literature reviews and comparative analyses made in this study show that regardless of numerous frameworks, frames of references, meta models, etc. suggested in the literature, there is none that would come even close to satisfying the stated goals (see the conclusions from the reviews and analyses in Sections 2.5, 3.10, 5.1.6, 6.4, 7.3, 8.5, 9.7, 9.8.6, and 12.2). They mostly cover only one or two sub-domains, and also in these sub-domains they address only part of the essential phenomena.

2.2 Theoretical Backgrounds

The purpose of this section is to shortly describe those disciplines, on which OntoFrame has been established, and derive fundamental views and principles of the framework from them. The disciplines are metamodeling and ontology engineering. For both of these disciplines we describe main concepts, approaches, principles and processes.

Metamodeling

Metamodeling is a discipline that studies models of models, called meta models, languages to represent meta models, and procedures to produce meta models. Generally speaking, a model is a thing that is used to help or enable the understanding, communication, analysis, design and/or implementation of some other thing to which the model refers. A representation of a model is called a model denotation (cf. Falkenberg *et al.* 1998, 55). A model denotation is presented in some modeling language. A meta model is a specification of the abstract syntax of a modeling language (cf. Oei 1995, 113). Also a meta model is represented in some language that is called a meta modeling language or a meta language. The abstract syntax of that language is specified by a model called a meta meta model.

Metamodeling is also seen as a process. It takes place on one level of abstraction and logic higher than the application modeling process (cf. Gigch 1991; Tolvanen 1998, 82). It comprises several sub-processes: e.g. abstracting from existing models, transforming from another meta model, translating from another meta model denotation, revising an existing meta model, and integrating other meta models. Metamodeling can be used to compare methods (e.g. Hong *et al.* 1993; Rossi 1996), to support standardization efforts (e.g. Booch *et al.* 1999), and to establish linkages between ISD methods (e.g. Ramackers 1994; Song 1997; Saeki 1998; Pohl *et al.* 1999).

Depending on what is the target of metamodeling, we can distinguish between different kinds of meta models: e.g. meta data models, meta process models, meta actor models, meta goal models, etc. Respectively, there are processes of meta data modeling, meta process modeling, meta actor modeling, and meta goal modeling.

The first attempts in metamodeling focused on data models (Teichroew *et al.* 1980). Meta models were built on some variants of semantic models (Hull *et al.* 1987), mostly on ER-based models (Sorenson *et al.* 1988; Welke 1988; Smolander 1991; Venable 1993) or NIAM–based models (Bommel *et al.* 1991; ter Hofstede *et al.* 1992; ter Hofstede *et al.* 1993a). Meta data modeling languages are based on set-theoretic constructs (Bergsten *et al.* 1989), predicate logic (Brinkkemper 1996; Harmsen 1997), attribute grammar (Katayama 1989), or object-oriented constructs (e.g. Object-Z, Saeki *et al.* 1994). Examples of specific meta data modeling languages are ASDM (Heym *et al.* 1992a), CoCoA (Venable 1993), GOPRR (Kelly *et al.* 1996), Telos (Jarke *et al.* 1990; Nissen *et al.* 1996), and MEL/MDM (Harmsen 1997). More about meta data modeling languages and differences between them can be found in Venable (1993), Saeki *et al.* (1994), Harmsen *et al.* (1996), and Tolvanen (1998, 155).

In the meta process modeling we can recognize several approaches to process modeling (Curtis *et al.* 1992; McChesney 1995; Heineman *et al.* 1994): e.g. process programming approach, functional approach, plan-based approach, Petri-net approach, and system dynamic approach. There are also different perspectives into processes (Curtis *et al.* 1992): functional, behavioral, organizational, and informational perspectives. Numerous languages for process modeling have been suggested (e.g. Bandinelli *et al.* 1993; Deiters *et al.* 1994; Christie 1993; Shepard *et al.* 1992; Dutton 1993; Kaiser *et al.* 1993). These are based on e.g. logic based rules, attribute grammars, state transition diagrams, Petri nets, etc. (Curtis *et al.* 1992; Armenise *et al.* 1993; Finkelstein *et al.* 1994; McChesney 1995). Some evaluations of and comparisons between process modeling languages are reported e.g. in Söderström *et al.* (2002).

Regardless of whether the purpose is to produce a meta data model, a meta process model, etc., three metamodeling approaches can be distinguished: top-down, bottom-up, and mixed approaches. In the top-down approach, concepts and constructs of a meta model are derived from relevant theories and/or generic views and assumptions on the relevant domains. The bottom-up approach focuses on modeling selected models generalizing their concepts and constructs. In the mixed approach, both generic views and existing models are sources for producing meta models.

To illustrate a process of metamodeling, we next apply the bottom-up approach and the ER model (Chen 1976). The process starts with the selection of models for metamodeling and with the decision on the aims of metamodeling. The aims can be related to evaluation, comparison, integration, mapping, etc. of models. They determine the level of detail, focus and emphasis of metamodeling. Second, main concepts in the models are identified and named. For example, in the DFD model (Yourdon 1989) the main concepts are 'process', 'store' and 'external'. They are regarded as entity types. Also main relationships between the concepts are recognized. In the DFD model, there is only one relationship called 'data flow' that relates externals, stores, and processes to one another. It is called the relationship type in the ER model. Third, main properties identifying or characterizing entity types or relationship types are recognized and named. These properties are attributes. For 'store' we can find two attributes, 'identifier' and 'name'. Further, roles that entities can play in relationships are identified. In the DFD model there are two roles, 'input' and 'output'. Fourth, constraints imposing on the concepts and the relationships are defined. In the DFD model there is a constraint that does away with a 'data flow' to directly relate 'external' and 'store' to one another. The steps described above are performed in an iterative way and in parallel with presenting the resulting meta model in a graphical notation. The process also comprises evaluation, refinements and restructuring of the meta model. If a target of metamodeling is large, like an ISD method, special steps are needed to integrate meta models (e.g. Song 1997; Harmsen 1997; Leppänen 2000).

Ontology Engineering

The notion of ontology appears in two meanings in the literature. It means, on one hand, a branch of philosophy dating back to 17th century[13]. In this sense, ontology is the study of existence, of all kinds of entities – abstract or concrete – that make up the world (Sowa 2000, 51). It concerns "what is out there" (Quine 1953) and "the basic traits of the world" (Bunge 1974, 38). Ontology engineering here means a discipline, which studies ontologies, ontology representation languages and procedures for engineering ontologies. On the other hand, the notion of ontology is used to refer to an ontology of some application or domain. In that sense, an ontology is regarded as "consensual knowledge represented in a generic and formal way to be reused and shared across applications and by groups of people" (Corcho *et al.* 2003, 44). Ontologies are kinds of frameworks unifying different viewpoints and serving as a basis for the communication between people, between people and systems, and between systems. Thus they function in a way like a lingua-franga (Chandrasekaran *et al.* 1999). Ontology engineering is a process, which aims to capture that consensual knowledge.

We define an *ontology* to mean an explicit specialization of a conceptualization of some part of reality that is of interest (cf. Gruber 1993, 199). A specialization can be presented in the form of a vocabulary, a taxonomy, a thesaurus, a conceptual framework, and a theory. A *vocabulary* or a glossary is a list of terms that have been enumerated explicitly. Each term has been defined or at least characterized properly. A *taxonomy* or a taxonomic skeleton is a classification of terms based on similarities in their meanings. In a taxonomy the

[13] Actually, one of the first philosophers having interests in ontological contemplations is said to be Heraclitus in the sixth century B.C.. He distinguished, among others, a set of top-level ontological categories (Sowa 2000, 56).

terms are related with subsumption or generalization relations[14]. A taxonomy mostly takes a form of a tree structure, but it can also have a lattice structure, or be in the form of multi inheritance graph. A *thesaurus* is a networked collection of vocabulary terms with associative relationships. A *conceptual framework* is a graph with concepts, relationships and rules for combining concepts (cf. Sugumaran *et al.* 2002, 253). A *theory*, commonly in an axiomatic form, defines concepts, relationships and rules formally with axioms (Guarino *et al.* 1995). Ontologies that are mainly in the form of taxonomies are called lightweight, and ontologies that model the domain in a deeper way and provide more restrictions on domain semantics are called heavyweight ontologies (Corcho *et al.* 2003, 44).

Another way of classifying ontologies is based on the level of generality (Guarino 1998, 7; van Heijst *et al.* 1997). *Top-level ontologies* describe very general concepts like space, time, matter, object, event and action, independently of a particular problem or domain. *Domain ontologies* and *task ontologies* describe, respectively, the vocabulary related to a generic domain (like medicine, or automobiles) or a generic task or activity (like diagnosing or selling). The concepts in those ontologies are specialized from the ones introduced in the top-level ontology. *Application ontologies* describe concepts depending both on a particular domain and task, and they are often specializations of both of the aforementioned ontologies. The boundaries between the kinds of ontologies are vague (van Heijst *et al.* 1997).

Further, we can distinguish between a meta ontology and an instance ontology. A *meta-ontology* provides concepts, relationships, and rules to specify instance ontologies (cf. Uschold *et al.* 1996, 15). In other words, a meta-ontology specifies semantics of an ontology language. In the simplest case, a meta-ontology is composed of concepts such as class, entity, and relation[15].

An ontology is represented in a language that can be informal or formal. A degree of formality varies from informal definitions of concepts expressed in a natural language to definitions stated in a formal language. Uschold *et al.* (1996) distinguish between highly informal, semi-informal, semi-formal, and rigorously formal ontologies. A highly informal ontology is loosely expressed in a natural language. A semi-informal ontology is expressed in a structured form of a natural language. A semi-formal ontology is expressed in an artificial, formally defined language. In a rigorously formal ontology the terms are meticulously defined with semantics and theorems.

Formality required from the ontology language is, to a large extent, dependent on the degree of automation in the tasks, which the ontology is to support. If an ontology is to be a framework for communication among people,

---

[14] In some cases, concepts within a taxonomy may also be related by the whole-part and type instance relationships.

[15] The term 'meta-ontology' is involved by synonym and homonym problems. van Heijst *et al.* (1997), for instance, use the term 'representation ontology' to mean meta-ontology (see also Davies *et al.* 2003), and Wand (1996, 281) calls any domain-independent ontology, like the ontology of his own, a meta ontology.

the representation of an ontology can be informal. If an ontology is to be used by software tools or intelligent agents, the semantics of an ontology must be made much more precise. Formal ontology languages include so-called traditional ontology languages, such as enriched first-order predicate logic (e.g. CycL, KIF, KL-ONE), frame-based languages (e.g. Ontolingua, F-logic, OCM), and description logic based languages (e.g. Loom) (Su *et al.* 2002, 4). In addition, there are web standards (e.g. XML, RDF) and web-based ontology languages (e.g. OIL, DAML+OIL, OWL, SHOE, XOL). Furthermore, graphical languages like CLEO (a graphical language for expressing ontologies (Falbo *et al.* 1998a)), LINGO (Falbo *et al.* 1998b) and UML (Cranefield *et al.* 1999; Kitchenham *et al.* 1999; Baclawski *et al.* 2001) have been used to present ontologies.

In the ontology engineering literature a variety of ontology engineering approaches are suggested (Noy *et al.* 2001, 6; Uschold *et al.* 1996, 20-22). The most common classification of the approaches is: top-down approach, bottom-up approach, and middle-out (or mixed) approach. A *top-down* engineering process starts with the definition of the most general concepts in the domain and continues with subsequent specializations of the concepts. A *bottom-up* engineering process starts with the definition of the most specific concepts, i.e. the leaves of the hierarchy, and continues with subsequent abstractions of these concepts into more general concepts. A *middle-out* or *mixed* engineering process is a combination of the two others. According to it, one defines the most important concepts first and then generalizes and specializes them appropriately.

Further, a large range of principles, guidelines, and even methods have been presented for ontology engineering (e.g. Uschold *et al.* 1995; Uschold 1996; Gruninger *et al.* 1995; Swartout *et al.* 1997; Fernandez-Lopez *et al.* 1999; Staab *et al.* 2001)[16]. However, none of the methods is fully mature if compared to methods in software engineering and knowledge engineering fields (Corcho *et al.* 2003). It is also common that each proposal applies its own approach. Nevertheless, we present next an outline of a process, which contains typical activities of ontology engineering (cf. Uschold *et al.* 1996).

The process starts with deciding why the ontology is wanted, what it will be used for and what kind of scope it has. Next, building the ontology is started. First, key concepts and relationships in the domain of interest are identified and defined ('capture phase'). Also terms with which to refer to such concepts and relationships are determined. Second, concepts and relationships as well as constraints related to them are presented in a chosen ontology representation language ('coding phase'). During the capture phase and/or the coding phase, possibilities to integrate parts of existing ontologies are examined, and if found beneficial, integration is carried out. The ontology engineered is evaluated based on defined criteria such as clarity, consistency and reusability. The ontology together with assumptions about the main

16 See also: Falbo *et al.* 1998a; Guarino *et al.* 2000; Noy *et al.* 2001; Schuster *et al.* 2001; Zhou *et al.* 2002; Kayed *et al.* 2002.

concepts is documented. Also the primitives used to express the definitions in the ontology (i.e. the meta-ontology) should be recorded.

Conclusions

Our objective is to build a conceptual framework by which knowledge about an ISD method and its engineering can be conceived, understood, structured and represented. This body of knowledge is related to five sub-domains: IS, ISD, ISD method, ME, and ME method. The knowledge should be able to be represented with generic concepts and easy-to-read notations in order to facilitate its general use.

Concluded from the descriptions of two disciplines above, we can state that both metamodeling and ontology engineering can provide theoretical "building blocks" for our framework. Metamodeling creates, extends, modifies, and integrates models of models that describe / prescribe particular sub-domain(s). Ontology engineering in turn collects, organizes and represents "consensual" knowledge of the concerned sub-domains. Ontologies can serve as unifying frameworks for different viewpoints. Both of these disciplines suggest languages to present artifacts, i.e. meta models and ontologies, respectively. The languages are specified with meta meta models in meta modeling and with meta ontologies in ontology engineering. These two disciplines allow a range of formality with which artifacts can be presented.

For the aforementioned reasons, we deploy concepts, approaches, principles and processes of metamodeling and ontology engineering in this work. To emphasize the "ontological roots" of our framework, we call it the ontological framework. The framework is a kind of "umbrella" comprising a generic ontology and a large set of domain-specific ontologies. The generic ontology is anchored in the philosophy of science. Most of the domain-specific ontologies are rooted on contextual concepts. The ontologies are presented in meta models composed of concepts, relationships and constraints. Some of the domain-specific ontologies (e.g. the model level ontology) are directly established on main constructs and principles of metamodeling. A process of engineering the domain-specific ontologies has been adapted from processes used in ontology engineering (see more closely in the next section). In the integration of the ontologies principles common in metamodeling are deployed.

2.3 Outline of OntoFrame

OntoFrame is an ontological framework, comprising a number of component ontologies with a multi-dimensional structure. These components range from highly generic ontologies to ME-specific ones. In this section we outline the framework by describing its sub-domains, structure and representation.

2.3.1 Sub-Domains

OntoFrame is a conceptual framework that should provide concepts and constructs for conceiving, understanding, structuring, and representing phenomena in ME as contexts and/or within contexts. In Section 2.1 we showed that the scope of the framework is very large, addressing several sub-domains. In this section we introduce the sub-domains by describing how reality is conceived through different views in this work.

We start with the most generic view according to which a human being lives in the 'physical' reality (the objective reality) and he/she has personal conceptions (the subjective reality) about it. He/She becomes aware of phenomena in reality through perceiving and interacting with it[17]. Interaction changes the subjective reality and often the objective reality as well. From this viewpoint, the reality "appears" to be single phenomena that are here called things (Bunge 1977). Taking a more specific viewpoint, things can be seen to be related to one another, thus constituting structures.

A still more specific view of reality is obtained when considering how human beings become to conceive reality and get their conceptions represented. This is facilitated by the use of the concepts of semiotics (Ogden *et al.* 1923), as well as of the notions of extension and intension. Furthermore, if there is a need to make sense of signs and their relations and meanings, basic concepts pertaining to the syntax and semantics of a language are taken into the use (Lyons 1977). Up till now, no view has been introduced to aid the recognition of changes in reality. For this purpose, the concepts of state and transition, known in systems theories (e.g. Klir 1969), can be used. Human beings have excellent abilities to build and maintain complex concept structures, infer from them and make abstractions from them. To understand abstractions and to support human beings in that requires a special view with abstraction concepts.

To conceive and make sense in more depth about how things are related to one another in reality and what are the meanings of those relationships, things must be viewed within meaningful contexts. Hence, it is necessary to have the notion of a context that clearly distinguishes the essential constituents of the context. At this stage reality manifests itself so complex and multifaceted that it is also necessary to have means to define and apply well-structured perspectives. It also becomes evident that phenomena are related to information processing at different layers and conceptualized by concepts at different meta levels. Hence, we need views that help us distinguish and structure processing layers and meta levels.

A still more specific view is needed if focusing on phenomena related to information systems and their development. These are special kinds of contexts, with particular kinds of purposes, actors, actions and outcomes. ISD methods, in turn, are prescriptions for ISD contexts. To conceive, understand, structure and represent features of ISD methods we need still another view, which offers

[17] Note that other human beings are part of the 'physical reality' and interacting with that also comprises the interplay with other human beings.

specific concepts and constructs related to e.g. the nature and structure of an ISD method. Also method engineering is a special kind of context. To make sense of phenomena related to that we need specific concepts and constructs.

In conclusion, conceiving reality in more depth necessitates the availability and use of more specific views. Views enable the recognition and representation of phenomena in a number of sub-domains. Implied from the above, we can distinguish between five sub-domains: IS, ISD, ISD method, ME and ME method. In addition, there is the generic sub-domain underlying and integrating the other sub-domains.

2.3.2 Overall Structure

OntoFrame is composed of four main parts that are: the core ontology, the contextual ontologies, the layer-based ontologies, and the method ontologies. Decomposition of the framework into the main parts has been made according to specificity of the ontologies (Guarino 1998, 7). Each main part is further divided into several sub-parts that are called *component ontologies*. The overall structure of the framework is presented in Figure 6. Arrows between the rectangles representing the main parts denote from which more specific concepts have been derived. In the following, we describe each of the main parts and their component ontologies in terms of their purpose, domain, and theoretical foundation (see Tables 1-4).

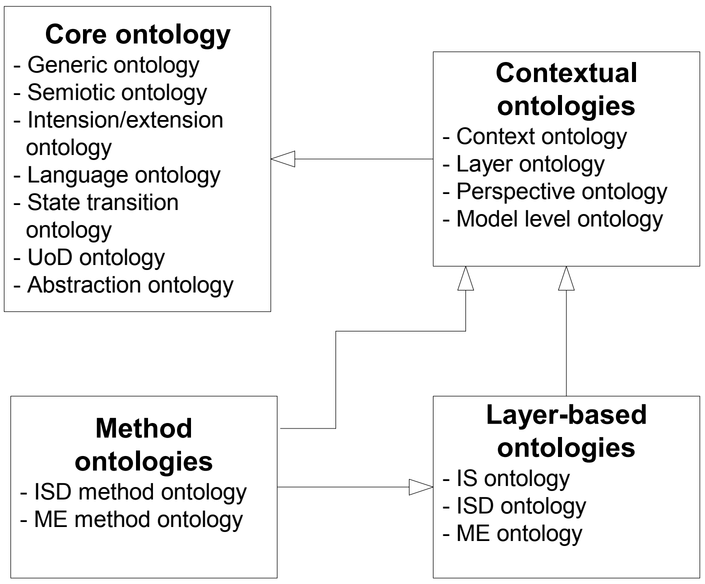


FIGURE 6 An overall structure of the ontological framework

The purpose of the *core ontology* is to provide the key concepts and constructs for conceiving, understanding, structuring and representing fundamentals of reality. It comprises seven component ontologies, each of which has its own purpose and role in the core ontology. The component ontologies are: the generic ontology, the semiotic ontology, the intension / extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology (Table 1).

TABLE 1 Core ontology

Ontology	Purpose	Domain	Theories
Generic ontology	To provide the most generic concepts from which all the other concepts can be derived	Reality	Philosophy of science
Semiotic ontology	To provide concepts for the recognition of semiotic phenomena	Linguistic, conceptual and physical reality	Semiotics
Intension/extension ontology	To provide concepts for categorizing concepts and defining their semantic meanings	Conceptual and physical reality	Philosophy of science
Language ontology	To provide concepts for defining the syntax and semantics of a language	Language	Linguistics
State transition ontology	To provide concepts for the recognition of dynamic phenomena	Static and dynamic phenomena	Systems theory
UoD ontology	To provide consolidated concepts for conceiving from a selected viewpoint	Subjective view	Systems theory, Philosophy of science
Abstraction ontology	To provide concepts for abstraction	Abstraction	Philosophy of science, Abstraction theory

The *generic ontology* provides the most generic concepts from which all other concepts can be derived by instantiation and/or specialization. This ontology corresponds to the top ontology in Guarino (1998). The most elementary concept is 'thing', which means any phenomenon in the 'objective' or subjective reality. The core ontology has roots in philosophy of science, especially in Bunge (1977). The *semiotic ontology* defines concepts that are needed to recognize semiotic phenomena. The main concepts, adopted from semiotics (Ogden *et al.* 1923), are 'concept', 'sign', and 'referent'. The *intension / extension ontology* serves a conceptual mechanism to specialize the notion of a concept and define its semantic meanings. The notions of intension and extension enable to differentiate between e.g. 'basic concept', 'derived concept', 'abstract concept', 'concrete concept', 'instance concept' and 'type concept'. Considerations of the intension / extension ontology are mainly based on the philosophical basis by Hautamäki (1986).

The *language ontology* provides concepts for defining the syntax and semantics of a language. Based on linguistics (e.g. Morris 1938), it contains concepts such as 'language', 'alphabet', 'symbol', and 'expression'. The *state transition ontology* is composed of concepts and constructs for the recognition of dynamic phenomena in reality in terms of states, state transitions, and events. The view of the ontology is based on systems theory (e.g. Klir 1969). The universe of discourse ontology, shortly the *UoD ontology,* is composed of consolidated concepts through which reality can be conceived as a totality determined by a selected viewpoint. These concepts are 'UoD state', 'UoD behavior' and 'UoD evolution'. On a general level, this ontology is based on systems theory and philosophy of science. The *abstraction ontology* serves concepts and constructs to abstraction by classification, generalization, aggregation, and grouping. Rooted on the intension / extension ontology, it also distinguishes between the first order abstraction and the second order abstraction (or predicate abstraction). It is based on the philosophy of science and abstraction theories.

The *contextual ontologies* help us recognize, understand and model phenomena in reality as contexts and within contexts. Among the contextual ontologies, there are four component ontologies. The ontologies are: the context ontology, the layer ontology, the perspective ontology, and the model level ontology (Table 2). The component ontologies are orthogonal to one another.

TABLE 2 Contextual ontologies

Ontology	**Purpose**	**Domain**	**Theories**
Context ontology	To provide concepts to conceive phenomena as contexts and within contexts	Social and human contexts	Pragmatics, Theories of human and social action
Layer ontology	To provide concepts to structure and relate information processing and its development	Information processing and its development	Systems theory, Information systems science
Perspective ontology	To provide concepts for distinguishing and applying perspectives	Information processing in the organizational context	Systems theory, Semantics, Abstraction theory
Model level ontology	To provide concepts for the creation, specification and presentation of models	Modeling, modeled, and model utilization contexts	Linguistics, Philosophy of science

The most essential ontology among the contextual ontologies is the *context ontology.* It recognizes seven contextual domains, called the purpose domain, the actor domain, the action domain, the object domain, the facility domain, the location domain, and the time domain. For each contextual domain, essential concepts and constructs are provided. The ontology is rooted on semantics, pragmatics and some theories of human and social action (Leont'ev 1978; Vygotsky 1978; Engeström 1987; Kuutti 1991). The *layer ontology* helps us structure and relate, on a general level, phenomena of information processing and its development at several layers. The layers are: information system (IS), information systems development (ISD), method engineering (ME), and research work (RW). This ontology is based on systems theory and information systems science. The *perspective ontology* provides a set of well-defined perspectives to focus and structure the perceptions of contextual phenomena. The perspectives are: systelogical, infological, conceptual, datalogical, and physical perspectives. The perspective ontology is based on systems theory, semantics, and abstraction theory. With the *model level ontology*, one is able to create, specify and present models, in different modes, about reality. The kernel in this ontology is a hierarchy composed of instance models, type models, meta models, meta meta models, etc. The ontology is based on linguistics and philosophy of science.

The third main part of OntoFrame is called the *layer-based ontologies.* While the layer ontology gives the basic structures for distinguishing between the processing layers and relating them to one another and the context ontology provides the generic concepts for recognizing contextual phenomena in any context, the layer-based ontologies elaborate the views on the IS, ISD and ME domains. These ontologies are: the IS ontology, the ISD ontology and the ME ontology (Table 3).

TABLE 3 Layer-based ontologies

Ontology	**Purpose**	**Domain**	**Theories**
IS ontology	To provide concepts to conceiving, structuring and representing contextual phenomena in the IS	IS context	IS theories
ISD ontology	To provide concepts to conceiving, structuring and representing contextual phenomena in ISD	ISD context	IS & ISD theories
ME ontology	To provide concepts to conceiving, structuring and representing contextual phenomena in ME	ME context	

The *IS ontology* helps us conceive, understand, structure, and represent phenomena in information-intensive contexts. Besides the information system, the ontology recognizes systems that are related to the IS (i.e. the object system, the utilizing system, and the controlled system). The *ISD ontology* provides concepts for the perception, understanding, structuring and representing of contextual phenomena in information processing development. The concepts are categorized along the perspective ontology. The ISD ontology has been built by selecting, abstracting, modifying and integrating concepts from multiple theories in the IS and ISD domains. Respectively, the *ME ontology* provides concepts to the perception, understanding, structuring and representing of contextual phenomena in method engineering. These concepts are also categorized along the perspectives. The ontology has been built on the basis provided by rather general and insufficient ME literature.

The fourth main part of the framework is called the method ontologies (Table 4). It provides concepts and constructs by which one can conceive, understand, structure, and represent the nature, structure and contents of a method. Instead of defining a generic method ontology, we define the ISD ontology and the ME ontology. This is done for two reasons. First, we are particularly interested in methods for development of information processing at various layers, and not in methods for product design, manufacturing, or the like. Second, ME is commonly seen to be analogous to ISD (Olle *et al.* 1988a; Kumar *et al.* 1992; Tolvanen 1998), and thus deriving the ME method ontology from the ISD method ontology is assumed to be a straightforward effort. In building the ISD and ME method ontologies, we have deployed the contextual ontologies and concepts from the ISD and ME domains, respectively.

TABLE 4 Method ontologies

Ontology	**Purpose**	**Domain**	**Theories**
ISD method ontology	To provide concepts for conceiving, under-standing and representing of the nature, structure and contents of an ISD method	ISD method	IS & ISD theories
ME method ontology	To provide concepts for conceiving, under-standing and representing of the nature, structure and contents of an ME method	ME method	

To summarize, the rationale to the decomposition of the ontological framework into the main parts and further into component ontologies is based on types of phenomena that are seen relevant to conceive and structure separately. According to the most generic view, reality is seen as being composed of things. This view is gradually specialized by applying several theories: philosophy of science, semiotics, semantics, pragmatics, and theories of human and social action. Finally, the views are "contextualized" into the domains of information systems, information system development and method engineering.

The modular structure of the framework benefits the building and use of OntoFrame in many ways. It helps us manage the complexity inherently resulting from a myriad of concepts and constructs. As seen from the above, there are multiple "dimensions" within the framework along which the concepts are situated. Without this modular structure it would be almost impossible to guarantee the coherence and consistence of the concepts and constructs. When applying the framework, the modular structure also guides the user to select the component ontology that is the most useful to his/her problem at hand.

2.3.3 Presentation

There are four degrees of formality by which the ontologies can be presented: highly informal, semi-informal, semi-formal and rigorously formal (Uschold *et al.* 1996). OntoFrame is mainly aimed at a means for the perception and communication between human beings. It is not aimed at the communication between people and computers, neither at the interaction between computers. On the other hand, the framework comprises a very large set of concepts and constructs, and most of them are highly abstract. For these reasons it is important to present the framework in a concise but understandable form. We have decided to deploy two forms of representation: semi-formal and highly informal. The semi-formal form is used to give an overview of the framework and facilitate communication between human beings with different backgrounds. This form is also used to specify constraints imposing the concepts and relationships. From a variety of semi-formal languages we have chosen a graphical language (see arguments by Guizzardi *et al.* 2001a, 2). Due to the inability of a graphical language to express details and deep meanings of things we also use a natural language to give a precise definition for each concept and construct. The definitions are embedded in the text and enclosed in Appendix 1.

There are many options for a graphical language. The first choice could have been made among special ontology representation languages, such as CLEO (a Graphical Language for Expressing Ontologies, Falbo *et al.* 1998a), LINGO (Falbo *et al.* 1998b), DAML+OIL (McGuinness *et al.* 2002) and OWL[18]. Most of these languages are designed to express deep generalization or subsuming hierarchies of concepts that are typical for ontologies built in artificial intelligence. The second choice could have been made among "traditional" graphical notations used in conceptual modeling, such as e.g. the ER notation (Chen 1976), the EER notation (e.g. Elmasri et. al. 2000), the NIAM notation (Nijssen *et al.* 1989), the GOPRR notation (Kelly *et al.* 1996), and the conceptual graph language (Sowa 2000). These notations have been designed to express a large range of relationship types, not only the generalization relationships. In addition, in these it is possible to include roles and multiplicity constraints in the graphical representations.

We, however, decided to select the UML language. UML (Unified Modeling Language) is a graphical language for visual object modeling. It has been standardized by OMG (Object Management Group) first in 1998, and the latest standardized version, UML 2.0, is from 2003 (OMG 2003). There are many reasons for this selection (cf. Kogut *et al.* 2002). UML has a very large and rapidly expanding user community, which guarantees that OntoFrame is easier to understand than if we represented it in some ontology representation language or in a traditional ER-like notation. UML has an intrinsic mechanism for defining extensions for specific domains, like for ontology modeling. The UML is supported by widely adopted CASE tools, which are more accessible

[18] http://www.w3.org/TR/2003/PR-owl-guide-20031215/ .

than current ontology tools such as Ontolingua[19] and Protégé[20], which require expertise in knowledge representation. Further, some research projects have also applied UML for ontology representation (e.g. Cranefield *et al.* 1999; Bergenti *et al.* 2000; Wang *et al.* 2001; Baclawski *et al.* 2002).

UML defines several types of diagrams that can be used to model static and dynamic features of a system (Booch *et al.* 1999). Because our framework is static, we apply the notation of the class diagram. The notation is very expressive. Although it is based on few key concepts (object class, association, role), it contains a large variety of specialized constructs, e.g. generalization, aggregation, and composition relationships. Constraints related to associations can be presented by multiplicities (1, 0..1, * , 1..*). If necessary, the special constraint expression language, called OCL (Object Constraint Language), can be used to formulate well-formed logic-based expressions

The class diagram contains many concepts that are not needed in the presentation of OntoFrame (e.g. class association, dependency association, operation/method etc.). Therefore, we separate a subset of the concepts of the class diagram, like OMG has done in defining the MOF (Meta Object Facility) model (OMG 2002). The MOF model is aimed at modeling the UML language itself, CWM (Common Warehouse Metamodel), and CCM (CORBA Component Model)[21]. Even from the set of the concepts and constructs of the MOF model we exclude some concepts, such as attribute, as unnecessary for our purposes. Our UML-based language for ontology representation is specified in Appendix 2.

An ontology representation language is a meta-ontology (Uschold *et al.* 1996, 15). By selecting the subset of UML as our graphical notation does not mean that we adopt the corresponding concepts as the concepts of our meta-ontology. As said above, our meta-ontology is the generic ontology, which is based on the notions such as 'thing' and 'relationship'. We only use UML as the notation and do not commit to its concepts.

In addition to constraints visible in the diagrams of the meta models, we define axioms to make some constraints within the component ontologies in the core ontology more explicit. Axioms are presented in the first-order predicate logic.

2.4 Approach and Process of Engineering OntoFrame

In this section we first describe what kinds of approaches and strategies to ontology engineering we have applied in building OntoFrame and why. Second, we describe the process by which we have engineered the component ontologies.

19 http://www.ksl.stanfrod.edu/sns.shtml
20 http://www.smi.stanford.edu/projects/protégé
21 www.dstc.edu.au/Research/Projects/MOF/Tutorial.html

Ontology engineering comprises categorizing, naming and relating things in an explicit way. There are two sources of ontological categories (Sowa 2000, 51): observation and reasoning. Observation provides knowledge of the physical world, and reasoning makes sense of observation by generating a framework of abstraction. This work is not based on observation of the physical world. Instead, we have extensively utilized and reasoned from the literature on the relevant sub-domains.

We distinguish between two approaches to the utilization of literature in ontology engineering. In the *inductive approach,* source material is collected from individual instance-level artifacts, i.e. ontologies, frameworks, and methods. A more generic framework is then abstracted from these artifacts. In the *deductive approach* some universal-like theoretic constructs are first selected from the literature and then used as underlying structures for a framework. We have applied both of these approaches. First, in building the core ontology we have made a thorough analysis of generic frameworks and ontologies (e.g. Bunge 1977; Wand 1988a; Wand *et al.* 1990a; Falkenberg *et al.* 1998) and derived by selection, integration, and customization our ontology from them. In contrast, in engineering the context ontology we have first searched for disciplines and theories (e.g. pragmatics (Levinson 1983) and theories of human and social action (e.g. Leont'ev 1978; Engeström 1987)) that address social contexts and derived from them the fundamental categorization of concepts into seven contextual domains. After that we have enriched the contents and structure of each domain deriving from existing artifacts. The elementary structures in the perspective ontology, the model level ontology and the layer ontology have also been derived from the relevant theories. For the rest of OntoFrame we have applied the deductive approach to derive lower-level ontologies from higher-level ontologies. In this process we have also heavily utilized the existing literature to complete and customize the derived concepts and constructs to fit in the concerned sub-domains.

We can find two pure representatives of the aforementioned approaches in the literature. Harmsen (1997) has built his MDM model (Methodology Data Model) by deriving from existing classifications and frameworks. The use of the inductive approach has resulted in a large set of IS-specific and ISD-specific concepts that are justified through their source artifacts. A drawback of this approach is that it does not encourage bringing forward new and innovative insights. In the BWW model[22] (Wand *et al.* 1990a; Wand *et al.* 1995b) for modeling information systems, fundamental concepts and constructs have been adapted from Bunge's ontology (Bunge 1977). By the use of the deductive approach the model pursues "universality" of concepts and constructs. The rationale behind the selection of Bunge's ontology has been (cf. Wand *et al.* 1995a, 287): An IS is a representation of another ("real-world") system. Because ontology is a branch of philosophy dealing with modeling reality, it is suitable to model information systems concepts. However, there is always a risk in selecting theories that have not originally been crafted for the field concerned.

[22] Here we consider the representation model only.

As Wand, Monarchi, Parson and Woo (1995) present, Bunge's "ontology is not generally accepted ontology, it seems to assume an 'objective reality', and it does not deal with the organisational and behavioural aspects of IS". Green and Rosemann (2000, 82) raise a question about whether the BWW model "is over-engineered", that is to say, whether it includes constructs that are not relevant. Further, Wand and Weber (1993) and Weber (1997) have recognized problems in understandability, comparability, and applicability of the BWW representation model constructs.

We have tried to overcome the aforementioned kinds of problems by applying both of the approaches. Theory-based constructs give underlying structures that are "tested", enhanced and elaborated by the inductive derivation from current artifacts. The use of theories advances not only the soundness of the framework but also innovations.

Another way to characterize the process of ontology engineering is to use the categorization of the approaches into top-down, bottom-up and mixed approaches (Uschold *et al.* 1996; Noy *et al.* 2001, 6). Our process has mainly followed the top-down approach. The process can be divided into four stages: (1) building the core ontology, (2) deriving the contextual ontologies, (3) deriving the layer-based ontologies, and (4) deriving the method ontologies (see Figure 6). The first versions of the abstraction ontology were made based on some preliminary assumptions and constructs within the generic, semiotic, and intension/extension ontologies (Leppänen 1984b). The process to define concepts and constructs was iterated and extended to cover the whole core ontology. Next, a comprehensive search for theories that address the notion of a context with the purpose of explicating the meaning of a thing was carried out. As a result the fundamental categorization of contextual domains was specified and later enhanced with contextual concepts and constructs integrating and adapting existing artifacts. Also some refinements in the core ontology were made at this stage. Third, the concepts and constructs of the ISD and ME domains were defined and structured to establish the ISD and ME ontologies, respectively. Finally, the method ontologies (the ISD method ontology and the ME method ontology) were defined based on the "lower" level ontologies.

The main strategies for ontology engineering are: (a) creation "from scratch", (b) adaptation of existing ontologies, and (c) integration of existing ontologies, or parts thereof. The ontology engineering literature most commonly applies the first strategy. Due to our source material and the multiplicity of sub-domains addressed in our work, we preferred to apply the integration strategy whenever it was possible. In this way we could import existing knowledge from sub-domains in which views and concepts are more stabilized and fit the overall premises. Adaptation was carried out when needed.

In line with aforementioned approaches and strategies we have specified a procedure of ontology engineering, which we have used in engineering each of the component ontologies in OntoFrame. The procedure is composed of the following steps:

- *Determine the purpose and domain of the ontology.*
  Decide for what purposes the ontology is to be used and what is the domain the ontology should address.
- *Consider reusing existing artifacts.*
  Review existing literature to find parts in ontologies for exploitation, preferably through integration and/or adaptation. Consideration of their usefulness is based on their fit in terms of purpose and domain.
- *Conceptualization.*
  Based on the review and analysis of the existing literature, identify key concepts and relationships of the ontology. Decide on terms for concepts and relationships and resolve possible synonym and homonym problems. Formulate definitions for the concepts. If some part can be extracted from existing ontologies, carry out actions to integrate that part to the body of the framework.
- *Formalization.*
  Present the concepts and relationships as well as constraints in the graphical form in UML. Sophisticated constraints are presented with axioms in the first-order predicate logic.
- *Evaluation.*
  Evaluate the ontology with a set of predefined criteria (i.e. clarity, consistency, coherence, extensibility).
- *Documentation.*
  Report on source materials used in the reviews as well as on decisions, with argumentations, made in the identification, selection and definition of concepts and constructs in the ontology.

For the whole framework, the description of the purposes and sub-domains given in Section 2.3.1 corresponds to the outcome of the first step. For each component ontology, the purposes and domains are shortly mentioned in Section 2.3.2. More detailed discussions of them are given for each component ontology in next chapters.

2.5 Comparative Review

The purpose of this section is to briefly describe relevant models, meta-models, and frameworks, called artifacts in short, and compare them with OntoFrame. First, we define the criteria for the selection of artifacts into our review and specify the issues to be considered. Then we report on the results of the review carried out at two levels of detail. The purpose of the review is to portray a general picture of the related work. This picture will be sharpened in next chapters where comparative analyses will be carried out in relation to specific component ontologies.

The literature suggests hundreds of frameworks, meta models, frames of reference and ontologies concerning IS, ISD, ISD methods, and/or ME. Merely for the assessment and comparison of ISD methods there are dozens of artifacts. Many of them are not ontological. They may just provide a list of features for characterizing ISD methods, or bring forward a set of taxonomies for the recognition and definition of approaches and viewpoints. To be ontological means that an artifact is composed of well-defined concepts and constructs addressing essential features of specific sub-domains.

We apply four criteria in the selection of artifacts for our review. The criteria are: purpose, coverage, familiarity, and specificity. *Purpose* means a reason for which an artifact has been developed. The purpose of an artifact must match with at least one of the intentions of OntoFrame (i.e. descriptive, analytical, and constructive intentions). *Coverage* means the sub-domain(s) that an artifact covers. The minimum requirement for coverage is that the concerned sub-domains of the artifact must belong to the set of sub-domains of our framework. In addition, we prefer artifacts with a large scope in this sense. *Familiarity* means that an artifact is well-known, i.e. published in a recognized journal or in the proceeding of a recognized conference. *Specificity* means that concepts and constructs within an artifact are defined in a specific way, and preferably supported by formal or semi-formal (graphical diagrams) representations. This criterion excludes those artifacts that only contain lists of features (e.g. Brandt 1983; Maddison *et al.* 1984). The use of the criteria led to the selection of 15 artifacts. Among them there are three models, nine frameworks and four meta models[23].

The comparative review is carried out in two parts. In the first part for each artifact, name, purpose, sub-domains and representation form are found out. *Purpose* of an artifact can be e.g. classification, categorization, analysis, comparison, evaluation, selection, integration, construction, etc. *Sub-domain* encompasses issues referred to by an artifact. It can be BS (business system), IS, ISD, ISD method, ME, and/or ME method. A business system means a system that utilizes the IS. *Representation form* means a way in which the concepts and constructs of an artifact are represented. Alternative forms are e.g. defined in a natural language, presented in a graphical notion, specified by axioms, etc. The summary of the first part of the review is presented in Table 5 (in the alphabetic order of the name of the first author or editor).

The fifteen artifacts can be classified into four groups according to the sub-domains they primarily address. The groups are: (1) comprehensive artifacts, (2) BS / IS domain-based artifacts, (3) ISD domain-based / ISD method-based artifacts, and (4) ME domain-based artifacts. In the following we shortly describe and analyze the artifacts within each group.

There are only two artifacts that can be considered comprehensive. They are the MDM (the Methodology Data Model) by Harmsen (1997) and the Frisco framework by Falkenberg *et al.* (1998). The MDM contains concepts referring to

[23] One artifact (Heym *et al.* 1992a; Heym *et al.* 1992b) is composed of a framework and a model.

TABLE 5 Names, purposes, sub-domains and representation forms of the reviewed artifacts

Nr.	Name/ Reference	Purpose	Sub-domain	Representation form
[1]	MADIS framework (Essink 1986, Essink 1988)	"aimed at providing capability for matching available methods and techniques to particular problem classes". .."aimed at providing means to integrate elements of different methods" (Essink 1988, 354)	IS, ISD	Definitions in English for some concepts. Concept classification in the matrix form.
[2]	Frisco-framework (Falkenberg *et al.* 1998)	"To provide an ordering and transformation framework allowing to relate many different IS modeling approaches to each other" (ibid p. 1)	IS	Definitions in English and in the first order predicate logic.
[3]	Organizational metamodel (Freeman *et al.* 1994)	"to represent all aspects of an information system, necessary for system understanding and software maintenance at four levels of abstraction" (ibid p. 283)	IS	Definitions in English. The meta-model in an extended ER diagram with some additional notational devices (no cardinalities).
[4]	Process meta-model (Grosz *et al.* 1997)	"an overview of the process theory for modeling and engineering the RE process" (ibid p. 115)	ISD, ME	Definitions in English. A process meta-model in an ER-like notation.
[5]	MDM (Methodology Data Model) (Harmsen 1997)	"To describe parts of ISD methods, thus supporting method engineering" (Harmsen *et al.* 1996, 218)	IS, ISD, ISD method ME	Definitions in English, supported with the use of first order predicate calculus, extended with functions and operators.
[6]	Framework and ASDM (a Semantic Data Model) (Heym *et al.* 1992a, 1992b)	a framework for describing ISD, and a semantic data model (ASDM) for describing ISD methods (Heym *et al.* 1992a, 215-16)	ISD, ISD method	Definitions in English. Semantic data model in a graphical notation (Lindtner 1992).

(continues)

TABLE 5 (continues)

Nr	**Name/ Reference**	**Purpose**	**Sub-domain**	**Representation form**
[7]	Conceptual framework (Iivari 1989a)	to facilitate "the systematic recognition, comparison and synthesis of different perspectives on the concept of an information system" (ibid p. 323)	BS, IS	Definitions in English, the framework in the OS (object system) graphs (Iivari *et al.* 1983).
[8]	Hierarchical spiral framework (Iivari 1990b)	"a hierarchical spiral framework for IS and SW development including evolution dynamics, main-phase dynamics, learning dynamics related to each other" (ibid p. 451)	ISD	Conceptual structures of ISD actions defined in English and partly described with diagrams and in a formal syntax.
[9]	Framework for understanding (Olle *et al.* 1988a)	"a framework for tacking systems planning, analysis and design, into which many existing methodologies can be fitted" (ibid p. vi)	BS, IS	Definitions in English. The framework in data structure diagrams.
[10]	Decision-oriented meta-model (Gupta *et al.* 2001)	a decision-oriented meta-model to be used for instantiating a method representation	ISD, ISD method	Definitions in English. Essential concepts and relationships in ER-like graphic diagrams; in addition a formal language (MRSL).
[11]	Meta-model (Saeki *et al.* 1993)	"a meta-model for representing software specification and design methods" (ibid. p. 149)	ISD	Definitions in English. The meta-model in an ER-like notation.
[12]	Framework (Song *et al.* 1992)	"A framework for aiding the understanding and handling the complexity of methods integration and thus making integration more systematic" (ibid. p. 116).	ISD, ISD method	Definitions in English, a schematic ER diagram for ISD method components.
[13]	ISA framework (Sowa *et al.* 1992)	to provide "a taxonomy for relating the concepts that describe the real world to the systems that describe an information system and its implementation" (ibid p. 590)	BS, IS	Concepts partly defined in English. The framework partly presented in an "ER style" graph.

(continues)

TABLE 5 (continues)

Nr	Name/ Reference	Purpose	Sub-domain	Representation form
[14]	Framework (van Swede *et al.* 1993)	"for classifying ISD modelling techniques, to assess the modelling capacity of development methods, and as a checklist for project leaders to construct project scenarios" (ibid p. 546)	IS	Definitions in English for most of the concepts, grouped by perspectives.
[15]	BWW model (Wand 1988a; Wand *et al.* 1989)	"is aimed at to be used as an ontology to define the concepts that should be represented by a modelling language, that is the semantics of the language" (Wand *et al.* 1995a, 287)	IS	Original definitions of concepts in English (Wand 1988a), later partly formalized with a mathematic notation (Wand *et al.* 1990a), and presented in an ER-like notation (Rosemann *et al.* 2002).

IS, ISD and ISD method and, to some extent, to ME. The Frisco framework focuses on the IS domain, but because it explicitly defines a very large set of concepts and constructs within it, thus covering several component ontologies within our framework, we regard it as a comprehensive artifact.

The BS / IS domain-based artifacts (Essink 1986; Essink 1988; Freeman *et al.* 1994; Iivari 1989a; Olle *et al.* 1988a; Sowa *et al.* 1992; van Swede *et al.* 1993; Wand *et al.* 1995a) contain concepts and constructs that facilitate the recognition, understanding and representation of structural and dynamic features of BS's and IS's. All but Wand *et al.* (1995a) also provide a set of perspectives to classify and structure the features according to pre-defined viewpoints. The artifacts in this group aim at e.g. "providing means to integrate elements of different methods" (Essink 1988, 354), "the systematic recognition, comparison and synthesis of different perspectives on the IS" (Iivari 1989a, 323), "relating the concepts that describe the real world to the systems that describe an IS and its implementation" (Sowa *et al.* 1992, 590), and "classifying ISD modeling techniques, assessing the modeling capacity of development methods and enabling a checklist for project leaders to construct project scenarios" (van Swede *et al.* 1993, 546).

The third group is composed of those artifacts, which provide concepts and constructs for the ISD domain (Essink 1988; Iivari 1990b; Grosz *et al.* 1997; Saeki *et al.* 1993) and / or for the ISD method (Heym *et al.* 1992a; Gupta *et al.* 2001; Song *et al.* 1992). For the ISD domain, Iivari (1990b) and Grosz *et al.* (1997) apply mainly a process view, whereas Heym *et al.* (1992a), Gupta *et al.* (2001), and Song *et al.* (1992) take a broader perspective on the ISD context. The purpose of the artifacts in this group is mostly to describe, assess, compare and integrate ISD methods or techniques.

In the literature, there are very few artifacts that describe structural and dynamic features of the ME domain. Among the artifacts reviewed here, such artifacts are those developed by Grosz *et al.* (1997) and Harmsen (1997). Grosz *et al.* (1997) apply a process meta-model for structuring ″meta-way-of-working″ in the meta-processes of the ME. Harmsen (1997) presents some conceptual structures that belong to the ME domain.

In all the artifacts the concepts are defined in English. In addition, some artifacts are represented in a graphical notion, based on an ER-like model (Freeman *et al.* 1994; Grosz *et al.* 1997; Gupta *et al.* 2001; Saeki *et al.* 1993; Song *et al.* 1992; Sowa *et al.* 1992) or on a more specific model (Heym *et al.* 1992a; Iivari *et al.* 1989; Iivari *et al.* 1990b; Olle *et al.* 1988a). Some artifacts also use more formal forms, such as first-order predicate calculus (Harmsen 1997) and a formal language (MRSL in Gupta *et al.* 2001). The BWW model, originally defined in English (Wand 1988a; Wand *et al.* 1989), was later formalized in a mathematical notation (Wand *et al.* 1990a) and presented in a graphical notation based on the extended ER model (Rosemann *et al.* 2002; Davies *et al.* 2003).

In the second part of the review we analyze the scope and emphases of the artifacts, using the overall structure of OntoFrame as the basis for the comparison. We give grades between 0 and 5 to show a degree to which the concepts and constructs in the artifact correspond, in terms of scope and quantity, to the concepts and constructs in our component ontology. The grades have the following meanings: 0 = not considered, 1 = considered slightly, 2 = considered fairly, 3 = considered equally, 4 = considered more, 5 = considered most. Note that the comparison is proportional to our framework, not to other parts of the artifact, neither to other artifacts. Still more detailed analyses will be presented in the next chapters, where we will look for concept-level correspondences. The overview of the results of the analysis is presented in Table 6. The numbers in the title row refer to the order numbers used in Table 5. Next, we discuss the results in the order of the main parts of OntoFrame.

Only five artifacts provide concepts and constructs belonging to the core ontology (comprises the first seven component ontologies in Table 6). The most comprehensive artifact in this respect is the Frisco Framework (Falkenberg *et al.* 1998), which addresses all seven component ontologies. However, concepts and constructs related to abstraction are only slightly covered. Next in comprehensiveness is the BWW model (Wand 1988a; Wand *et al.* 1989), which provides a rather large set of fundamental concepts and constructs (cf. the generic ontology) and a particularly deep consideration of phenomena related to states and state transitions. In contrast, it overlooks phenomena related to semiotics and language. The third artifact addressing the core ontology is Iivari's (1989a) framework. It is rather strong in defining concepts and

TABLE 6 Scopes and emphases of the reviewed artifacts compared to the component ontologies of OntoFrame

Ontology	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**[8]**	**[9]**	**[10]**	**[11]**	**[12]**	**[13]**	**[14]**	**[15]**
Generic ontology	0	3	0	0	1	0	0	0	0	1	0	0	0	0	3
Semiotic ontology	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
Intension/ extension ontology	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Language ontology	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
State transition ontology	0	3	0	0	3	0	2	0	0	0	0	0	0	0	5
UoD ontology	0	2	0	0	0	0	2	0	0	0	0	0	0	0	1
Abstraction ontology	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
Context ontology	2	2	1	0	2	0	3	0	2	0	1	1	3	1	0
Layer ontology	1	0	0	1	2	0	0	0	0	0	0	0	0	0	0
Perspective ontology	2	0	2	0	0	0	3	1	2	0	0	0	2	2	0
Model level ontology	1	2	0	1	2	1	0	0	0	0	0	0	0	0	0
ISD ontology	0	0	0	2	1	3	0	1	1	1	1	0	0	0	0
ISD method ontology	0	0	0	0	2	3	0	0	0	2	0	1	0	0	0
ME ontology	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
ME method ontology	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

constructs belonging to the state transition ontology and the UoD ontology but does not consider the other parts of the core ontology. In the MDM (Harmsen 1997) some fundamental concepts and abstraction concepts are defined. In addition, it contains a few concepts and constructs related to the state transition ontology. The decision-oriented meta-model, presented by Prakash (1999) and refined by Gupta *et al.* (2001), is derived from "the simplest meta-model" (Gupta *et al.* 2001, 140). The meta-model is composed of the concepts 'thing' and 'is_related', which have their counterparts in our generic ontology.

All the artifacts, except the BWW model (Wand 1988a; Wand *et al.* 1989), provide concepts related to our four contextual ontologies (i.e. the context ontology, the layer ontology, the perspective ontology, the model level ontology). The most comprehensive treatment is given in artifacts, which are focused on the BS / IS domain. The context ontology and the perspective ontology are most strongly addressed by the frameworks of Iivari (1989a) and Sowa *et al.* (1992). Other artifacts defining concepts and constructs of those component ontologies are the frameworks of Essink (1986, 1988) and Olle *et al.* (1988a). Falkenberg *et al.* (1998) and Harmsen (1997) define a large set of concepts and constructs related to the context ontology but they offer nothing to the perspective ontology. The two other contextual ontologies, the model level ontology and the layer ontology, are addressed by Essink (1986, 1988), Falkenberg *et al.* (1998); Grosz *et al.* (1997), Harmsen (1997) and Heym *et al.* (1992a), but on a rather general level.

Concepts and constructs related to the ISD ontology and/or to the ISD method ontology are defined on a detailed level in Heym *et al.* (1992a), while the artifacts of Harmsen (1997) and Gupta *et al.* (2001) remain in their definitions on a coarse level. The other artifacts addressing these ontologies, although only slightly, are Grosz *et al.* (1997), Iivari (1990b), Olle *et al.* (1988a), Saeki *et al.* (1993), and Song *et al.* (1992).

The only artifacts that reach the ME ontology are the process meta-model by Grosz *et al.* (1997) and the Methodology Data Model by Harmsen (1997). Their definitions for concepts and constructs cover, however, only a small part of the ME domain.

To summarize, among the reviewed artifacts there is none that would come even close to OntoFrame when regarding the coverage of the concerned sub-domains. In this phase we must, however, to point out that most of the artifacts have been published in articles for which space allowed in journals or proceedings is quite limited. Thus, it is not fare to expect these artifacts to be as coverable as those presented in dissertation theses or the like[24]. With these words in mind, we conclude that the most comprehensive artifact is the MDM by Harmsen (1997) but it also ignores many of the essential ontologies (e.g. the

[24] There are some researchers (e.g. Falkenberg, Iivari, Jarke) and research groups (e.g. Wand and Weber), which have contributed to several sub-domains in separate articles. Because the articles have been published at different times, they do not necessarily form unified and coherent wholes. It has not been possible for us to analyze those collections of articles in this work.

semiotic ontology, the intension/extension ontology, the language ontology, the perspective ontology) and addresses some others too superficially. The Frisco framework (Falkenberg *et al.* 1998) contains a large set of concepts but the concepts are exclusively related to the core ontology, the context ontology and the model level ontology.

We have above considered the comprehensiveness of artifacts in terms of their coverage of the IS, ISD, ISD method, and ME domains. We have good reasons for that. In the IS/ISD literature, coherence and uniformity among the basic concepts is commonly demanded. Our review showed in a concrete way that this demand is far from being fulfilled. Some artifacts concentrate on the fundamental concepts, while others define concepts and constructs related to e.g. the ISD domain or the ME domain. In the definitions on the "lower levels" the fundamental concepts are taken as granted thus jeopardizing the consistence and coherence of the defined concepts. Our approach aims to assure that all the ontologies, from top to bottom, share the same basic assumptions and views. Deriving specific concepts from fundamental concepts on the "bottom levels" helps us guarantee the consistence and coherency of the concepts and constructs within all fifteen component ontologies.

As said above, we will refer to the reviewed artifacts and compare them with OntoFrame in more detail in the next chapters. Besides those artifacts considered here, we will widely discuss a large number of other presentations that are not comprehensive enough to be catered in this section. To show the literature foundation on which we have built OntoFrame, we present references to the most relevant literature in Table 7. It is a small sub-set of those, which will be referred in the next chapters. The references are grouped according to the ontologies in OntoFrame and presented in the alphabetic order. We do not distinguish the IS ontology here, because it mostly shares the literature mentioned in relation to the context ontology. There are hardly any articles, which specifically address the ME method ontology. In Table 7 we mention some works, which we have used when deriving the ME method ontology from the ISD method ontology.

2.6 Summary

The purpose of this chapter was to give an overview of OntoFrame. First, we motivated OntoFrame with needs for a unified, coherent, and consistent conceptual foundation. Second, we anchored OntoFrame on the theoretical foundations of metamodeling and ontological engineering. Both disciplines are interested in shared knowledge and its representation in meta models or ontologies, respectively. Third, we outlined OntoFrame by describing various views from which reality can be conceived for specific purposes and distinguished a multi-layered structure of sub-domains that OntoFrame should address. Fourth, we described the overall structure of OntoFrame. It is

TABLE 7 Other relevant literature

Ontology	References
Core ontology	Berztiss 1999, Dominques *et al.* 1997, Goldstein *et al.* 1999, Hautamäki 1986, Henderson-Sellers *et al.* 1999a, Kangassalo 1982, Krogstie 1995, Mattos 1988, Motschnig-Pitrik *et al.* 1999, Motschnig-Pitrik *et al.* 1995, Mylopoulos *et al.* 1990, Opdahl *et al.* 1994, Schrefl *et al.* 1984, Sowa 2000, Wand *et al.* 1999.
Context ontology	Barros 1991, Baskerville 1996, Bittner *et al.* 2002, Engeström 1999, Fillmore 1968, Herbst 1995, Koubarakis *et al.* 2000, Kavakli *et al.* 1999, Kerola 1980, Levinson 1983, Liu et al, 2002, Loucopoulos *et al.* 1998, McDermott 1982, Mesarovic *et al.* 1970, Ramackers 1994, Randell *et al.* 1989, Sowa 2000, Searle 1979, Stamper, 1975, Yu *et al.* 1997, Zhou *et al.* 2000.
Layer ontology	Bertalanffy 1974, Falkenberg *et al.* 1992a, Gasser 1986, Nonaka 1994, Stamper 1996, Verrijn-Stuart 1989, Welke *et al.* 1982.
Perspective ontology	Avison *et al.* 1996, van Griethuysen 1982, ISO 1996, Langefors *et al.*1975, Olive 1983, Peirce 1955, Sol 1992, Welke 1977.
Model level ontology	Bergheim *et al.* 1989, Brinkkemper 1990, Gigch 1991, ter Hofstede *et al.* 1997, ISO 1990, Jarke 1992, OMG 2002, Wijers 1991.
ISD ontology	Baskerville 1989, Boehm 1988, Bracchi *et al.* 1984, Checkland 1988, Cysneiros *et al.* 2001, Glasson 1989, Goldkuhl *et al.* 1993, Hirschheim *et al.* 1989, Hirschheim *et al.* 1992, Iivari 1991, Iivari *et al.* 2001, Jarke *et al.* 1992, Kruchten 2000, Lyytinen 1986, Nature Team 1996, Saeki 1998, Simon 1960, Thayer 1987, Vlasblom *et al.* 1995, Wood-Harper *et al.* 1982.
ISD method ontology	Avison *et al.* 1996, Fitzgerald *et al.* 2002, Hidding *et al.* 1993, Hirschheim *et al.* 1995, Iivari 1983, Iivari *et al.* 2001, Karam *et al.* 1993, Lyytinen 1986, Mathiassen *et al.* 1986, Schön 1983, Stamper 1973, Tolvanen 1998, Vlasblom *et al.* 1995, Zhang *et al.* 2001.
ME ontology	Backlund *et al.* 2003, Brinkkemper *et al.* 1999, Henderson-Sellers *et al.* 1999c, Hidding *et al.* 1993, Kruchten 2000, Kumar *et al.* 1992, Vlasblom *et al.* 1995, Ralyte 2002, Ralyte *et al.* 2001, Ralyte *et al.* 2003, Rolland *et al.* 1996, Saeki 1998, van Slooten *et al.* 1993, Song 1997, Tolvanen 1998, Zhang *et al.* 2001, Veryard 1987.
ME method ontology	Harmsen 1997, Iivari *et al.* 2001, Kinnunen *et al.* 1996, Rossi *et al.* 2005, Saeki 2003, van Slooten *et al.* 1993, Stamper 1973.

composed of four main parts, called the core ontology, the contextual ontologies, the layer-based ontologies, and the method ontologies. Each main part was further divided into component ontologies. For each component ontology, the purpose, domain and theoretical foundations were described. Theories in this work include e.g. philosophy of science, semiotics, semantics, pragmatics, theories of human and social action, systems theory, and information systems science.

Fifth, we discussed alternative forms of presenting OntoFrame and decided to present definitions in English and meta models in a sub-set of UML. Sixth, we described various approaches and strategies to engineering OntoFrame and selected to use the mixed approach and the integration strategy. We also presented a procedure for engineering component ontologies.

Seventh, we presented a comprehensive review of the artifacts in the literature, to give an overview of the related work and to compare the scope and emphases of the artifacts with OntoFrame. From the analysis we concluded that there is no artifact that would come even close to OntoFrame when regarding the coverage of the concerned sub-domains. Even the most comprehensive artifact, the one by Harmsen (1997), ignores many essential ontologies and addresses many other sub-domains too superficially. Some artifacts concentrate merely on the fundamental concepts, while the others define concepts and constructs for "lower-level" domains, such as ISD and ME. In the latter case, the fundamental concepts are taken as granted, thus jeopardizing the consistence and coherence of the totality. Our approach aims to assure that all the ontologies, from top to bottom, share the same basic assumptions and views.

3 CORE ONTOLOGY

The purpose of this chapter is to present the core ontology that constitutes the topmost level in OntoFrame. The core ontology provides key concepts and constructs to conceiving, understanding, structuring and representing the fundamentals of reality. It is composed of seven component ontologies: the generic ontology, the semiotic ontology, the intension/extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology.

The chapter is organized as follows. First, we make a short survey of the related work. Second, we describe the overall structure of the core ontology. In the next seven sections we present the component ontologies. For each component ontology, the concepts and constructs are defined and described in meta models. Also, relevant literature is extensively referred to and compared. After presenting the component ontologies, we provide a categorization of the relevant literature and make a comparative analysis of two of the most prominent presentations. The purpose of the analysis is to reveal the objectives, ontological positions, basic structures, coverage and emphases of the presentations. The chapter concludes with a summary.

3.1 Related Work

What are things? What is the essence that remains inside things even when they change in color, size, etc? How are things related? Do concepts exist outside our mind? These are questions that Ontology has tried to answer for thousands of years. Work of philosophers and scientists has resulted in a large variety of categorizations and generic ontologies, providing primitive concepts for making sense of the essence of things and of their existence. In this thesis it is not possible, or even reasonable, to discuss widely various schools of thought and approaches of engineering generic ontologies. Instead, to relate the core ontology to the prior work, we present a short overview of philosophical positions, well-known categories and generic ontologies. A more detailed analysis of two presentations is presented in Section 3.10.

In defining the fundamental concepts for perceiving and conceiving the essentials of reality one has to commit oneself to some philosophical or metaphysical positions and assumptions ("Weltanschauung"). In the past few decades, there has been a vivid discussion going on, also in the information system field, about these paradigmatic assumptions (e.g. Klein *et al.* 1987; Floyd 1987; Nurminen 1988; Hirschheim *et al.* 1989; Iivari 1991; Stamper 1999; Orlikowski *et al.* 1991; Hirschheim *et al.* 1992a; Dahlbom *et al.* 1993; Iivari *et al.* 1998a; Chen *et al.* 2004). This discussion has yielded various classifications of the assumptions about e.g. the nature of reality (ontology), and what is human knowledge and how it can be acquired (epistemology). Here we consider the ontological assumptions as presented in Falkenberg *et al.* (1998)[25]:

- *Objectivism.* There is one reality, independent of any observer and interpreter. That is why reality and perceived reality are the same.
- *Constructivism.* There is one reality, independently of any observer. Each human being perceives and conceives reality differently. That is why reality and perceived reality are not exactly the same.
- *Mentalism.* There is more than one reality, because reality is perceived and conceived solely by the senses of human beings, and hence reality is completely dependent on the observer.

One of the first philosophers, Aristotle, distinguished between different modes of being and defined a system of categories. Aristotle's categories include substance, quality, quantity, relation, activity, having, situatedness, spatiality, and temporality[26]. Later, Immanuel Kant presented his categorization (Kant 1787), which is organized into four classes each of which presents a triadic pattern: quantity (unity, plurality, totality), quality (reality, negation, limitation), relation (inherence, causality, community) and modality (possible, existence, necessity). Since then, there have been several categorizations made in the philosophy of science (e.g. Husserl, Whitehead, Pierce, and Heidegger).

In recent decades, research into fundamental categorizations, or ontologies, has extended into the fields of Artificial Intelligence (AI) and Information Systems (IS). In AI, some of the best known top-level ontologies, or upper ontologies, are Sowa (2000), Cyc and SUMO. Sowa's top-level ontology (Sowa 2000) includes the basic categories and distinctions derived from a variety of sources in logic, linguistics, philosophy, and artificial intelligence. The ontology, containing 27 concepts, has a lattice structure where the top concept is the universal type. The universal type contains all the possible

[25] The trichotomy is based on three semantic principles presented by Stamper (1992b, 28). There are several other options for the classification, especially in the philosophy of science (e.g. Niiniluoto 1999). Here, we are satisfied with the one in Falkenberg *et al.* (1998) because it is simple enough for our purposes.

[26] This system of categories was presented in the Categories, the first treatise in Aristotle's collected works (cf. Sowa 2000, 56).

instances of the ontology. The direct subclasses of the universal type are: independent, relative, mediating, continuant, physical, abstract, and occurrent.

Cyc's upper ontology aims to provide the most general concepts of human consensus reality (Lenat *et al*. 1990). The Cyc Knowledge Base contains thousands of terms and millions of assertions. In the following we only mention the most generic terms in the ontology (see more in http://www.cyc.com/cycdoc/vocab/vocab-toc.html). The class 'thing' is the root of the ontology. A thing can be an individual, a partiallyIntangible or a mathematicalOrComputationalThing. An individual is a temporalThing, a spatialThing, or a partiallyIntangibleIndividual. A temporal thing can be somethingExisting or a timeInterval. The upper ontology also contains terms such as event, situation, relation, attributeValue, predicate, role, and collection.

SUMO[27] (Suggested Upper Merged Ontology) is promoted by the IEEE Standard Upper Ontology Working Group and was officially approved as an IEEE standard project in December 2000. The root of the ontology is entity that can be either physical or abstract. A physical entity is either an object or a process. An abstract entity can be a quality, an attribute, a class, a relation, a proposition, a graph, or a graphElement.

Also in the IS field some suggestions for top-level ontologies have been made (e.g. Wand *et al*. 1990a; Falkenberg *et al*. 1998). The BWW model (e.g. Wand 1988a; Wand *et al*. 1989; Wand *et al*. 1990a; Wand *et al*. 1990b) is based on Bunge's ontology (Bunge 1977) according to which the world is made of things that possess properties. Things are concrete or conceptual. Properties are intrinsic or mutual. Properties of conceptual things are termed attributes. Attributes are characteristics assigned to things according to human perceptions. The Frisco[28] framework (Falkenberg *et al*. 1998) aims to provide an ordering and transformation framework to allow relating different IS modelling approaches to each other. The concepts in the framework have been derived from one single concept, thing, by specialization (Falkenberg *et al*. 1998, 34). The other primitive concepts include predicator, predicated thing, composite thing, elementary thing, relationship, state, transition, entity, type, etc.

There are several other, not so well-known, presentations which aim to establish a common foundation for modelling the real world. For instance, Opdahl *et al*. (1994), based on some modifications of the DFD language, propose primitive concepts such as item, attribute, value, domain, substance, and data. Krogstie (1995, 8) defines elementary concepts for modelling computerized information systems. He starts with defining the notion of a phenomenon to mean "something as it appears in the mind of a person" (ibid p. 8). Related to a phenomenon he defines the notions of property, state, transition and event.

Ontologies differ from one another in terms of their purpose, extensiveness and contents. They also differ in how they view changes in reality. Static ontologies primarily describe what things exist, their attributes and relationships (Mylopoulos 1998, 136). Dynamic ontologies view dynamic

[27] http://suo.ieee.org/

[28] Frisco = FRamework of Information Systems Concepts.

aspects in terms of states, state transitions and processes. There are also ontologies in which things and events are equally treated (e.g. Feibleman 1951; Brody 1980; Tiles 1981). Mylopoulos (1998) also distinguishes the intentional ontologies and the social ontologies. The intentional ontologies address the world composed of agents and things agents believe in, want, prove or disprove, and argue about. The social ontologies cover settings, permanent structures or shifting networks of alliances and interdependencies. They are characterized in terms of actors, positions, roles, authorities, commitments, etc. These two kinds of ontologies provide views that go beyond the scope of our core ontology.

In defining the core ontology we have utilized those existing ontologies that have been engineered in particular for the IS field. To ensure the uniformity and coherence of the core ontology, we have been obliged to fill some 'gaps' not addressed by any existing ontologies, and make some adaptations to get the "pieces" fit together.

3.2 Overall Structure

The purpose of the *core ontology* is to provide key concepts and constructs for conceiving, understanding, structuring and representing fundamentals in reality. It is composed of seven component ontologies: the generic ontology, the semiotic ontology, the intension/extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology (see Figure 7). In the following, each component ontology and its theoretical basis are discussed.

The generic ontology aims to provide the most generic concepts from which the concepts of all other ontologies in OntoFrame can be derived. The most generic concept is thing. Derivation of the concepts of the other ontologies from the generic concepts is carried out by instantiation and specialization. The core ontology has its roots in the philosophy of science.

The semiotic ontology is based on the basic concepts in the theory of signs – semiotics (Ogden *et al*. 1923; Peirce 1955; Morris 1946). The semiotic ontology specializes the notion of a thing into three semiotic notions: sign, concept and referent. Between the notions there are three well-defined relationships: signifies, refersTo, and standsFor. In the intension/extension ontology the notion of a concept, in turn, is elaborated into more specialized concepts, such as basic concept, derived concept, individual concept, and generic concept.

In the language ontology the focus is on the notion of a sign represented in a language. Based on the linguistics, a language is defined as a composition of a vocabulary, syntax and semantics, and sub-concepts of a sign are specialized. To enable distinguishing between static and dynamic features in reality, the state transition ontology with the notions of state, transition and event are defined. The concepts in this ontology have been established on general system

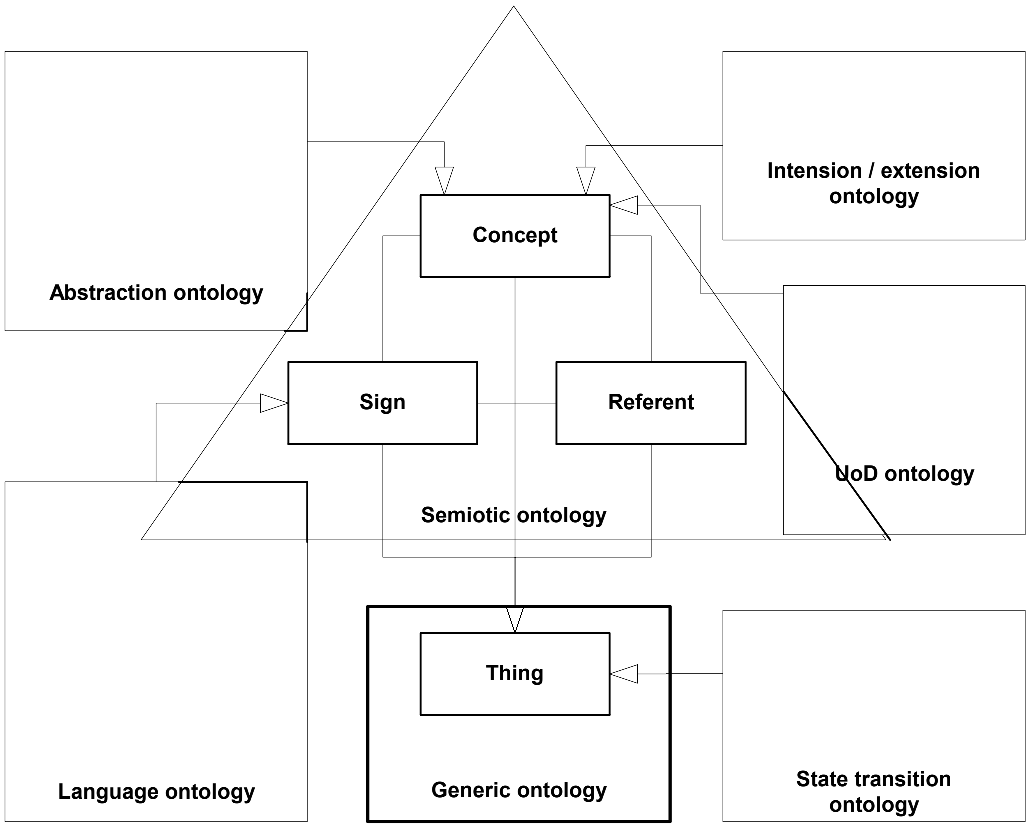


FIGURE 7 An overall structure of the core ontology

theory (von Bertalanffy 1968; Klir 1969). The UoD ontology provides concepts related to the universe of discourse (UoD): UoD state, UoD behavior and UoD evolution. Through these concepts, personal or inter-personal points of view on reality can be distinguished, discussed and compared. The UoD ontology is rooted on theories of conceptual modelling (e.g. van Griethuysen 1982; Brodie *et al*. 1984).

The abstraction ontology specializes the notion of a concept, providing sub-concepts for four kinds of abstraction: classification, generalization, composition, and grouping. Abstraction, rooted in Aristotelian philosophy, means ignoring irrelevant things to uncover the features relevant to the problem at hand. Depending on whether the things are entities with at least some independence or predicates merely used to characterize the entities, we can distinguish between the first-order abstraction and the predicate abstraction (or the second-order abstraction).

In Figure 7 the seven component ontologies of the core ontology are depicted with six rectangles and one triangle. For the generic ontology and the semiotic ontology the most essential concepts are also presented. Later in Section 3.10 also the other "boxes" are filled by the concepts defined in the next sections. The relationships between the ontologies are presented with arrows standing for specialization. Because the figure is highly simplified, all relationships are not made visible.

3.3 Generic Ontology

The *generic ontology* provides the most generic concepts from which the concepts of all the other component ontologies in Ontoframe can be derived.

Reality is anything that exists, has existed or will (possible) exist. We distinguish between the subjective reality and the objective reality (Bunge 1977; Lyons 1977)[29]. The *subjective reality* (or the perceived reality) is the result from our mental processes. The *physical reality* (or the reality in short) is independent of any human thinking. It is the source of sense data, which we obtain, and it is thus external to us. We agree that it is not possible to say anything sure about the physical reality, because our conceptions of it are ultimately dependent on our senses, skills of understanding and personal points of view. However, there is no doubt that some kind of physical reality really exists, independently of us, and that reality manifests itself through a huge variety of phenomena. Conceptions an individual has about the physical reality may be quite different from the ones other individuals have. That does not prevent us from trying to distinguish and name the phenomena of reality. Consequently, we accept the constructivist position (cf. Stamper 1992b; Falkenberg *et al.* 1998).

We define a *thing*[30] to mean any phenomenon in the physical or subjective reality. That is all we can say about the physical reality. Saying more would require the use of more specialized concepts and structures, which necessarily means using perceptions and conceptions of a human being[31], that is, becoming part of the subjective reality. A thing may be a ball, the weight of the ball, the intention of a player running to reach the ball, or the number on the back of the player.

In the subjective reality things are characterized with one or more properties. A *property* is a thing that is used to characterize other thing(s). A

29 Habermas (1984, 100) divides reality into three worlds: objective world (the totality of all entities about which true statements are possible), subjective world (the totality of the experiences of the speaker to which he/she has privileged access), and social world (the totality of all legitimately regulated interpersonal relations). According to Wand *et al.* (1995a, 290) the physical reality can be replaced by an "inter-subjective" reality when necessary.

30 We have selected the term 'thing' to stand for the most elementary concept for two reasons. First, many well-known ontologies in the field (e.g. Wand *et al.* 1990a; Lenat *et al.* 1990; Miller 1990; Falkenberg *et al.* 1998) use the same term, yet with somewhat different meanings, to denote the elementary concept. Second, another alternative, 'object', is overloaded with other kinds of use, also in this study. Besides 'thing' and 'object', other terms are also suggested in the literature (e.g. 'phenomenon' in Krogstie (1995, 8)), but we consider them instances of specific approaches with no large support.

31 Actually, seeing the physical reality consisting of things is also an assumption, which contains mental interpretation. But we want to have some concept with which we can refer to phenomena in reality. Falkenberg *et al.* (1998, 29) talk about "parts" or "aspects" of the "world".

characterized thing is a thing that is characterized by at least one property. A ball is a thing that is characterized by the property weight. Things may be related to other things in many ways. A *relationship* is a thing that relates two or more characterized things together, each one associated with one property characterizing the role of that thing within that relationship. A *role* is a property that reflects a position the thing holds, or a function the thing conducts, in the relationship[32]. Serving is the relationship that relates a player and a ball, and ownership is the relationship, which shows that a shirt belongs to a player. Because the relationship is a thing, relationships between relationships can also be recognized. A characterized thing can be related through one or more relationships, and a relationship can relate two or more things (see Figure 8).

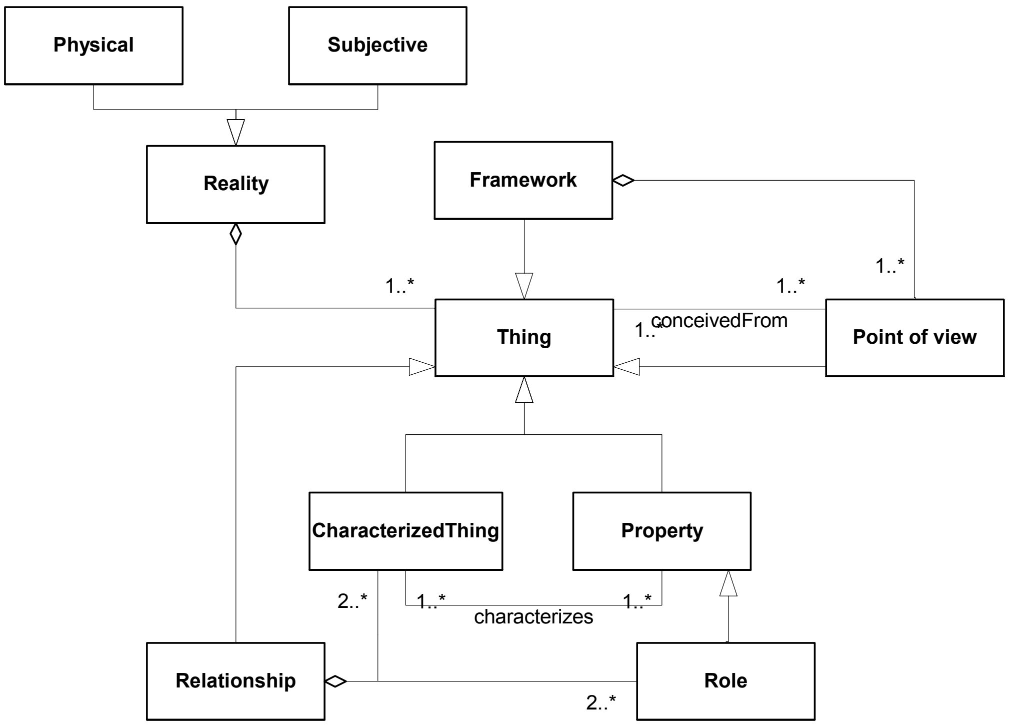


FIGURE 8 Generic ontology

A thing can be perceived and conceived in various ways. Let the thing be the player with the number 9 on his back. He is known as 'John Smith', having a six years career in professional football teams in the UK. He is also known as a partner of the company investing on young talented players. Third, the thing is conceived as a close relative, who should take care of his injured knee. Consequently, for the same thing in the physical reality there are at least three different conceptions: one possessed by the TV-commentator, the other by a

[32] There is a large number of literature considering the nature, characteristics and evolution of role (e.g. Kaasboll 1995; Lindgreen 1995; Gottlob *et al.* 1996; Halpin 1998; Steinmann 2000; Dahchour *et al.* 2002; Coulondre *et al.* 2002). It is not possible here to discuss them further.

young player dreaming of a professional career, and the third one by his wife looking at TV. The human mind produces a variety of subjective conceptions from the same thing in the physical reality, depending on the point of view adopted.

The notion of a point of view is vague in everyday life as well as in scientific treatises. Here we see it as a way to view or consider something (cf. Webster 1989). To put it more precisely, we can say that every thing has many properties, and to adopt a point of view is to consider some of these properties relevant. Using a *point of view,* some things and some properties of the thing(s) are selected because they are more relevant than others. When a statement is made from that point of view, then the reasons for the statement are just selected properties (cf. Hautamäki 1986, 65). A point of view itself is, of course, a thing.

Applying a point of view leads to a more or less limited or "predefined" conception of certain things and their properties in reality. To derive and relate the views, a framework is commonly deployed. A *framework* is a thing that guides a human being to select the points of view that are the most appropriate for the case or the problem at hand. A framework can be intuitive or formally established, vague or rigid. The framework relating the viewpoints of the human beings interested in the thing on the football field is an example of the intuitive and vague framework. It could be called the "sociometrical" framework by which one explores what members of a group perceive, think and feel about the other members of the group. The categorization of the reality into two parts, subjective and physical, is based on the philosophical framework. It is more rigid because it is grounded on the ontological and epistemological theories, which state the possible points of view, their conceptual contents and relations. Another example of the rigid framework is the semiotic framework, which we shall apply in the next section.

3.4 Semiotic Ontology

The *semiotic ontology* provides concepts and constructs to recognize semiotic things in reality. It specializes things according to the semiotic framework based on the theory of signs - semiotics[33]. In semiotics, three realms are distinguished: the realm of signs, the realm of concepts, and the realm of referents. By applying Ogden's and Richards' meaning triangle (Ogden at al. 1923), the semiotic framework can be illustrated such as in Figure 9[34].

---

33 The semiotics, as sketched by Peirce (1955) and de Saussure (1931) and elaborated by Morris (1946), concerns signs and their relations to the other things that are essential to the creation, use and understanding of the signs. The term "semiotics" originates from Greek in which "semeion" means a sign or a mark.

34 We use Ogden's and Richards' meaning triangle instead of other alternatives (e.g. Morris 1946; Peirce 1955) because of its simplicity and familiarity.

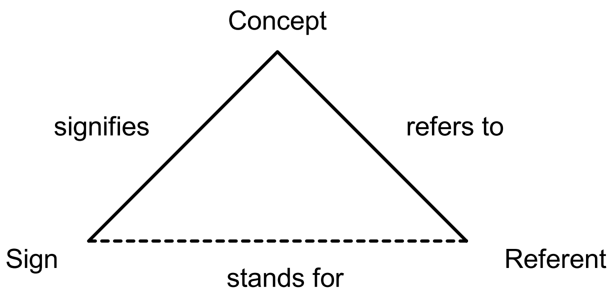


FIGURE 9 The semiotic framework as the meaning triangle

Using the semiotic framework, we can distinguish between three kinds of things: concept things, sign things and referent things. *Concepts* are mental things, words of mind (cf. Hautamäki 1986). In philosophy and psychology they are regarded as ideas, thoughts or mental constructs by means of which the mind apprehends or comes to know things. They are basic epistemological components of human knowledge. We call wholes, which are composed of related concepts, *constructs*. A *referent* is a thing in reality to which a concept refers. It can be a physical thing, a process, an event, Wonderland that Alice visited, or the like. A *sign* or a symbol is any thing, which can stand for something else. It is a representation of a concept expressed in a symbolic or iconic language. Our world is full of things that are used as signs: words, pictures, facial expressions, body postures, films, traffic lights, etc. Here we mainly consider verbal representations.[35].

In the meaning triangle the relationships between the concepts, the referents and the signs are denoted by the edges. Based on the triangle we present the semiotic ontology in the meta model in Figure 10. A sign *signifies* or designates a concept. A concept *refers to* a referent. A sign *stands for* a referent, but it is not directly associated with a referent because a sign may have several meanings leading to different referents[36].

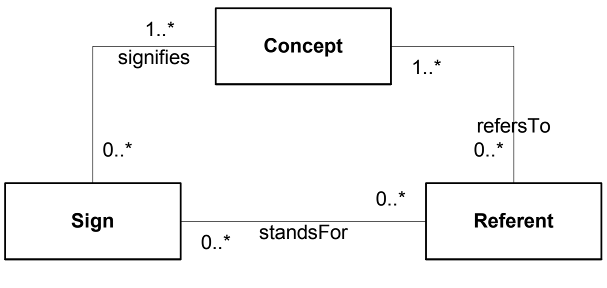


FIGURE 10 Semiotic ontology

35 We want to be faithful to the original term 'sign' although its denotation may give rise to a conception of a more elementary linguistic thing.

36 It is only the case of an idealized observation in which it is assumed that each referent in reality leads to at most one sensation or concept construct, and each sensation has at most one sign.

We can distinguish between three elementary human processes through which the semiotic things are produced (cf. Falkenberg *et al*. 1998, 46-47). *Perceiving* means a process whereby a human being observes reality with his/her senses and forms a specific pattern of visual, auditory or other sensations of it in his/her mind. *Conceiving* means a process whereby perceptions are organised, abstracted and derived to form concepts. *Representing* means a process whereby a human being describes some of his/her concepts in a language.

Three points of view based on the semiotic framework lead to the certain comprehension of the arrangement of things: (a) the things under the observation or consideration are in the position of referent things, (b) abstract things produced from these through perception and other mental processes are concept things, and (c) the things used to signify concept things are sign things.

The significance of the semiotic framework becomes more obvious when several contexts are concerned. A thing seen as the sign thing Sign_1 in the context Cxt_1 in Figure 11 may be regarded as a referent thing in the context Cxt_2. Further, the referent thing in the context Cxt_2 is referred by another concept thing and signified by another sign thing (Sign_2 in Figure 11). The semiotic framework is here horizontally shifted[37].

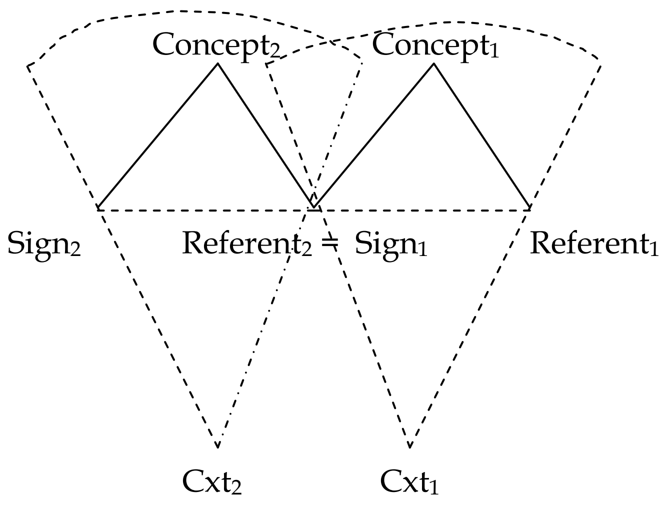


FIGURE 11 Horizontal shift in the semiotic framework

In everyday language, the term 'thing' is commonly used to mean a referent. Therefore, also in this study the term 'thing' is used to denote a referent whenever there is no danger of confusion. Otherwise, more precise terms like 'sign (thing)', 'concept (thing)', and 'referent (thing)' will be used. To distinguish between the names of various kinds of things, quotation marks are used in the following way. The signs are enclosed in simple quotes (e.g. 'John'). The names of referents are expressed in double quotes (e.g. "John"). The names of concepts

37 We call the shift horizontal when a sign is seen as a referent or a referent is seen as a sign in another context. The shift is vertical when a concept is seen as a referent (see more in Section 7.2).

are expressed without any quotes (e.g. John). We use initial capital letters in the names of the things in the examples.

3.5 Intension/Extension Ontology

The *intension/extension ontology* provides concepts and constructs to specialize the notion of a concept into more specific notions such as basic concept, derived concept, individual concept, generic concept, etc. It is established upon the concepts of intension and extension (cf. Lyons 1977). In the philosophy, intension and extension are related to the concept things, whereas in the linguistics they are envisioned as associated with the sign things. Here the concepts are defined from the philosophical viewpoint.

The *intension* or comprehension of a concept consists of all its concept predicates, shortly predicates. *Predicates* are concepts, which are used to characterize the (original) concept (cf. Hautamäki 1986, 37). They are properties of things referred by the concept. They determine the applicability of the concept[38]. For instance, the concept Animal is a predicate of the concept Cat. An intension makes up an idea, and none of its constituent parts can be removed without destroying the idea (cf. Arnauld 1964). The core intension of a concept consists of all those peculiar predicates (earmarks) that are essential to handle the concept. Usually, a working definition of the concept is based on the core intension (Bunge 1977, 67).

To put it more formally, the intension of the concept c_i, $IN(c_i)$, can be defined as follows[39]:

$$IN(c_i) = <p_{i1},\dots,p_{in}>,$$

where p_{ik} is a predicate constructed from the characteristics of the concept c_i.

The *extension* of a concept is the set of all (referent) things to which the intension of the concept applies. The things exist, have existed in the past, or will possibly exist in the future[40]. The *population* of a concept is the set of the

38 According to Wiggins (1980) there are two kinds of predicates: sortal predicates and non-sortal predicates. Sortal predicates are divided into substantial (like apple or human being) and non-substantial (like food and student) predicates, while non-sortal predicates include generic predicates such as thing and characterizing predicates such as red.

39 Predicates within the intension are interrelated in many ways, constituting complicated concept structures (cf. Kangassalo 1982, 150). To explicitly define those structures would require the mobilization of a much more comprehensive set of basic concepts. This goes beyond this study. Here the intension is defined as a whole of concept predicates. The notion of a whole is defined in Section 3.9.2.3.

40 In the literature, two kinds of views of the notion of extension are presented. According to the first view, the extension refers to the set of the existing referents (e.g. Kangassalo 1982, 155; Bubenko *et al.* 1984, 131, 286; Tsichritzis *et al.* 1982). This

existing (referent) things to which the intension of the concept applies. The formal definition for the extension of the concept c_i is:

$$EXT(c_i) = \{r_{i1}, \ldots, r_{im}, \ldots\},$$

where r_{ij} means a referent to which the concept c_i applies. The extension of a concept may be a finite set, as it is in the case of John's shirt, or an infinite set, as it is in the case of natural numbers.

If two concepts have the same intension, then they always have the same extension. In fact, they can be considered to be the same concept. Two concepts may have the same extension, but have different intensions.

A concept can be defined analytically or extensionally[41]. An analytic definition specifies the meaning by providing a concept with the intension. An extensional definition specifies the range of application, or an extension of the concept. Often extensions are sets that are too unwieldy to be observed in their entirety, so they cannot serve as a basis of practical definitions.

A *basic concept* (or primitive concept, Dominques *et al*. 1997) is a concept the intension of which is specified without using other concepts in question (i.e. based only on epistemological knowledge). A *derived concept* is a concept the intension of which is derived from predicates of other concepts.

For some concept, one corner of the meaning triangle may be absent: a person may have a concept referring to a referent thing for which he knows no sign, or he may have a sign for a concept that has no (real) extension. The concepts with no referent things are called *abstract concepts*. An example of the abstract concept is a Unicorn. It is defined as a mammal with one horn in the middle of its forehead, but because it does not exist (or exists in one's imagination), its (real) extension is an empty set. The other concepts are called *concrete concepts.* The concepts, which can only refer to one thing, are called *individual concepts* or particulars. The concepts referring to many things are *generic concepts* or universals.

In the fields of conceptual modelling (e.g. Chen 1976), information systems (e.g. Olle *et al*. 1982; Olle *et al*. 1983) and knowledge engineering (e.g. Brodie *et al*. 1984; Meersman *et al*. 1990), a generic concept is called a *type concept,* or shortly a *type.* It is normally specified by an analytical definition. Elements of the extension of a concept type are called *instances***.**

We have above defined the extension of a concept as being composed of "real" referents to which the concept refers. In some cases it is beneficial to

set is also called denotation (Sowa 1984; Stachowitz 1985). The second view regards the extension as the set of all possible referents to which the intension of the concept may apply (e.g. Carnap 1956; Falkenberg 1976). In this case, the concept of population is used to refer to the set of existing referents (Falkenberg 1976, 22; Gustafsson *et al.* 1982, 8; Falkenberg *et al.* 1998). We prefer to apply this latter view. Bunge (1974, 68) uses the terms 'total extension' and 'actual extension', or 'population' to make the distinction between the two views.

41 Smith and Medin (1981) discuss these further in terms of classical, probabilistic and prototype definition.

deploy the notion of a conceptual extension (cf. Hautamäki 1986, 37). A *conceptual extension* is composed of those referent concepts that apply to the intension of the type concept. We discuss this notion further in Chapter 7.

In Figure 12 the concepts and relationships of the intension/extension ontology are presented in the meta model.

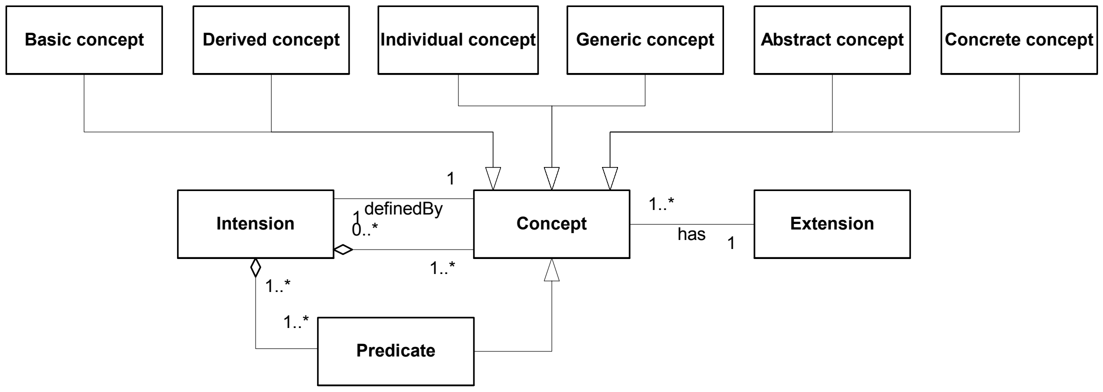


FIGURE 12 Intension/extension ontology

3.6 Language Ontology

The *language ontology* provides concepts and constructs to specify the syntax and the semantics of a language. In the discussion above a sign is used to mean any thing that signifies a concept and stands for a referent. Here, we specialize the notion of a sign and associate it with a language.

A sign is always represented in some language. A *language* is an abstract thing that is used in communication among people, between people and computers, or among parts of the computers[42]. A language is composed of syntax and semantics[43]. Syntax consists of two parts: an abstract syntax and a concrete syntax. An *abstract syntax* gives the conceptual components of a language and rules for connecting them, leaving out representational details (ter Hofstede *et al*. 1998, 520). A *concrete syntax* gives notational elements, called the symbols in the vocabulary of a language, and rules for connecting them with one another and with the concepts (cf. signification rules). *Semantics* of a language defines the relations of symbols to the referents to which the symbols are applicable (Morris 1938). It is composed of inter-subjectively agreed rules of what the different expressions mean (cf. Krogstie *et al*. 1996, 285). A *vocabulary* of a language is a non-empty and finite set of symbols (Falkenberg *et al*. 1998, 47). A *symbol* is a special sign used as an undividable part of an expression (Falkenberg *et al*. 1998, 47). Examples of symbols are "cat" and "," in the English

[42] There is also communication between animals but we ignore that.

[43] In some literature (e.g. Levinson 1983), the definition of a language includes pragmatics, too.

language and an arrow in the graphical language of UML. A linguistic *expression* is a sign of a language and a non-empty and finite "arrangement" of symbols taken from a vocabulary, constricted by the syntax and semantics of the language. An arrangement can be - like in case of a natural language – a sequence of symbols, or multi-dimensional, like in case of a graphical language.

A *formal language* is a language with a precisely defined syntax and semantics. A *semi-formal language* is a language with a precisely defined syntax (Krogstie *et al*. 1996, 285). An *information language* is neither formal nor semi-formal (Krogstie 1995, 475). There are many alternative approaches to define a formal semantics. It can be defined as translational semantics, operational semantics, denotational semantics, and axiomatic semantics (Meyer 1990; ter Hofstede *et al*. 1998, 520). Sometimes semantics is seen as a composition of two parts: static semantics and dynamic semantics. The static semantics of a language defines how an instance of a concept construct should be connected to other instances to be meaningful, and the dynamic semantics define the meaning of a well-formed construct. The meaning of an expression written in the language is defined if the expression is well formed (i.e. if it fulfills the rules defined in the static semantics).

A *label* is an elementary sign used to signify a particular concept in an elementary way. If there are several labels signifying the same concept, the labels are called synonyms. If the same label signifies several concepts, it is a homonym situation (Falkenberg *et al*. 1998, 49). If a label signifies a particular, it is called a *proper name.* If a label signifies a universal, it is called a *common noun.*

In Figure 13 the concepts and relationships of the language ontology are presented in the meta model.

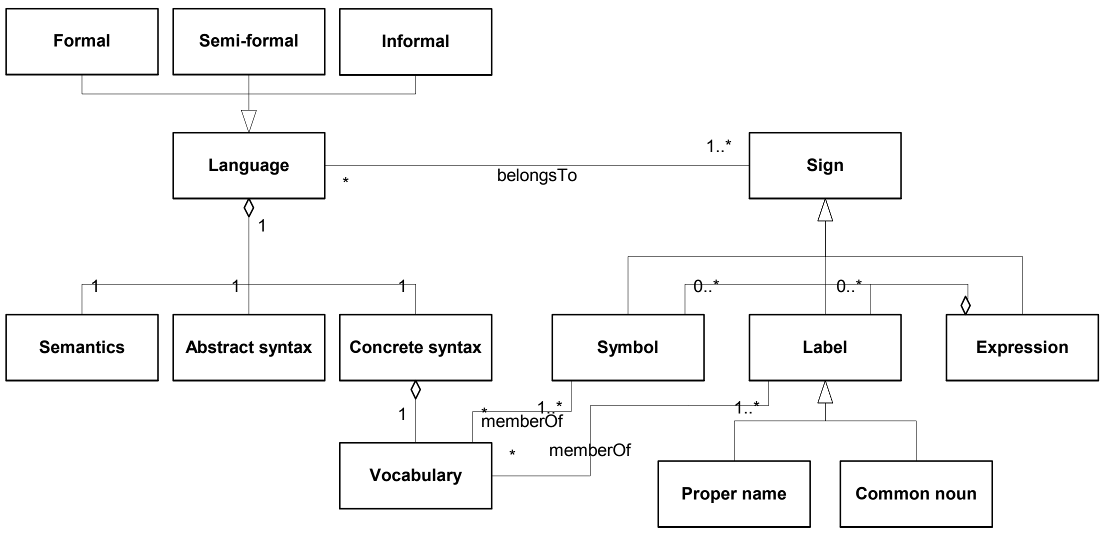


FIGURE 13 Language ontology

3.7 State Transition Ontology

Until now we have discussed the things without considering at all whether they are static or dynamic. Most of the concepts derived from the notion of a thing equally apply to structural things (e.g. person) and dynamic things, such as events (e.g. marriage ceremony) and processes (e.g. car driving). The *state transition ontology* provides concepts and constructs for conceiving static and dynamic things in reality[44].

Some things are conceived as having a static existence. A *state* is a thing, which is seen to have some duration. For instance, "John is waiting for Mary" is a state, as is also "Bell is ringing". A common type of state is a so-called *state of existence.* The state of existence can be instance-level, like "John exists" meaning that the person called John really exists, or type-level, like "Person exists", meaning that the concept Person is recognized through the more or less persistent intension. The state of existence can be related to any structural thing.

Some other things are conceived as changes of states. A *transition* is a binary relationship between two different things, called the pre-state and the post-state of that transition (Falkenberg *et al*. 1998). The pre-state of a transition is the state valid before that transition. The post-state of a transition is the state that is valid after that transition. In the transition, at least one part of the pre-state must not be included in the post-state, or vice versa (Falkenberg *et al*. 1998, 38). Examples of transitions are the relationship between the states "John is waiting for Mary" and "John and Mary enter the restaurant Estrella", and the relationship between the states "Bell is ringing" and "Bell is quiet".

An *event* is a thing, which may trigger a transition from the pre-state to the post-state (Falkenberg *et al.* 1998). It is an instantaneous happening with no (significant) duration. Examples of events are "Mary arrives" and "the button is pressed". An event does not necessarily mean the transition because it may require occurrences of other events as well as some specific conditions become true. For example, after Mary has arrived, she and John enter the restaurant, provided that they like the menu outside the door. There are several kinds of events. Some of them may be caused by transitions from certain pre-states to certain post-states.

Transitions can be related to each other to form *transition structures.* Given the transitions [transition]: t_x (s_1, s_2) and [transition]: t_y (s_3, s_4), we can distinguish between three basic transition structures (Falkenberg *et al*. 1998, 38):

1. *sequence*
   sequence(t_x , t_y) is a sequence of transitions if s_3 is a part of s_2. The resulting state-transition structure has s_1 as pre-state and s_4 as post-state.
2. *choice*
   choice(t_x , t_y) is a choice of transitions if the intersection of s_1 and s_3 is not empty, and the result is either transition t_x or t_y, but not both.

[44] Note that it may depend on the selected point of view whether a certain thing is regarded as a static thing or a dynamic thing.

3. *concurrence*
   concur(t_x , t_y) means concurrent transitions if the intersection of s_1 and s_3 is not empty and the transition t_z can be defined with the pre-state 's1 union s3 ' and the post-state 's2 union s4 '.

Further, we can distinguish between an elementary transition and a composite transition. A *composite transition* is a transition structure with a unique pre-state and a unique post-state. An *elementary transition* does not contain any transition structure (Falkenberg *et al*. 1998, 39).

For each basic concept in the state transition ontology defined above, type and instance concepts can be distinguished. Thus, we have a state type and a state instance, a transition type and a transition instance[45], and an event type and an event instance.

A *life cycle* of a thing consists of all the states, state transitions and events that are related to the existence of a thing (e.g. John), starting from the event of coming into existence (e.g. birth), continuing with changes in its states (e.g. in the marital status), discretely or continuously, and ending up with the event of termination (cf. Sakai 1983). A life cycle can be linear or contain some cyclic parts.

The main concepts and relationships of the state transition ontology are presented in the meta model in Figure 14.

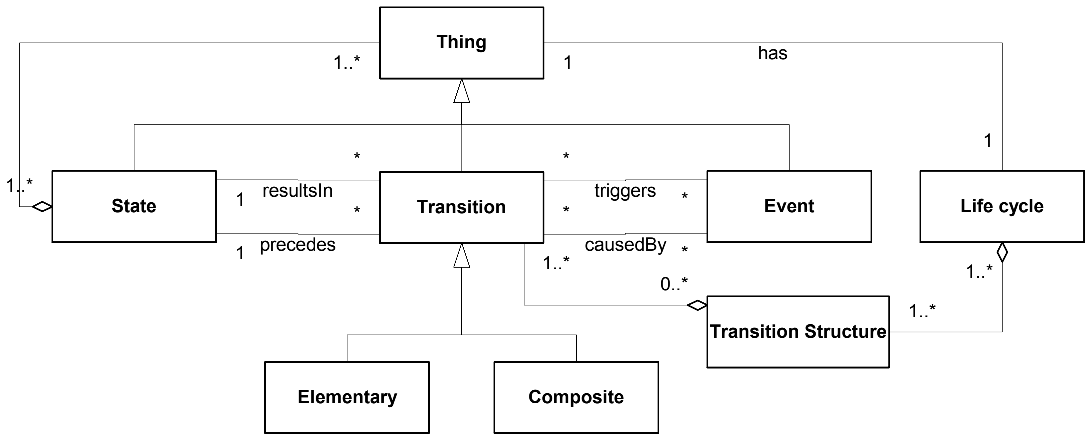


FIGURE 14 State transition ontology

3.8 UoD Ontology

Human beings become conscious of reality through the concepts and constructs they possess and apply. Depending on a situation, they select and deploy points of view through which they "look at " reality. Through the adoption of a point

45 A transition instance is called a transition occurrence in Falkenberg *et al.* (1998, 39).

of view, a basic set of concepts and constructs becomes selected. We call a *universe of discourse* (UoD) a subjective reality that becomes relevant from the point of view adopted. The notion of UoD is formally defined as follows.

$$\text{UoD} := <c_{ij}>,$$

meaning that the UoD is a whole consisting of concepts. The UoD is defined as a whole to emphasize that the concepts included in the UoD are highly interrelated (see the definition of the notion of a whole in Section 3.9.2.3). The *UoD ontology* provides concepts and constructs for perceiving and conceiving the UoD's, UoD states, UoD behavior, and UoD evolution from a certain point of view.

Based on the concepts defined in the preceding section, we can say that a *UoD state* is composed of all the related states of those (concept) things that are included in the UoD. Let the UoD be the slice of reality concerned by a secretary responsible for order processing. So the UoD consists of products, customers, orders, reorders, invoices, etc. The UoD state is composed of the states in which orders are (e.g. to be issued, processed, delivered, invoiced, or paid), in which products in the inventory are, etc.

Transitions from UoD states to other UoD states reflect the dynamic nature of the UoD. There are two kinds of transitions occurring in the UoD: extensional and intensional. Extensional transitions encompass e.g. emergence of new things of certain type (e.g. new products have been ordered), shifts of things from one type to another (e.g. from invoicedOrder to paidOrder), changes in certain qualities or quantities (e.g. decrease in quantity_on_hand), etc. Thus, extensional transitions in the UoD concern the population of the thing types in the UoD. They constitute the *UoD behavior.* Intensional transitions affect the intensions of the types in the UoD. More types may emerge and existing types may be changed, reinterpreted or vanish. In the example of order processing, intensional transitions may be caused by emergence of a new type of things, say Suppliers (earlier all the products were manufactured in the enterprise), a new way of paying (ePaying), or by some changes in the intensions of current types. These kinds of transitions constitute the *UoD evolution*[46]. It goes without saying that the borderline between extensional transitions and intensional transitions depends on the chosen point of view. The UoD of our work will also evolve along the elaborations accomplished into OntoFrame.

The concepts and relationships of the UoD ontology are presented in the meta model in Figure 15. In the figure we can see that the UoD is composed at least one UoD state. If it contains several states, it may also involve extensional transitions (the UoD behavior) and/or intensional transitions (the UoD

[46] Ramackers *et al.* (1990) distinguish between the first-order and second-order dynamics in the IS. Falkenberg *et al.* (1992a, 1992b) define the concepts of first-order evolution and second-order evolution. In the former, types can be changed, and in the latter, also meta types can be changed.

evolution) in the states. The UoD ontology, in a sense, integrates the other component ontologies in the core ontology. The notion of UoD state reflects all kinds of structural phenomena in the UoD perceived from the selected point of view. Likewise, UoD behavior and UoD evolution stand for all kinds of changes occurring in the UoD. Due to this integrative nature, the meta model of the UoD ontology is presented in a general level in Figure 15.

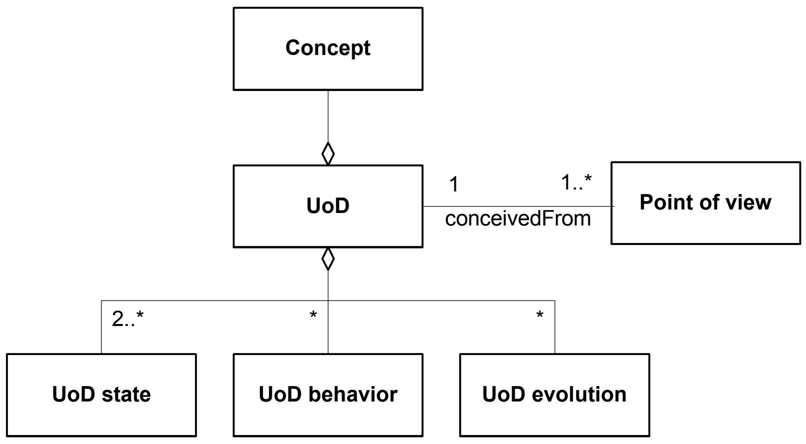


FIGURE 15 UoD ontology

3.9 Abstraction Ontology

Conceiving is a complex process that is carried out by epistemological methods to organize knowledge. Doing this, human beings unconsciously apply some abstraction, which permits one to suppress details of particular things and to emphasize those features that are pertinent to the problem at hand. Besides organizing, abstraction is a fundamental means for producing knowledge. To take a full advantage of abstraction requires that the concepts and principles underlying it are made explicit and deployed consciously.

The purpose of this section is to explicitly define the concepts and principles of abstraction. First, the main categories of the principles are presented based on the general definition of abstraction. Second, the concepts and principles in the first category, called the first-order abstraction, are intensionally and extensionally defined. Third, the principles are jointly discussed to highlight their interrelations. Fourth, the concepts and principles of the second category, called the second-order abstraction or the predicate abstraction, are defined. This section ends with the summary and discussions.

3.9.1 Abstraction Categories

Abstraction is not an easy concept to pin down. It means different things to different people and tends to be used in an incantatory rather than a scientific manner. The term comes from Latin[47] and means a withdrawal or a removal. It has long roots in the history of science[48]. It is used to mean abstraction from unnecessary details, abstraction from the "how" to the "what", abstraction from instance-level to type-level, and so on. On the other hand, abstraction is used to refer to a mental and representational process, or to a principle for, or to a result of, that process. In this study, *abstraction* is seen as the principle by which irrelevant things are ignored and the things relevant to understanding some problem of interest are uncovered. Abstraction is used to manage the complexity. This implies that some information is lost. If this is not the case, then there is no abstraction, just a transformation. The principle inverse to abstraction is called *concretizing.*

Abstraction can be performed in many ways. In the following, some examples are given. Instead of looking at individual persons (John, Mary, and Paul), the attention can be entirely focussed on Persons in general, or more specifically, on Systems Analysts and System Designers in which roles John, Mary and Paul are acting. Likewise, a Machine with its functional features may be of interest, and not its Components. For a discussion a Labor Union with its properties can be more relevant than Persons as its members. These cases exemplify abstraction, which concerns things in different scopes and meanings: e.g. as types, subtypes, wholes, and groups. This kind of abstraction that concerns the concept things and their abstraction relationships is called the *first-order abstraction.*

On the other hand, there are cases in which only some properties about the things are interesting. For instance, what a customer wants to know about a Machine may be related to its functional properties only. Characteristics related to its electrical wiring and other physical features are irrelevant for him/her. Likewise, somebody may be interested in the financial status of a Person. For someone else, the physical skills that a Person possesses are more relevant. Mental health is an example about still another aspect abstracted from a large variety of properties related to a Person. Essential to all these cases is that abstraction here concerns predicates of the given thing (a Machine or a Person). The process of abstraction is guided by a specific criterion. In the case of a Machine, the criterion is related to independence from the physical structure. The cases of a Person illustrate specific criteria related to finance, physics or healthy. The abstraction, which mainly concerns predicates of the concept things, is called the *predicate abstraction* or the second-order abstraction.

The main abstraction categories, the first-order abstraction and the predicate abstraction, are closely intertwined. As will be shown in Section 3.9.3, the predicate abstraction mostly behaves like the first-order abstraction, except

[47] Trahere, abstrahere, abstractio means "to pull", "to separate", and "to draw from".

[48] In Aristotelian philosophy, abstraction is a form of inquiry by which the mind separates "form" from "matter" in search of the "universals" (Reese 1980). To Locke is attributed the following definition: "Abstraction takes place by drawing out what is common to a group of individual things, on the basis of a comparison of their similarities and differences" (Baldwin 1940).

that it operates with the predicates (i.e. the so-called secondary things). Furthermore, these main categories of abstraction are shown to be un-orthogonal.

Abstraction is of major importance to all kinds of human action. Therefore, it is widely discussed in the AI and IS literature (e.g. Smith *et al*. 1977a; Smith *et al*. 1977b; McLeod *et al*. 1980; Brachman 1983; Winston *et al*. 1987; Mattos 1988; Iivari 1992; Motschnig-Pitrik *et al*. 1995; Mylopoulos 1998; Motschnig-Pitrik et al. 1999; Goldstein *et al.* 1999; Wand *et al.* 1999). Unfortunately, the discussion has brought out insights that are, to a considerable extent, vague and confusing. First, there are quite divergent views on the principles and concepts of the basic forms of abstraction[49]. Second, there are different opinions about what conceptual mechanisms are included in abstraction. Mylopoulos (1998), for instance, considers contextualization, materialization, parameterization, and normalization to be instances of abstraction mechanisms. It goes beyond the scope of our study to analyze and compare discrepancies among the suggestions in the literature. Our aim here is to build a uniform and coherent *abstraction ontology,* which provides concepts and constructs for abstraction. It should be strongly rooted on the concepts in more elementary parts of the core ontology defined above, and specify in a simple and comprehensible fashion structural and behavioral features of abstraction.

3.9.2 First-Order Abstraction

In this section we define four main principles of the first-order abstraction. They are: classification, composition, generalization and grouping[50]. All these principles are semantically irreducible modeling primitives that enable us to conceive reality more clearly. To provide a proper understanding of the abstraction principles, it is necessary to specify the structural properties, explicit or derived, of the principles, in a formalized way. To meet this requirement, the definitions of the principles of classification, generalization, composition and grouping are divided into three subsections. For each type of the first-order abstraction, we shall first define structural rules. In doing this, we introduce the basic concepts and constructs. Second, we shall complete the structural rules by specifying structural constraints. Third, we specify rules for the intensional and extensional derivation of predicates in each kind of abstraction[51].

[49] To give just one example, Ralyte *et al.* (2003, 108), for instance, regard abstraction as a reverse principle to instantiation, and consider specialization/generalization and aggregation/decomposition to be separate from abstraction.

[50] Goldstein *et al.* (1999, 296) calls classification and generalization with the common term 'inclusion abstraction'.

[51] Defining completely the semantics of the abstraction principles would also require the discussion of operations, which create, change and dismiss each instance structure. This study, however, does not aim at conceptualizing the environment of the ontology engineering. Therefore, we regard it adequate to use the structural constraints, including minimum and maximum multiplicities, to bring out a necessary set of behavioral properties of the abstraction principles.

3.9.2.1 Classification

Classification is the principle of abstraction by which the concept c^{ty}, called the *type,* is generated from other concepts c_i^{in}, called *instances.* By classification, features special to individual things are ignored to uncover features common to all the things of interest. Thus, the type is a generic characterization of all the predicates shared by each instance of that type. Respectively, a thing is an instance of the type if it has all the predicates defined in the type, and at least some of them are instantiated. The principle inverse to the classification is called *instantiation.* It is used to obtain instances that conform to the constraints associated with the predicates specified by the type. Classification serves two primary functions: cognitive economy (indexing instances to facilitate storage and retrieval) and inference (reasoning about instances based on the types to which they are assigned) (Rosch 1978; Smith 1988).

Consider the example of Person and John. Person is a type characterized by the predicates hasName, hasAddress and isMarriedTo. John is an instance of the type Person. It is characterized by the predicates [hasName]:John and [hasAddress]:MainStreet3.

Based on the informal definition above, we can define the *instanceOf* relationship to be the relationship between an instance c_i^{in} and a type c^{ty} as follows:

instanceOf (c_i^{in}, c^{ty})

The relationship above was defined intensionally. Defining it extensionally would require the enumeration of all the instances that are considered to apply the intension of the type. The extension of a type stands for the set of things, each of which fulfils the intension determining that type. The relationship instanceOf is non-reflexive, non-transitive and non-symmetric.

The semantics of the concepts of type (c^{ty}) and instance (c_j^{in}) can be elaborated by the following axioms[52]:

Axiom 1. for each i,j: partOf (p_i, IN(c^{ty})) -> partOf (p_i, IN(c_j^{in}))

Axiom 2. for each i,j exist k: partOf (p_i,IN(c^{ty})) -> (instanceOf (p_{ijk}, p_i) and partOf (p_{ijk}, IN (c_j^{in})))

By Axiom 1 we state that all the predicates p_i contained in the intension of the type c^{ty} are also contained by the intensions of all the instances c_j^{in}. Axiom 2 states that at least some of the predicates of the type must be included as being instantiated in the intension of each instance.

In Figure 16 the basic concepts and relationships related to the principle of classification are presented in a metamodel. Let us consider it with a simple example. Assume that the type is Person and an instance is John. Then the

[52] In the axioms we use the partOf relationship that will be defined in Section 3.9.2.3.

TypeExtension consists of all those persons, including John, who are referred to by the instance concepts of the type Person. The concepts of Type and Instance are defined by their intensions, which are composed of predicates, TypePredicates and InstPredicates, respectively. All the predicates in the TypeIntension (e.g. hasName, hasAddress, hasTwoLegs) are included in the InstIntension, some of them being instantiated (e.g. [hasName]: John).

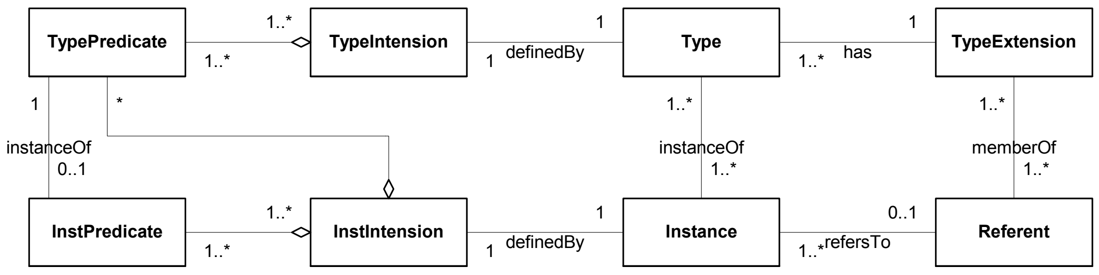


FIGURE 16 Meta model of the concepts and relationships of classification

Applying the principle of classification iteratively, a hierarchy of concepts with the instanceOf relationships is established. Hence, also a type can be regarded as an instance of some other thing. For instance, a Person is an instance of the type Concept. A Concept is defined, as we have learned in Section 3.3, by the intension composed of predicates. When instantiating a Concept into a Person, the predicates characteristics to a Person have to be specified. When instantiating a Person, in turn, still more specific predicates have to be provided. To make the difference between a type (Person) and its types (Concept), we take into use the term 'meta type'. A *meta type* is a type, instances of which are types.

Structural Constraints

Until now, the relationships between the concepts pertaining to the classification have been considered on a general level. Here, we elaborate the concepts by considering structural constraints enforced to them[53]. The constraints are presented as multiplicity constraints in Figure 16. Let us start with a simple case and then extend it with diverging assumptions.

In Figure 16 we can see that there may be one or more instances that apply to the intensional definition of the certain type. A thing can be an instance of one or more types. For a type, there is one and only one TypeExtension. It can be empty or include several referents. An instance that is related to a certain type, refers to only one referent, if any. A referent can be referred to by several instances, provided that the instances apply to different types. Likewise, a

53 We are aware that there are various classification theories and models (cf. Rosch 1978; Smith *et al.* 1981; Sowa 2000) that have different conceptions about what classes and instances are. Our aim is here to consider the structural constraints on a general level.

referent can be a member of several TypeExtensions. A TypeIntension is composed of one or more type-level predicates. An InstIntension is composed of one or more predicates that are either type-level predicates or instance predicates (InstPredicates). Instance predicates are instantiations of some type-level predicates (TypePredicates).

The meta model presented in Figure 16 is based on four basic assumptions: it reflects (a) one person's view (b) at a certain time. Further, it is assumed that (c) each instance applies to at least one type, and (d) an instance is an individual concept. Accepting or rejecting one or more of these assumptions affects how the principle of classification is understood. This means that a set of specializations of the principle of classification emerges. It is worth of noticing that cases, which deviate from the basic assumptions, are commonly conceived as exceptions and therefore either ignored or handled in an inadequate way in the literature. Let us see to what kinds of specialized forms the deviations from the assumptions lead.

First, each person interprets phenomena in reality subjectively. Consequently, from the inter-subjective viewpoint, for a type there may be several TypeExtensions. For instance, different persons may conceive a Customer in various ways. Thus, we can distinguish between the *objective classification* and the *subjective classification.* Second, comprehensions about the meanings of the concepts can evolve in time. This implies that we may have an instance, which refers to more than one referent. To cope with this, we need the dichotomy of the *permanent classification* and the *evolving classification.*

Third, the assumption of associating an instance to at least one type is questioned in several fields of information processing. Especially, in the object-oriented programming (e.g. Borning 1986; Liberman *et al.* 1988; Sciore 1989) there are approaches that, instead of the strict classification of instances, prefer to postpone typing as late as possible[54]. Consequently, we can distinguish between the *strict classification* and the *non-strict classification.*

Fourth, we consider a case in which an instance concept is a generic concept and the corresponding type concept is a meta type (see Figure 17[55]). Let the instance be Person and the type Concept. Like above, the type has its TypeExtention but an instance (e.g. Person) does not refer to one referent only. Therefore in the meta model there is InstExtension. Note that in this case TypeExtension does not contain (real) referents referred to by its instances. In some literature (e.g. Hautamäki 1986, 37) TypeExtension is said to contain

54 In the object-oriented programming (e.g. Bertino *et al.* 1995) approaches such as instance_by_instance approach, explorative programming, and template-approach are suggested. In the information modeling field Parsons and Wand (1997, 2000) suggest a two-layered approach where instances (things, objects, or entities) are not tied to particular classes. This approach affords flexibility in accommodating multiple views, evolving the views, and integrating information from different sources.

55 Note that the meta model in Figure 17 does not reflect subjective differences between conceptions about extensions, nor evolution of the extension in time.

instance concepts (e.g. TypeExtension of Concept contains Person, Car, Building etc.), meaning that TypeExtension is conceptual[56]. We adopt this view.

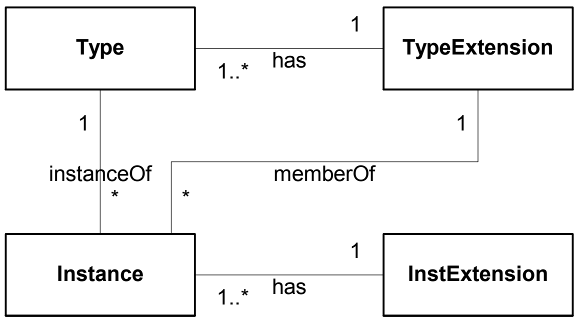


FIGURE 17 Meta model of the concepts and relationships of classification in the case of generic and meta type concepts

Predicate Derivation

As defined in Section 3.5, each concept is defined by its characterizing concepts, called predicates. These predicates are included in the intension of a concept. Some of the predicates are derived from predicates of some other concepts. This is traditionally called "property inheritance" (e.g. Smith *et al.* 1977b). In this study, "inheritance" is discussed more generally under the notion of predicate derivation. For each principle of the first-order abstraction there are specific rules for predicate derivation. Before discussing the derivation in conjunction with classification, we distinguish between two kinds of predicates, factual and definitional (cf. Mylopoulos *et al.* 1980, 188; Schrefl *et al.* 1984, 121).

Factual predicates mainly contain individual concepts. *Definitional predicates* are composed of solely generic concepts expressed in common nouns. Most of the predicates of the type are definitional, while individual concepts have also factual predicates. In fact, factual predicates are instances of some definitional predicates of the type. For instance, while John is instantiated from the type Person, Age and Salary, as predicates of Person, are instantiated into the predicates [Age]: 45 and [Salary]: 5000.

There are two kinds of predicate derivation, intensional and extensional. In the *intensional predicate derivation*, each predicate of the type is expected to apply to the corresponding instance concepts. Thus, the derivation proceeds downwards from the type to its instance concepts. Derived predicates are usually definitional although factual predicates are also possible. For example, the predicate "The manager earns at least 500 dollars more than his subordinates" should be true for each individual Manager. Sometimes in defining a new generic concept, the predicate derivation can proceed from the bottom up: individual predicates of the instances effect the selection and

[56] Note that Concept can be associated to Person with two kinds of relationships: instanceOf and isA. If the relationship between Person and Concept was the isA relationship, the TypeExtension would contain the real referents (i.e. real persons).

specification of predicates of a new type. This approach may be called "concept prototyping".

On the other hand, we can logically infer some properties of a population. Note that the populations are also concepts. Assume that Age and Salary are predicates of the concept Person. Then Average_Age and Maximum_Salary can be predicates of the type PersonPopulation. For a specific PersonPopulation, factual predicates can be derived from the factual predicates of the instances included in the extension of PersonPopulation. This kind of derivation is known as the *extensional predicate derivation.*

3.9.2.2 Generalization

Generalization is the principle of abstraction by which differences between some types, called *subtypes* c_i^{sb}, are suppressed and a new type, called a *supertype* c^{sp}, is generated based on the commonalities of the subtypes. By generalization the number of predicates in the intension is reduced, and hereby the extension is enlarged. The inverse principle, called *specialization,* is used to achieve subtypes from a supertype.[57]

By generalization one can focus on the things on a proper level in the specific/generic dimension. For example, instead of considering a Vehicle as the type, Trucks, Helicopters, Cars or Gliders may be regarded as being more appropriate for considerations. Through the subtypes it is also possible to specify, elaborate and employ the point of view that best suits the problem at hand. The supertypes offer a means to integrate "local" views. For example, features of Vendors, Customers, Employees and Employers can be joined with the supertype Person. Our work contains more examples of the use of generalization/specialization. The generic concept Thing has been first specialized by the semiotic framework into the notions of Concept, Referent and Sign. Further, Abstract and Concrete Concepts, as well as an Individual Concept and a Generic Concept are derived from the concept of Concept through specialization based on the notions of intension and extension.

The relationship between the subtype c_i^{sb} and its supertype c^{sp} is called the *isA* relationship. It is formally defined as follows:

$$isA(c_i^{sb}, c^{sp}).$$

The isA relationship is reflexive, non-symmetric and transitive[58]. The semantics of the isA relationship between the subtype c_i^{sb} and the supertype c^{sp} is elaborated by the following axioms:

---

57 ter Hofstede and van der Weide (1993b, 71) state that because specialization and generalization originate from different axioms in set theory and have a different expressive power, they are not inverse to each other. We do not agree with them.

58 In the literature various meanings to the isA relationship are given. Brachman (1983), for instance, presents a list of ten different meanings, containing a kind_of relationship, conceptual containment, role value restriction, set membership,

Axiom 3. for each i,j: partOf (p_i, IN(c^{sp})) -> partOf (p_i, IN(c_j^{sb}))

Axiom 4. for each k,j: memberOf (r_{kj}, EX(c_j^{sb})) -> memberOf (r_{kj}, EX(c^{sp}))

The first (intensional) axiom states that all the predicates of c^{sp} are also contained by the intension of each c_i^{sb}. According to the second (extensional) axiom, known also as the specialization constraint (Gomez *et al.* 2002, 469), referents r_{kj} of each subtype must be members of the extension of the supertype. The extension of the supertype and the extension of the subtype are named the *superset* and the *subset,* correspondingly.

The principle of specialization itself can be specialized based on the criteria used in specialization. Subtypes can be specified according to (a) factual predicates (called discriminators in the UML terminology (Booch *et al.* 1999)), (b) specifications given by users, or (c) operators used in the specialization (cf. McLeod *et al.* 1980; Hammer *et al.* 1981). For example, using Sex of a Person as the criterion results in the subtypes Man and Woman. A particular type of specialization by factual predicates is the case in which a subtype is derived from the population of the subtype excluding those instances that belong also to the extension of some other subtype. For instance, Unmarried may be defined as a subtype of Person, excluding Married (Gomez *et al.* 2002, 470). User-specified subtypes do not depend on any particular predicate but on the personal views or opinions stated explicitly (e.g. Excellent_student, Good_student, Poor_student). These two subtypes of specialization are intensional by their nature. An example of the extensional one is a set operator-defined subtype. It is established by set operations over the population of the type (e.g. Vehicles_owned_by_John).

If a predicate is used as a criterion for the specialization, the predicate of the supertype must be the supertype of the corresponding predicates of the subtypes (cf. Mylopoulos *et al.* 1980, 195). For example, when the predicate Age is used to specialize the supertype Person into the subtypes Adult and Child, Age is the supertype of the subtypes Age_of_Adult (e.g. 18-100) and Age_of_Child (e.g. 0-17 years).

A supertype may be regarded, from another point of view, as a subtype of another supertype. For instance, a Customer is a Person, which, in turn, is a Living_thing. Thus, the iterative use of the principle of generalization generates a hierarchy of concepts within which the concepts are interrelated with each other by the isA relationships. Each subtype hierarchy must have a unique root, and no cycles are allowed in the hierarchy[59].

predication, abstraction, etc. However, he also includes into his list some forms of abstraction that are here considered to be classification (e.g. set membership). In this study we cannot go into details of his taxonomy.

[59] The formal definitions of the aforementioned axiomatic rules are omitted here (see more in Dart *et al.* (1988, 279).

In Figure 18 the meta model of the concepts and relationships related to generalization / specialization is presented. The figure shows only those relationships that are essential to the principle of abstraction. Next, we shall consider the multiplicities of the relationships.

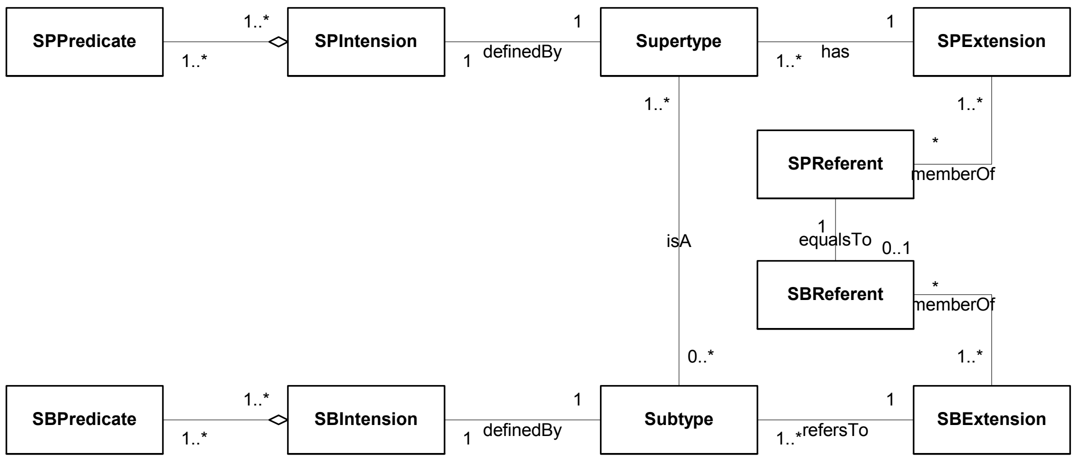


FIGURE 18 Meta model of the concepts and relationships of generalization

Structural Constraints

Just as with classification, structural constraints for generalization are also specified through the multiplicities of the relationships between the key concepts (Figure 18). Based on the kind of the isA relationship between the supertype and the subtype, we can distinguish between the *one-type specialization,* the *hierarchical specialization,* and the *lattice specialization.* In the first case, for each supertype there is only one subtype. In the second case, for each supertype there are several subtypes. In the lattice specialization, for a subtype there may be two or more supertypes. The isA relationship is depicted according to lattice specialization in Figure 18.

Based on the multiplicity of the equalsTo relationship between a SPReferent and a SBReferent, we can distinguish between the total specialization and the partial specialization. In the *total specialization,* for each SPReferent there is always one SBReferent (e.g. Hourly_Employee and Salaried_Employee). In the *partial specialization,* there may be SPReferents for which there are no SBReferents (e.g. Person can be Secretary, Technician, Engineer, or some other that is not specified).

Based on the kind of relationship between the extensions (SBExtension and SPExtension), we can define the disjoint specialization and the overlapping specialization. Let SBExtension$_i$ stand for the extension of the subtype c_i^{sb}. Specialization is *disjoint* if the following axiom holds:

Axiom 5. for each i, j (i # j):
$isA(c_i^{sb}, c^{sp})$ and $isA(c_j^{sb}, c^{sp})$ ->
$SBExtension_i \cap SBExtension_j = \{\emptyset\}$

Otherwise specialization is *overlapping.* Hence, there are four basic compound types of specialization: disjoint and total, disjoint and partial, overlapping and total, and overlapping and partial (Elmasri *et al.* 2000). In some cases, a special name is used for specialization with certain specific properties. Gomez *et al.* (2002, 469), for instance, calls specialization that is both disjoint and complete the partition.

Predicate Derivation

How predicate derivation is carried out in conjunction with generalization depends on the form of the generalization structure. Here, we first discuss predicate derivation in the hierarchical specialization and then describe how it is carried out in conjunction with the lattice specialization.

The predicate derivation, originally introduced as property inheritance in the artificial intelligence (Brachman 1983), refers to the principle by which all the predicates of a supertype are passed on to all of its subtypes. Thus, since Name is a predicate of Person, it is also a predicate of Engineer, Secretary and Trucker. Likewise, the definitional predicate "Person can be married only to one Person at a time" implicitly obliges the instances of every subtype of Person. The intensional definitions of subtypes can be further particularized by the predicates that are specific for the subtypes. So the intensional predicate derivation proceeds from the top to the bottom. The axiom for the intensional predicate derivation can be formulated as follows:

Axiom 6. for each i, j:
(isA (c_i^{sb}, c^{sp}) and partOf (p_j, IN(c^{sp}))) -> partOf (p_j, IN(c_i^{sb})),

where c_i^{sb} is a subtype, c^{sp} is its supertype and p_j is a predicate. Extensional predicate derivation in conjunction with generalization occurs such as in classification.

There are three basic modes of predicate derivation, known as the strict derivation, the default derivation and the exceptional derivation. In the *strict derivation*, the relationship isA (c_i^{sb}, c^{sp}) implies that c_i^{sb} necessarily inherits all the predicates of c^{sp}, without any exception[60]. In reality some exceptions always appear (cf. the well-known example about a Bird and a Penguin: a Penguin is a Bird but it does not fly). A way to manage them is to take the derivation as a default, and allow some of the predicates of the supertype to be overridden. This is called the *default derivation* (cf. Borgida *et al.* 1984, 92-93; Mylopoulos 1998). A special case of this is the way in which a predicate is refined during derivation. For example, a predicate of Person is "Age is between 0 and 120". A Student is a Person but its predicate is "Age is between 16 and 60"[61]. Another

[60] The strict derivation is also called the is_a –derivation (Zdonik et al. 1990). The non-strict derivation is called the is_like –derivation or the a kind_of –derivation (Wegner 1987).

[61] Mylopoulos (1998, 141) uses the term 'strict' for this kind of inheritance.

way to prepare for the exceptions is to explicitly specify the exception types as the special kinds of types (Mylopoulos *et al.* 1980; Borgida 1988, 438).

For the lattice specialization, the derivation principles presented above are refined by special rules. The most common form of predicate derivation here is the *multiple derivation* (cf. Wagner 1988, 270), which provides a mechanism to derive predicates from multiple higher-level supertypes (cf. Peckham *et al.* 1988, 161) applying special derivation strategies. By the AND-strategy, a subtype inherits all the predicates of each supertype (e.g. Amphibious_Vehicle vs. Land_Vehicle and Sea_Vehicle). In the OR-strategy the predicates of only one supertype are inherited by a subtype (e.g. Owner vs. Person and Company).

3.9.2.3 Composition

Composition[62] is the principle of abstraction by which a type, called a *whole type* c^w, is composed of other types, called *part types* c^p. Composition can also be used to abstract *part instances* into *whole instances*. For example, Work_Station is a whole type composed of part types Processor, Main_Memory, Display, etc. In the composition, predicates of and relationships between the parts are abstracted to form a whole. Besides the abstracted predicates, the intension of a whole contains predicates that characterise the whole itself. These are called emergent predicates (cf. Bunge 1997; Wand *et al.* 1999; Varzi 1996). Processing power of Work_Station is an emergent predicate because it depends on qualities of several parts. The inverse to composition is *decomposition* by which a whole (type) is decomposed into inter-related part(s) (types). A thing that cannot be decomposed is called an *elementary thing* (cf. Falkenberg *et al.* 1998).

Essential to a whole is that its parts are interrelated, in contrast to a group whose "elements" are considered to be unrelated (see Section 3.9.2.4). Parts can be characterized by one or more of the following properties (Motschnig-Pitrik *et al.* 1999, 781): (a) spatial and/or temporal proximity with respect to one another and/or the whole; (b) propagation of some structural and behavioral properties from a part to a whole; (c) propagation of some structural and behavioral properties from a whole to a part; and (d) particular ordering or constellation of parts.

Composition can concern sign things or non-sign things. For example, a Vehicle can be seen as a whole that is composed of the following parts: Identification_Number, Manufacturer, Price, Weight, Medium_Category and Propulsion category (Smith *et al.* 1977b, 114). This kind of composition is known as the *syntactic composition.* The *semantic composition* deals with the non-

[62] This is also called the aggregation (e.g. Smith *et al.* 1977a; Goldstein *et al.* 1999), whole-part (Barbier *et al.* 2001), part-whole and meronymic relation (see Opdahl *et al.* 2001b)

sign things. For example, a Train is seen as being composed of an Engine and a number of Coaches and/or Wagons.[63]

The relationship between the part (type) c_n^p and the whole (type) c^w is referred to as the *partOf* relationship. It is defined as follows:

partOf (c_n^p,c^w)

For example, a Term is a part of a Formulae, and a Coach is a part of a Train. The partOf relationship is irreflexive and antisymmetric. The semantics of the principle of composition can be elaborated by the following axioms:

Axiom 7. $IN(c^w) = E \cup X\ IN(c_i^p)$

Axiom 8. WExtension $\subseteq$ X PExtension$_i$

where c^w is a whole (type), c_i^p is a part (type) of c^w, E stands for the emergent predicates and X is used for Cartesian product. The first axiom expresses how the intension of a whole is constructed. As can be seen, the intension is much more complicated that the union of the intensions of the parts. According to the second axiom, the extension of the whole is a subset of Cartesian product of the extensions of the parts (cf. Furtado *et al.* 1986, 81)[64].

Also a whole (type) can be regarded, from another viewpoint, as a part (type) of another whole (type). For example, a Piston is a part of a Motor, and a Motor is a part of a Car. The principle of composition thus generates a composition hierarchy in which the concepts are interrelated with one another by the partOf relationships. Each composition hierarchy may have multiple roots but it cannot contain any cycles. In the hierarchy, the partOf relationship may be transitive, but only in cases where the parts and the wholes are of the same kinds. According to Winston *et al.* (1987) there are at least six kinds of whole-part relationships: (a) component / object (e.g. Processor / Computer); (b) member / collection (e.g. Conductor / Orchestra); (c) portion / mass (e.g. Slice / Pie); (d) stuff / object (e.g. Steel / Bike); (e) feature / activity (e.g. Spoon / Eating); (f) place / area (e.g. Helsinki / Finland)[65]. For transitivity of the partOf relationship, this means that, for instance, partOf(Conductor_arm, Conductor) and partOf(Conductor,Orchestra) does not imply that partOf(Conductor_arm, Orchestra) (Motschnig-Pitrik *et al.* 1999, 781).

[63] Iivari (1992) distinguishes between aggregation as a conceptual abstraction and aggregation as a linguistic abstraction. This division corresponds to our dichotomy of the semantic composition and the syntactic composition.

[64] To consider the issue more deeply would call for the introduction of concepts such as conceptual containment, sum and product (Kauppi 1967; Kangassalo 1982). It is not possible to go into such detail here.

[65] Note that only some of the kinds of the partOf relationships listed by Winston *et al.* (1987) actually are partOf relationships in our terminology.

In Figure 19 the meta model of the concepts and relationships related to the principle of composition is presented. Next, we consider the multiplicities of the relationships in more detail.

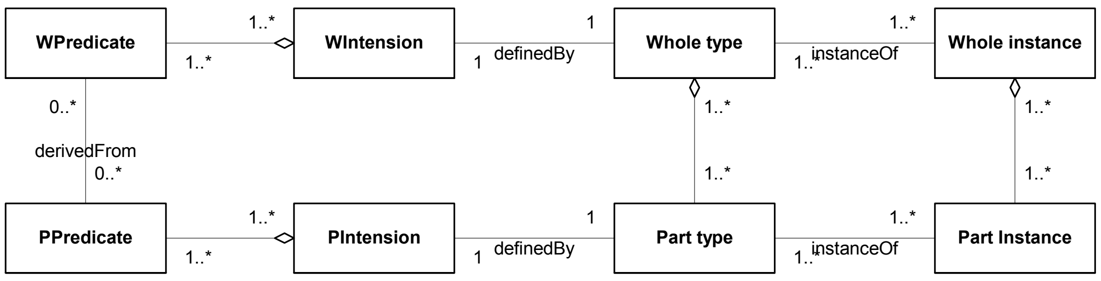


FIGURE 19 Meta model of the concepts and relationships of composition

Structural Constraints

The multiplicities of the partOf relationship depend on the nature and properties of the relationship. In the literature several classifications for the partOf relationship are presented (e.g. Winston *et al.* 1987; Motschnig-Pitrik 1993; Odell 1994; Gerstl *et al.* 1996; Saksena *et al.* 1998; Henderson-Sellers *et al.* 1999a; Motschnig-Pitrik *et al.* 1999; Snoek *et al.* 2001; Barbier *et al.* 2001; Albert *et al.* 2003). Henderson-Sellers and Barbier (1999a) and Barbier *et al.* (2001) base their classification on the division of the properties of the partOf relationship into primary properties and secondary properties. A primary property is such that any form of the partOf relationship must own it. Secondary properties are used to distinguish between special kinds of partOf relationships. The primary properties are: (a) there exist emergent predicates, (b) there exist resultant predicates, (c) the relationship is irreflexive at the instance level, (d) the relationship is antisymmetric at the instance level, and (e) the relationship is antisymmetric at the type level. Resultant properties require collaborations between wholes and parts while emergent properties do not. For instance in the case of an egg, its freshness is an emergent property and its taste is a resultant property (Barbier *et al.* 2001, 22). Irreflexivity at instance level means that no thing can be a part of the thing itself. Antisymmetry at instance level and at type level means that if a thing A is related through the partOf relationship with another thing B, then B cannot be a part of A.

The idea of the primary and secondary properties of the partOf relationship can be further refined with two dimensions distinguished by Motschnig-Pitrik *et al.* (1999). The dimensions are: degree of sharing and degree of dependence. The degree of sharing indicates to which extent a part can be shared by more than one whole. This dimension gives rise to purely static constraints. The degree of dependence means how mandatory and persistent is the relationship between a part and a whole. Based on the degree of sharing we can distinguish two extremes, namely total exclusiveness and arbitrary sharing. The partOf relationship is *total exclusive* if a thing can be a part of only one whole. For example a Motor can be a part of only one Car (see Figure 20). The partOf relationship is *arbitrary shared* if a thing can be a part in arbitrary many wholes. For example, a Figure can be a part of a Book_Chapter, an Article and a Documentation (Motschnig-Pitrik *et al.* 1999, 785).

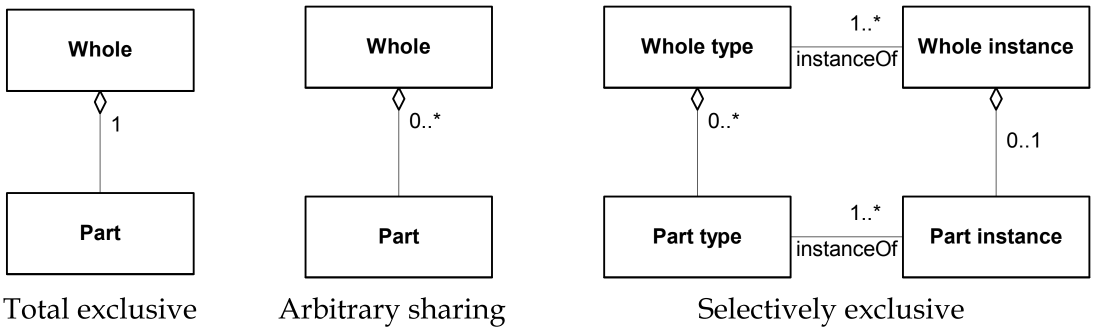


FIGURE 20 Special types of composition based on the degree of sharing

Depending on whether the partOf relationship is considered to hold between the types (type level relationship) or the instances (instance-level relationship), the impacts of the degree of sharing on the relationship vary. Type-level sharing - or interclass sharing as Motschnig-Pitrik *et al.* (1999, 785) call it - means that although a certain Motor cannot be used as a part of more than one Car, Motors of certain type can be used as parts in Cars of more than one type. An example of type-level exclusiveness is the case in which a Windows message may be part of several Windows programs, but not of anything else. Further, we can distinguish *selectively exclusive* sharing (Motschnig-Pitrik *et al.* 1999, 786), which means that a thing can be a part of one whole but of more than one alternative type (e.g. parts like Screw and Battery).

Within the dimension of the degree of dependence we have two extremes (see Figure 21). The partOf relationship can bind a part to the whole with a lifetime dependence, meaning that the existence of a part instance totally depends on the existence of the whole instance. In another case, there may be things of certain part type that are related to things not of the whole type. This kind partOf relationship is called *optional.*

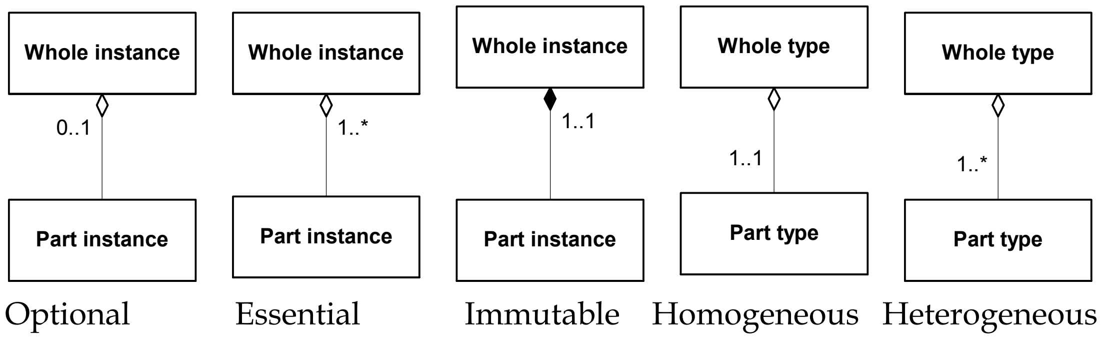


FIGURE 21 Special types of composition based on the degree of dependence or alternatively on the variety of parts

Related to the degree of dependence is the notion of essentiality. The partOf relationship between a part type and a whole type is *essential* (or mandatory) if each part instance must be interconnected to at least one arbitrary whole instance of that type. Thus, essentiality imposes a weaker constraint, and forms a prerequisite to the lifetime dependence. For instance, a Module must be a part of some Workspace. The extreme kind of lifetime dependence requires that since its "birth" the thing is permanently related to the whole (cf. the composition relationship in UML (Booch *et al.* 1999)). This kind of relationship is called *immutable*[66].

Until now we have considered the kinds of partOf relationships from the viewpoint of a part. Similar treatment can be made from the viewpoint of a whole. Hence, we can recognize the following kinds of wholes. A *homogeneous whole* is a thing that is composed of things of one part type (e.g. a Puzzle). A *heterogeneous whole* is a thing that is composed parts of several part types (e.g. a Train). A *single-part whole* is a thing that contains only one thing of a certain part type (e.g. a Train with one Engine). A *multi-part whole* is a thing that contains several things of a certain part type. A *flexible-structure whole* is a thing in which parts of some part type can be missing (e.g. a Room without a Window). A *fixed-structure whole* is a thing, which must be composed of one or more parts of all the defined part types (e.g. a Train with an Engine and at least one Wagon).

Predicate Derivation

Predicate derivation within the composition hierarchy is not so common as in conjunction with generalization. Values of quantitative predicates of non-sign things can increase or decrease when going upwards in the composition hierarchy. For example, the weight of a whole can be derived accumulating the weights of the parts. This kind of predicate is monotonically increasing (cf. Mattos 1988, 343). There are also monotonically decreasing predicates. An example of semantic derivation rules is the one declaring that the Name of a Family is determined according to the Name of the Mother or the Father (cf. Stamper 1978b, 303-304).

Intensional predicate derivation is suggested in the literature (e.g. Brodie 1978), especially in conjunction with sign things. But in the most cases the real essence of predicate derivation is misunderstood. For instance, Brodie (1978, 4) argues that "each property of a constituent (i.e. part) becomes a constituent property of the aggregate". If this would be the case, there would be no abstraction. Unfortunately, it is not possible here to discuss these issues further.

66 Motschnig-Pitrik *et al.* (1999, 789-790) further distinguish between three kinds of immutable relationships.

3.9.2.4 Grouping

Grouping[67] is the principle of abstraction by which a concept, called a *group type* c^g, is generated from other concepts, called *member types* c_i^m. Grouping can also be used to abstract a group instance from member instances. By grouping, a group (type) as a unity is examined rather than its members (member types) and the features of members (member types) are abstracted away to obtain the essentials of the group (type). The principle inverse to grouping is called *individualization* by which a member (type) is distinguished from a group (type) for a more detailed consideration.

Examples of groups are a Labor_Union whose members are Employees, a School consisting of Departments, and an Enterprise comprising Divisions. Essential to grouping is that the members of a group are of one type, and there is no internal structure between the members within a group.

The relationship between a member (type) and a group (type) is called the memberOf relationship. It is defined as follows:

memberOf (c_i^m, c^g)

meaning that c_i^m is a member (type) of a group (type) c^g . The relationship is irreflexive, antisymmetric and intransitive. The semantics of the principle of grouping can be elaborated by the following axioms:

Axiom 9. IN(c^g) = A U E , where partOf (A, IN(c_i^m))

Axiom10. for each i, exists j: (memberOf (c^m, c^g) and instanceOf (c_i^m, c^m) and instanceOf (c_j^g, c^g)) $\rightarrow$ memberOf (c_i^m, c_j^g)

At the type-level the first axiom states that the intension of a group type is composed of some part (A) of the intension of the member type (c_i^m), as well as of the predicates (E) specific to the group type as such (e.g. Name, Address and Budget of a Labor Union). Correspondingly, at the instance level the axiom states that the intension of a group is composed of some part of the intensions of the members, as well as of the predicates specific to a group. This makes the notion of a group different from the notion of a set. While two sets are equal if and only if they have the same members, this is not necessarily so for groups. Two groups having the same members, for example, two specific clubs, may differ in their internal identifiers or by the values of some predicates associated with the group. Such a predicate can be the minimum age required to become a member of a club (Motschnig-Pitrik *et al.* 1995, 153). The second axiom binds the type-level concepts and the instance-level concepts together by stating that if a type c^m is a member of another type c^g, then there is the memberOf relationship

[67] This principle is also called set membership (Falkenberg *et al.* 1998), association (Brodie 1981; Peckham *et al.* 1988; Goldstein *et al.* 1999), partitioning, and cover aggregation (Schrefl *et al.* 1984) (see also Potter *et al.* 1988).

between each instance of the member type and some instance of the group type. This holds, of course, only if the grouping is mandatory.

A membership rule for a group is either predicate-defined or user-defined. The predicate-defined membership is stated explicitly in the intension of a group type while the membership of the second type is determined instance-by-instance by a human being.

A group can be regarded, from another point of view, as a member of another group. For instance, Unions can form an organisation called United_Unions. Thus, the principle of grouping generates a hierarchy of concepts within which the concepts are interrelated with each other by the memberOf relationships. Note that as the relationship is intransitive, an Employee is a member of a Union but not a member of a United_Unions.

Figure 22 presents the meta model of the key concepts and relationships of grouping.

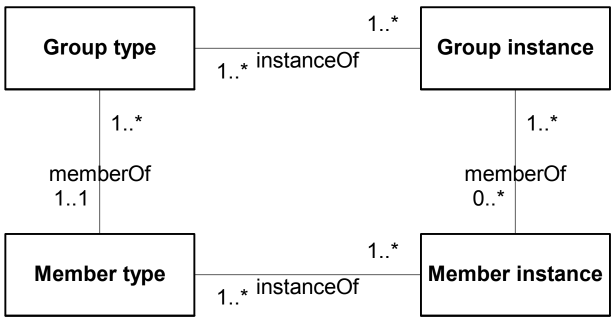


FIGURE 22 Meta model of the concepts and relationships of grouping

Structural Constraints

Based on the multiplicity constraints related to the memberOf relationships in Figure 22, we can distinguish between different kinds of grouping. We illustrate these with examples presented in the meta models in Figure 23[68]. Let us first consider type level variations. In *homogeneous grouping,* for a group type there is only one member type. In *heterogeneous grouping* a group can be formed from members of several member types. Groups can also differ in type-level sharing (Motschnig-Pitrik *et al.* 1995, 160). In *categorical grouping* a member type is related to one group type at a time. In *shared grouping* a thing can be a member type of several group types. For example, an Employee may be a member of a Union as well as a member of a Working group.

A set of kinds of grouping can be enlarged with instance-level discussions. *Disjoint grouping* means that an instance cannot be a member of more than one

68 The meta models in Figure 23 are not aimed to be complete specifications of the "subtypes" of grouping. Multiplicity constraints are presented in those ends of the partOf relationships, which are specific to the types. For instance, for a mandatory grouping it is essential that each member instance is related to at least one group instance.

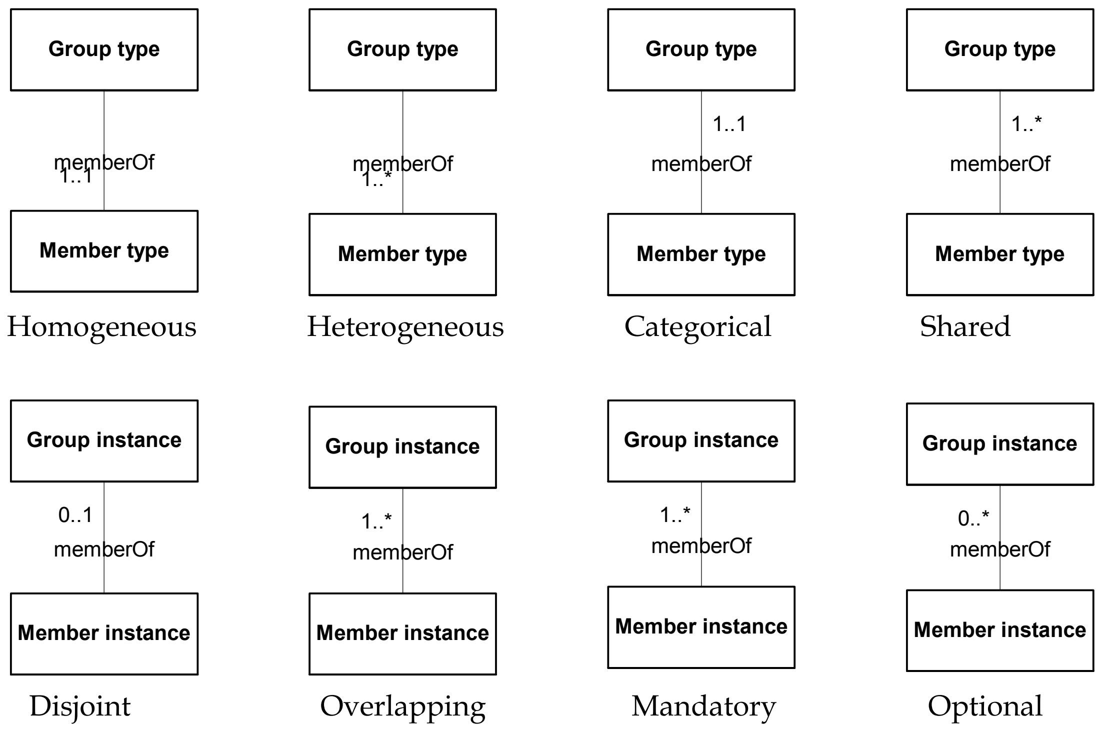


FIGURE 23 Special kinds of grouping

group (of the same or different type). In *overlapping grouping* an instance is allowed to be a member of several groups (of the same or different type). In *mandatory grouping*, each member must belong to some group, whereas in *optional grouping* an instance can exist without any memberOf relationship.

Predicate Derivation

Predicate derivation within the grouping hierarchy is addressed in only a few studies (see Mattos 1988, 338; Schrefl *et al.* 1984, 122). This would indicate that derivation is not possible in conjunction with grouping. Contrary to this opinion, we can recognize both extensional and intensional predicate derivation, although derivation rules are case-specific. Some factual predicates of a group instance can be derived e.g. by counting the number of its members (i.e. cardinality), or by applying aggregate functions (e.g. Avg_Age, Max_Salary) to the factual predicates of member instances. This kind of extensional predicate derivation proceeds in the bottom-up manner. Likewise, as Brodie *et al.* (1983, 597) suggests, predicates of a member type can establish a basis for the specification of predicates of the group type (cf. Axiom 9). The intensional predicate derivation proceeds upwards. For example, the intensional specification of a Union states that persons of a certain kind (e.g. employees) can become members of a Union.

3.9.2.5 Synthesis and Integration

In the sections above we have defined the principles, the key concepts, and the structural rules of four kinds of abstraction and discussed predicate derivation related to them. A variety of things in the reality is so immense that it is impossible to expect any set of abstraction principles to completely cover all the occurrences of abstraction to which a human being is capable in his/her observing and conceiving reality. We have identified the most basic kinds of principles of abstraction – as a matter of fact, by applying abstraction by generalization among the abstraction principles. In doing so we have not considered all the details and interpretations related to various principles of abstractions (cf. Brachman's (1983) taxonomy for the isA relationship). To sum up the discussions, we present the names of the principles, the inverse principles, the relationships and the key concepts of the first-order abstraction in Table 8.

TABLE 8 Summary of the first-order abstraction

Abstraction principle	**Concretizing principle**	**Relationship**	**Key concepts**
Classification	Instantiation	instanceOf (c_i^{in}, c^{ty})	instance, type
Generalization	Specialization	isA (c_i^{sb} ,c^{sp}).	subtype, supertype
Composition	Decomposition	partOf (c_n^{p} ,c^{w})	part (type) whole (type)
Grouping	Individualization	memberOf (c_i^{m},c^{g})	member (type), group (type)

The principles of abstraction are seldom applied one at a time. Most typically, two or more principles are deployed in an integrated fashion. For instance, Person is seen as a type, of which Mary and John are instances. At the same time Person is a supertype for Man and for Woman, in the role of Secretary in Project organization, as well as a member type of the group type Union. To illustrate the integration of the principles of the first-order abstraction at the type level, a conceptual model is presented in Figure 24, which contains examples of each abstraction relationship.

To have an integrated view on the abstraction principles on the meta level, we present below an integrated meta model of the key concepts and relationships of the first-order abstraction. As seen in Figure 25, the common basis for all the abstraction principles is the concept Thing (see Section 3.4.) from which all the concepts of abstraction are specialized. Classification is used to distinguish between the types and the instances (and the meta types). Generalization concerns the types only. The principles of composition and grouping can be formulated specifically for the types and the instances. As shown in the sub-sections above, the multiplicities of the relationships vary substantially depending on the nature and structure of abstraction. In Figure 25 we present the multiplicities of the most common structures.

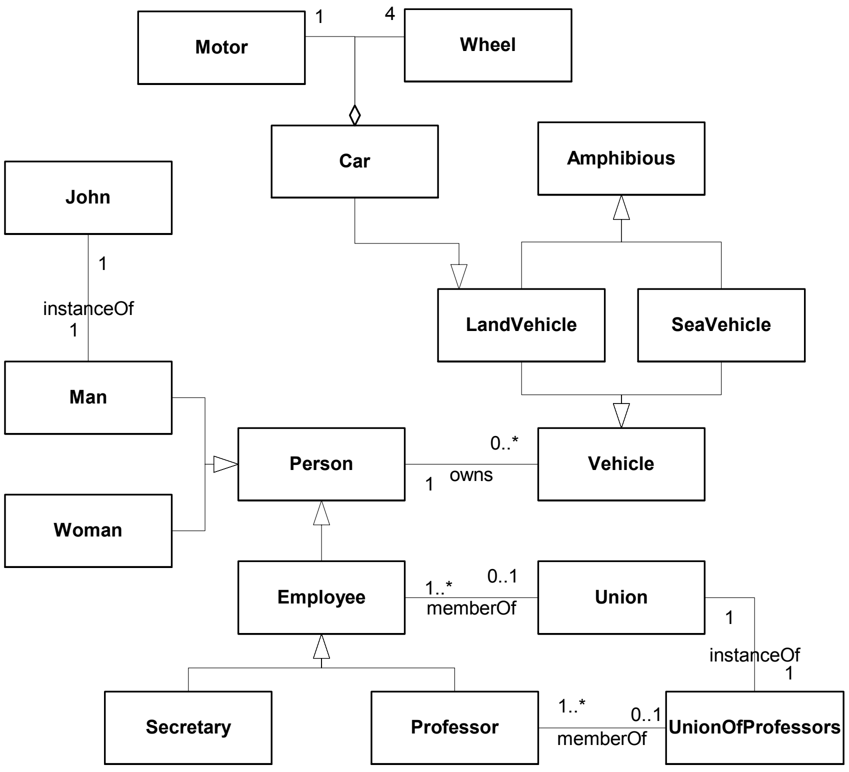


FIGURE 24 A type-level example of applying four principles of the first-order abstraction

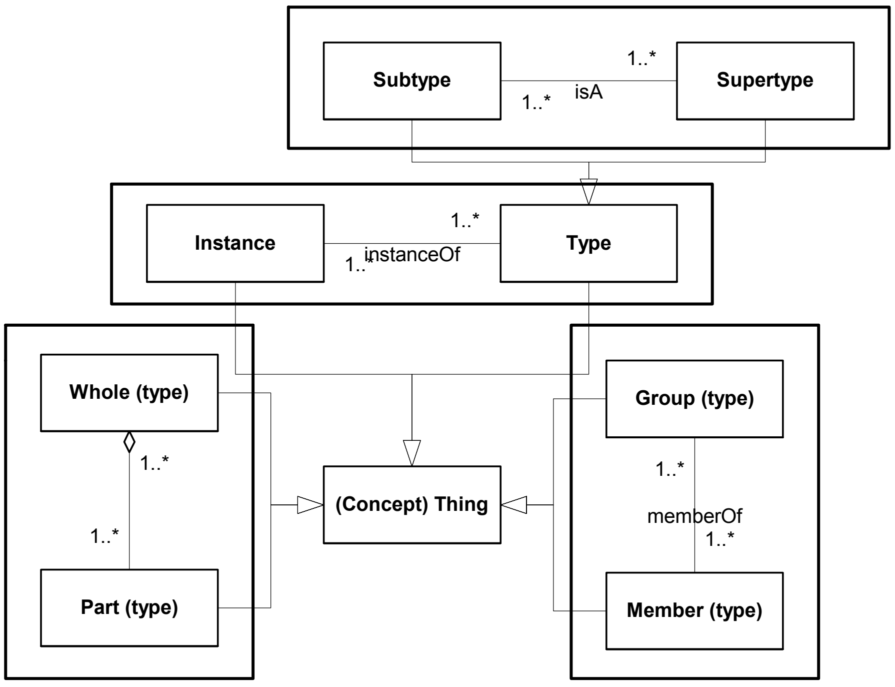


FIGURE 25 Integrated meta model of the key concepts and relationships of classification, generalization, composition and grouping

3.9.3 Predicate Abstraction

Hitherto, four principles of the first-order abstraction have been specified. The first–order abstraction can also be called the vertical abstraction because in carrying out an abstraction process one perceives and builds structures of things (i.e. instances, types, wholes, and groups) on several abstraction levels. There is, however, another kind of abstraction in which one proceeds to another direction. For example, from the chosen things all the features except those causing financial consequences are abstracted away. This kind of abstraction concerns the predicates of (concept) things and is called the predicate abstraction or the second-order abstraction or the horizontal abstraction. The purpose of *predicate abstraction* is to hide irrelevant predicates in order to reveal the predicates significant for the issues addressed.

As implied from the definitions given for the concepts of a point of view and a thing in Section 3.3, the predicates can also be treated as things. For example, the instance John of the type Person is characterized by the predicates: [Age]: 20, [Lenght]: 190, [Eyes_Color]: Blue. The predicate [Eyes_Color]: Blue can be regarded as an instance of the predicate type Eyes_Color. Further, Eyes_Color is a subtype of the predicate type Color. So, it clearly depends on the selected point of view what phenomena are regarded as things. To express explicitly the chosen point of view, the former things (e.g. Person) are called the *primary things,* and the latter things (e.g. Eyes_Color) are called the *secondary things.* Respectively, the predicates of the secondary things can be seen as the tertiary things. For example, the secondary thing Eyes_Color is characterized by the "value set" and the semantic rules for the interpretation of this secondary thing. Some of the characterizing predicates may be values[69].

We can conclude that the predicates can be regarded as secondary, tertiary, etc. things (see Figure 26), and consequently, the principles of predicate abstraction can now quite simply be derived by specializing from the principles of the first-order abstraction.

In this section, we first define the principles of predicate abstraction and give some examples of them. Second, we consider consequences of predicate abstraction to the primary things. We show that the vertical and horizontal abstractions are not orthogonal. The section ends with some conclusions.

By predicate classification the features special to individual predicates are ignored in order to uncover features common to all the predicates of interest. A predicate type is a generic characterization of all the features (i.e. secondary predicates) shared by each predicate instance. For example, [Has_Color]:Blue is

[69] The notion of a value is philosophically extremely vague (Bunge 1974) and has different meanings in various fields (cf. Kent 1978; MacLennan 1982). The definitions of values such as numbers are usually implicitly contained in some branch of the mathematical or logical theory (Bunge 1974). They may also contain a set of auxiliary rules, which express in a specific context certain properties of the value class (e.g. Max_Value). It is beyond the scope of our study to examine this issue further.

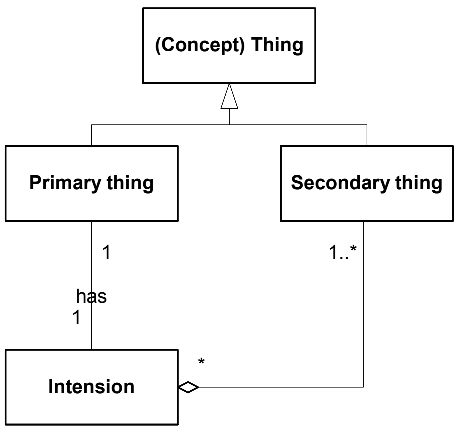


FIGURE 26 Meta model of the primary and secondary things

a predicate instance of the predicate type Has_Color. Likewise, Owned_by is a predicate type, and one of its instances might be [Owned_by]:John. In Section 3.9.2.1, the concepts of definitional predicate and factual predicate were introduced. Now we can state that the *predicate classification* means the definitionalization of the predicate instances into the predicate types, and the *predicate instantiation* means the factualization of the predicate types into the predicate instances.

Since the intensional definition of a concept is composed of predicates, it is obvious that predicate abstraction directly affects the concept formulation. Let Ball be a type and one of its concept predicates be Has_Color. Factualization of this predicate type to e.g. the predicate instance [Has_Color]:Blue creates a new primary type Blue_Ball, which is a subtype of the type Ball. The type Ball has many other predicates. Each act of factualization restricts the extension of the type so that finally we get one instance of the primary type. Thus we can say that definitionalization and factualization of predicates are closely related to abstraction and concretizing of the primary things.

By *predicate generalization* special features of predicate subtypes are ignored in order to uncover the features common to all the predicate subtypes. This results in a predicate supertype. For instance, the primary type Enterprise has the concept predicate Owned_by. Predicate subtypes of that are Owned_by_Person and Owned_by_Organization. Likewise, the predicate type Has_Weight of the primary thing type Automobile has at least two predicate subtypes: Has_Gross_Weight and Has_Net_Weight. Another predicate type of Automobile is Owned_by_Person. This can be specialized into two subtypes: Owned_by_Man and Owned_by_Woman. These examples show that the predicate generalization and specialization can have varying effects upon the corresponding primary types. Generally speaking, predicate specialization induces the first-order specialization. This is exemplified in the case of Enterprise: a new primary type Enterprise_owned_by_Person is specialized from the type Enterprise. But this does not hold for all cases. In the example of Weight no changes to the primary type is caused, because for each Automobile it applies to specify Gross_Weight and Net_Weight. What happens instead is that the specification of the predicate type is made more precise. Also non-strict predicate derivation causes exceptions to the general principle mentioned above.

By *predicate composition* a predicate as an entire construct, called a predicate whole (type), rather than its predicate part(s) (types) is/are examined. Predicate decomposition backgrounds a part of the predicate whole (type) for a more detailed consideration. For example, the predicate whole type Born_in_Date of the primary type Person can be decomposed into Born_in_Year, Born_in_Month and Born_in_Day. The same can be done for the predicate whole type Living_in_Address. Predicate composition/ decomposition has usually no direct effect upon the abstraction of the corresponding primary types. Only predicate(s) (types) may become more detailed.

By *predicate grouping* a predicate group (type) rather than its predicate member(s) (types) is/are examined. The inverse process is predicate individualization. For example, the predicate type Has_Color_Composition[70] contains a reference to the predicate group type Color_Composition, which can be individualized into member colors. It is worth of noting that as far as only the colors themselves are concerned we use grouping. But if portions that colors have in the composition are of any importance, the predicate abstraction follows the principle of composition resulting in the predicate part types Color and Portion. Predicate grouping/individualization has no direct effect on the first-order abstraction.

To conclude, we have defined the key concepts for each principles of the predicate abstraction and given illustrative examples. The principles, the relationships and the key concepts are summarized in Table 9. The discussion has been firmly based on the presumption that the predicates are special kinds of concept things (i.e. isA (p, c)). This has given us a reason to argue that the structural rules and constraints, as well as the rules for predicate derivation given for the primary things, hold for the predicates, too. Due to this generative

TABLE 9 Summary of the predicate abstraction

Predicate abstraction principle	**Relationship**	**Key concepts**
Predicate classification	instanceOf (p_i^{in}, p^{ty})	predicate instance, predicate type
Predicate generalization	isA (p_i^{sb} , p^{sp})	predicate supertype, predicate subtype
Predicate composition	partOf (p_i^{p}, p^{w})	predicate part (type), predicate whole (type)
Predicate grouping	memberOf (p_i^{m}, p^{g})	predicate group (type), predicate member (type)

[70] Color composition is a common expression in English. Unfortunately, it refers to the other principle of abstraction, viz. composition.

nature of the abstraction framework, the space required for formulating the predicate abstraction is less than the significance of this issue would suggest.

3.9.4 Summary and Discussions

In this section we have defined the abstraction ontology that is based on two basic categories of abstraction. The first-order abstraction concerns the primary things and their basic relationships. The second-order abstraction or the predicate abstraction concerns the predicates and their relationships. We have distinguished between four basic abstraction principles: classification, generalization, composition, and grouping. For each type of principle we have provided the definitions of the key concepts, the structural rules, and discussions about the predicate derivation. This has been done first for the first-order abstraction, and then for the predicate abstraction. Finally we have also discussed the relationships between the first-order abstraction and the predicate abstraction. To promote the consistence and coherence of the abstraction ontology we have presented the key concepts and relationships of each abstraction principle in meta models. As far as we know, this has not been done in any earlier work.

As mentioned above, in the IS literature there are quite divergent views on the principles and concepts of the basic forms of abstraction. It has not been possible for us to analyze or compare divergent conceptions here. Instead next we shortly consider conceptual mechanisms that are suggested to involve abstraction, although they do not belong to any single basic category.

Mylopoulos (1998) includes into the abstraction "mechanisms", besides classification, generalization, and aggregation, also materialization, contextualization, normalization, and parameterization. Materialization, originally introduced by Pirotte *et al.* (1994), relates a class, such as Car_Model, to a more concrete class, such as Car. The Car_Model class contains information, such as Model_Name, Sticker_Price, and available options for Engine_Size. The class Car models information about individual cars, such as Manufacture_Date, Serial_Number, and Owner. The formal properties of materialization constitute a combination of those of classification and generalization (Pirotte *et al.* 1994, Pirotte *et al.* 2004).

The term contextualization is used to cover a large variety of ways to apply a certain point of view. It means "an abstraction mechanism which allows partitioning and packaging of the descriptions being added to an information base" (Mylopoulos 1998, 142-143). Mechanisms to apply points of view comprise partitioned networks in knowledge representation (e.g. Hendrix 1979), workspaces, versions and configurations in CAD and software engineering (e.g. Katz 1990), views in data bases (e.g. Elmasri *et al.* 2000; Motschnig-Pitrik *et al.* 1996), perspectives in hypertext bases (e.g. Prevelakis *et al.* 1993), and viewpoints (e.g. Finkelstein *et al.* 1992). In terms of our abstraction ontology, contextualization means abstraction with multiple principles of the first-order abstraction as well as of the predicate abstraction.

Normalization means "normal-case first abstraction" (Borgida 1985) where the common things, states and events are first modeled, while special and exceptional situations are later dealt with. Exceptions are discussed in programming languages (e.g. Borgida 1988) as well as in specifications (e.g. Schoebbens 1993). Abstracting exceptional things and aspects away is a very complicated process, which can only partly be supported by the basic principles of abstraction. Parameterization is a common mechanism to enhance reusability in programming and specification languages. Parameters, such as resource and customer, can be used in programs, which are to be used, for instance, in a library (book and libraryUser) and a car rental company (car, renter). Parameterization can be modeled by the principles of generalization.

There are also presentations that discuss the so-called "role-abstraction" (e.g. Albano *et al.* 1993; Li *et al.* 1994; Wieringa *et al.* 1995; Kaasboll 1995; Steinmann 2000; Dahchour *et al.* 2002). Dahcjour *et al.* (2002), for instance, present a generic role model, which is based on the role relationship that connects a superclass (e.g. Person) to more than one role classes (e.g. Student, Employee). Unlike in generalization, each instance of the superclass can be related to any number of role classes.

Finally, Olive (2002, 675) uses the term 'generic relationship type' to mean relationship types that may have several realizations in a domain. He includes abstraction relationships such as the partOf and memberOf relationships but also the materialization relationship (e.g. Pirotte *et al.* 1994) and the owns relationship (e.g. Yang *et al.* 1994).

These examples concretely show how complex and multifaceted concept abstraction is. They also exhibit to what extent abstraction and its all modes have been addressed and used in knowledge representation, information systems, data bases, programming languages, etc. We believe that our ontology, although covering only two categories and four basic principles promotes the understanding and application of abstraction. Parts of abstraction that are excluded from the abstraction ontology are partly addressed through the notions of point of view (Section 3.3) and of perspective (Chapter 6).

3.10 Comparative Analysis

Conceiving reality and its phenomena with fundamental concepts and constructs is widely discussed in the literature. It is not possible for us here to make an extensive review of the relevant literature. Instead, we first present a general categorization of the literature and give some examples of each category. Second, we select two presentations for a more detailed analysis. We define the goals of the analysis, describe the selected presentations and compare them with one another, as well as with the core ontology.

3.10.1 Categorization of Relevant Literature

The relevant literature can be sub-divided into three categories. The first category comprises universal, top-level ontologies that aim to provide conceptual foundations for the representation of common sense knowledge. Examples of the presentations belonging to this category are the top-level ontologies of Bunge (1977), Chisholm (1996), Feibleman (1951), Brody (1980) and Tiles (1981). This category also includes upper ontologies developed for artificial intelligence and knowledge representation, such as Cyc (Lenat *et al.* 1990), SUMO[71] (Suggested Upper Merged Ontology), and Sowa's categorization (Sowa 2000).

The second category contains comprehensive conceptual frameworks and ontologies established for the IS field. The best representatives in this category are the Frisco framework (Falkenberg *et al.* 1998) and the BWW model (Wand 1988a; Wand *et al.* 1990a). Other presentations are Telos (Mylopoulos *et al.* 1990), Opdahl *et al.* (1994), and Krogstie (1995).

The third category is composed of those presentations in the IS field that cover only some part of our core ontology. From a large array of this kind of presentations we mention here only those that address the abstraction ontology. We can distinguish two kinds of presentations. First, there are suggestions that give a comprehensive consideration for multiple principles of abstraction. The most notable presentations of this kind are Schrefl *et al.* (1984), Mattos (1988), Kaasboll (1995), Mylopoulos (1998), and Goldstein *et al.* (1999). Second, there are presentations that address only some of the abstraction principles, such as classification (e.g. Parsons *et al.* 2000), generalization (e.g. Smith *et al.* 1977b; Brachman 1983; Gomez *et al.* 2002), composition (e.g. Smith *et al.* 1977a; Winston *et al.* 1987; Mostchnig-Pitrik 1993; Motschnig-Pitrik *et al.* 1999; Varzi 1996; Gerstl *et al.* 1996; Guarino *et al.* 1996; Henderson-Sellers *et al.* 1999a; Opdahl *et al.* 2001b; Barbier *et al.* 2001), and grouping (e.g. Motschnig-Pitrik *et al.* 1995).

3.10.2 Targets and Goals of Analysis

From the aforementioned categories we regard the second one (i.e. comprehensive conceptual frameworks and ontologies for the IS field) as the most relevant for our purposes. From instances of this category we select two most prominent presentations for our comparative analysis. They are the Frisco framework and the BWW model. The Frisco framework (Falkenberg *et al.* 1998) is the result of the work of the IFIP WG 8.1 Task Group FRISCO (FRamework of Information System COncept), established in 1988. The framework "provides a reference background for scientists and professionals in the IS field comprising a consistent and coherent system of concepts and a suitable terminology.." (Falkenberg *et al.* 1998, 2). The Frisco framework has been selected for the analysis because it is the most serious attempt to reach an agreement on the common terminology and conceptual foundation in the IS field. It lasted for

[71] http://suo.ieee.org/

approximately ten years and involved a large community of distinguished scientists. The BWW model (Wand 1988a; Wand *et al.* 1989; Wand *et al.* 1990a; Wand *et al.* 1990b; Wand *et al.* 1993; Wand *et al.* 1995b) is based on Bunge's ontology (Bunge 1977). It consists of three models: the representational model, the state-tracking model, and the good decomposition model. We consider here only the representational model. It is aimed to be used "as an ontology to define the concepts that should be represented in an IS modeling language, that is the semantics of the language" (Wand *et al.* 1995a, 287). The BWW model has been selected for the comparative analysis because it is the only presentation in the IS field that is firmly based on a universal ontology. The model has also been widely applied for more than ten years.

The comparative analysis is composed of two parts. In the first part we present an overview of objectives, ontological positions, and structures of the presentations. Objectives refer to purposes for which the presentations are aimed. Ontological positions are analyzed by using the trichotomy of objectivistic, constructivistic and mentalistic viewpoints (Stamper 1992b; Falkenberg *et al.* 1998). The basic structure means a way in which the sets of concepts are sub-divided and organized in the presentations. The aim of the second part is to reveal the coverage and emphases of the presentations. For this purpose we distinguish their key concepts and constructs, and using the core ontology as a "yardstick" we consider which parts are missing in the presentations and which parts are missing from our ontology, compared to the presentations. Emphases are seen in numbers of concepts and constructs defined for certain domains.

3.10.3 Results of Analysis

3.10.3.1 Overview

Table 10 summarizes the general features of the core ontology, the Frisco framework and the BWW model in terms of objectives, ontological positions and basic structures. The core ontology is aimed to provide key concepts and constructs for conceiving, understanding, structuring and representing fundamentals in reality, and to derive more specific concepts from them. It has been constructed based on the assumptions of the constructivist position. The core ontology is composed of seven parts, each having its own particular view on reality. The generic ontology provides the minimal set of concepts to conceive things, their relationships and properties in reality. The semiotic ontology applies a universal theory, the semiotics, to distinguish between three realms. The extension/intension ontology serves as a conceptual mechanism to specialize the notion of a concept and to define its semantic meaning. This ontology is needed in particular for establishing the abstraction ontology. The language ontology provides concepts for defining the syntax and semantics of a language. The state transition ontology is composed of concepts and constructs for the recognition of dynamic phenomena in reality. The UoD ontology is composed of consolidated concepts through which reality can be perceived as a totality, whether static, dynamic or evolving, determined by a selected point of view. Finally, the abstraction ontology serves a rich collection of concepts and constructs with which human beings can cope with the complexity of reality.

TABLE 10 Summary of the objectives, ontological positions and basic structures of the presentations

	Core ontology	**Frisco framework**	**BWW model**
Objective	"to provide the key concepts and constructs to conceive, understand, structure and represent fundamentals in reality"	"to provide an ordering and transformation framework to relate different IS modeling approaches to each other" (Falkenberg *et al.* 1998, 1)	"is aimed to be used as an ontology to define the concepts that should be represented by a modeling language, that is the semantics of the language" (Wand *et al.* 1995a, 287).
Position	Constructivist position	Constructivist position	Objectivist position
Structure	*Modular structure*: generic ontology, semiotic ontology, extension/intension ontology, language ontology, state transition ontology, UoD ontology, abstraction ontology	*Layered structure:* fundamental layer, layer of actors, actions and actands, layer of cognitive and semiotic concepts (+ two more layers)	No (explicitly) defined structure

The Frisco framework is aimed "to provide an ordering and transformation framework to relate different IS modeling approaches to each other" (Falkenberg *et al.* 1998, 1). It is also hoped that "based on the definitions of basic concepts, it will be possible to achieve a clear understanding of the various kinds of information systems" ... "and to provide a bridge between the various disciplines involved, in particular between computer science and social sciences" (ibid p. 1). The framework has adopted the constructivist position according to which reality exists independently of any observer, but the observer is aware of the fact that we only have access to our own (mental) "conceptions" (ibid p. 26). The only concepts referring to the physical reality are domain (a domain comprises a "part" or "aspect" of the "world" under consideration (ibid p. 46)), domain component, and domain environment. All the other concepts are related to the mental reality.

The framework is composed of several layers. At the bottom there is the fundamental layer, which "specifies universal and generic view" (ibid p. 32). The layer contains concepts such as thing, relationship, entity, type, state, transition, and time. From the concepts at this fundamental layer, all the other concepts are derived. The other layers are: the layer of actors, actions, and actands, the layer of cognitive and semiotic concepts, the layer of system concepts, and the layer of organizational and information systems. From these layers three lowest ones contain concepts that correspond to concepts in our core ontology. The layer of actors, actions and actands is motivated with the need to consider explicitly what causes transitions (i.e. actions performed by actors). The cognitive and semiotic concepts, in turn, are needed to represent one's conceptions of some domain.

The BWW (Bunge-Wand-Weber) model is "aimed at to be used as an ontology to define the concepts that should be represented by a modeling language, that is the semantics of the language" (Wand *et al.* 1995a, 287). It can be used to evaluate the capability of a method by examining the mapping between ontological constructs and the constructs of the method. The fundamental premise is that any modeling grammar must be able to represent all things in the real world that might be of interest to users of information systems; otherwise, the resultant model is incomplete (Rosemann *et al.* 2002, 78). The model has been applied, for instance, to evaluate relational and object-oriented schema diagrams (Sinha *et al.* 1995), NIAM model (Weber *et al.* 1996), various ISAD grammars (Green 1997), modeling constructs within the OPEN Modeling Language (OML) (Opdahl et al. 2001a), and views in the Architecture of Integrated Information Systems (ARIS) (Green *et al.* 2000).

The BWW model is based on Mario Bunge's ontology (Bunge 1977), which is premised on assumptions of objectivists, who believe that the real world exists independently of any observer and merely needs to be mapped to adequate descriptions. The BWW model distinguishes between the real word, composed of things, and the world we know via the models of things, that is the mental world.

In the BWW model no clear internal structure among the key concepts has been defined. However, there is a set of fundamental concepts (i.e. thing, property, state, transition, and stable state) from which all other concepts have been derived (Wand *et al.* 1990b; Wand *et al.* 1995b, 211). We can also distinguish concepts that are related to (a) things and their properties, (b) states (e.g. laws and lawful states), (c) history (e.g. event, coupling), and (d) systems (Wand *et al.* 1990b). These divisions are, however, not based on any explicitly defined principle.

3.10.3.2 Concepts and Constructs

In this section we present the key concepts and constructs in the core ontology, the Frisco framework and the BWW model. Since the core ontology has already been defined in the preceding section, we present here only the integrated meta model of its concepts. For the two other presentations, we present the definitions of the concepts and constructs as they are given in the original references and described in meta models.

A. Core Ontology

The meta model in Figure 27 gives an overview of the structure and contents of the core ontology and illustrates its generative and modular nature. We can see that the generic ontology with the fundamental concepts gives the basis from which first the semiotic concepts (i.e. sign, concept, and referent) and later the other particular concepts are specialized. The semiotic ontology is in an important position in differentiating things into signs, concepts and referents. The extension/intension ontology serves as the basis for the fundamental categorizations of concepts on which the abstraction ontology, in turn, builds

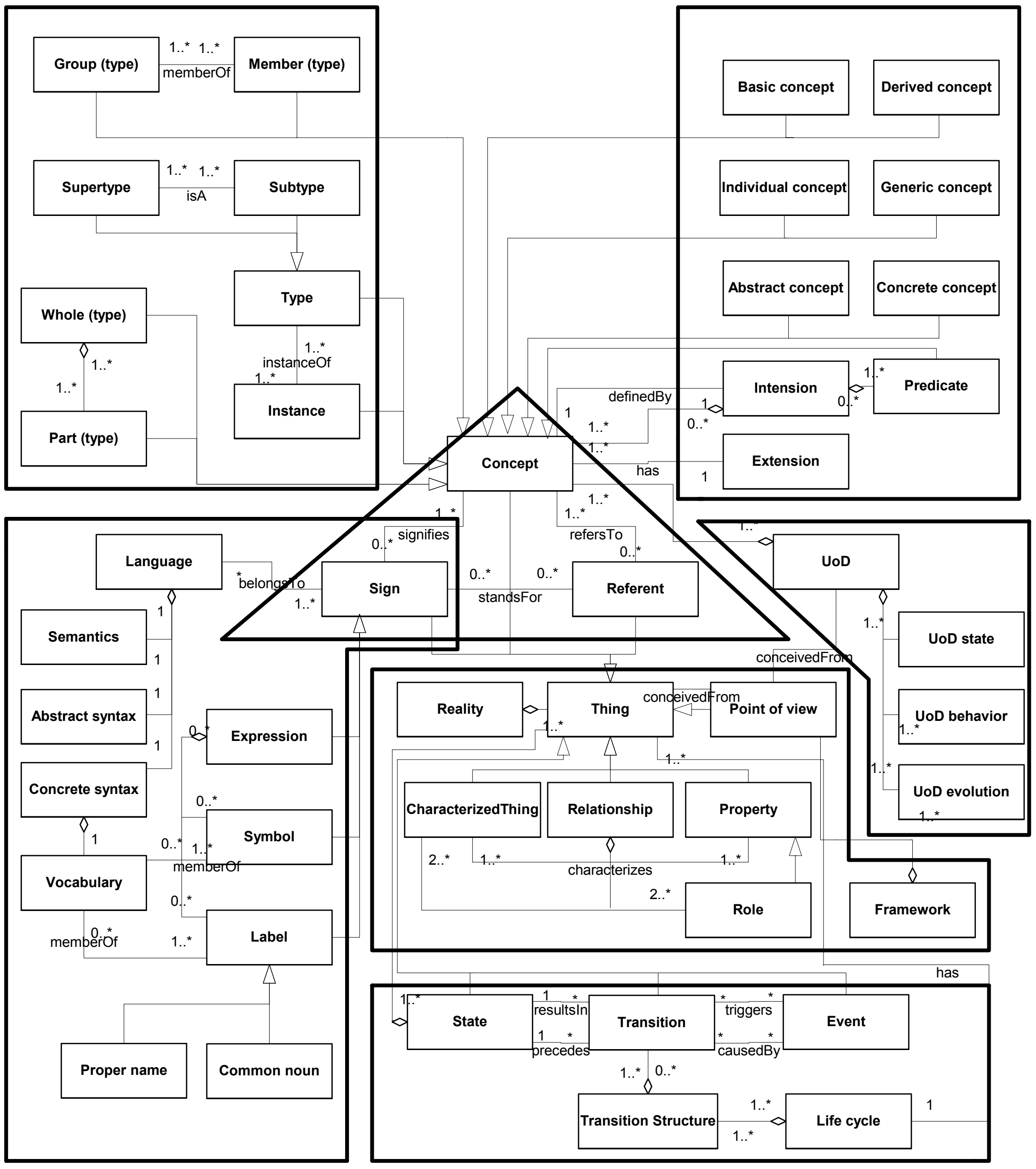


FIGURE 27 Core ontology

more specialized concept structures. The state transition ontology concerns the behavior of things. The UoD ontology gives means to integrate relevant conceptions into a totality, known as the universe of discourse, which is perceived either from the static, behavioral or evolutionary viewpoint. The language ontology derives its language-related concepts from the concept of a sign. In the large abstraction ontology the common base of the principles of abstraction is the concept thing, from which all the concepts of abstraction are specialized. Classification is used to distinguish between types and instances. Generalization concerns types. Composition and grouping can be applied to types and instances. The principles, concepts and constructs of the first-order abstraction mostly apply to the predicate abstraction, which concerns predicates in the intentions of the primary concept things.

B. Frisco Framework

The Frisco framework is composed of almost 100 well-defined concepts on five layers. The layers are: the fundamental layer, the layer of actors, actions, and actands, the layer of cognitive and semiotic concepts, the layer of system concepts, and the layer of organizational and information system concepts. From the concepts, those on the three lowest layers are mostly relevant for the analysis here. Next, we first present the definitions of the comparable concepts, as they are given in Falkenberg *et al.* (1998), and then describe the meta model of the concepts and constructs.

A *thing* is any part of a conception of a domain. The set of all things under consideration is the *conception* of the domain. A *predicator* is a thing, used to characterize or qualify other things. A *predicated thing* is a thing being characterized or qualified by at least one predicator. A *relationship* is a special thing composed of one or several predicated thing(s), each one associated with one predicator characterizing the role of that predicated thing.

A *set membership* is a special relationship between a thing, characterized by the predicator called 'has-element', and another thing, characterized by the predicator called 'is-element-of'. An *elementary thing* is a thing, not being a relationship and not being characterized by the predicator 'has-element'. A *composite thing* is a thing, not being an elementary thing. An *entity* is a predicated thing as well as an elementary thing. A *type* of things is a specific characterization applying to all things of that type. A *population* of a type of things is a set of things, each fulfilling the type characterization. An *instance* of a type of things is an element of a population of that type.

A *transition* is a special binary relationship between two different composite things, called the pre-state and the post-state of that transition. A *state* is a composite thing, involved as pre-state or as post-state in some transition. Transitions can be related to each other, to form *state-transition structures*. A *composite transition* is a state-transition structure with a unique pre-state and a unique post-state. A *transition occurrence* is a specific occurrence of a transition. A *rule* determines a set of permissible states and transitions in a specific context.

An *actor* is a special thing conceived as being "responsible" or "responsive" and as being able to "cause" transitions. An *action* is a transition involving a non-empty set of actors in its pre-state. An *actand* is a thing involved in the pre-state or post-state of an action. A human actor is a responsible actor.

A *domain* comprises any "part" or "aspect" of the "world" under consideration. A *perception* is a special actand resulting from an action whereby a human actor observes a domain with his senses and forms a specific pattern of visual, auditory or other sensations of it in his mind. A *perceiving action* is a special action of a human actor having a domain as input actand and a perception as output actand. A *perceiver* is a human actor involved in a perceiving action. A *conception* is a special actand resulting from an action whereby a human actor aims at interpreting a perception in his mind. A *conceiving action* is a special action of a human actor having a perception and possibly some action context as input actand and a conception as output actand. A *conceiver* is a human actor involved in a conceiving action.

A *symbol* is a special entity used as an undivisible element or a representation in a language. An *alphabet* of a language is a non-empty and finite set of symbols. A *symbolic construct* is a non-empty and finite "arrangement" of symbols taken from an alphabet. A *language* is a non-empty set of permissible symbolic constructs. A *representation* is a special actand describing some conception(s) in a language. A *representation action* is a special action of a human actor having a conception and possibly some action context as input actand(s) and a representation as output actand. A *representer* is a human actor involved in a representing action. A *label* is a special entity being an elementary representation and used to referring to some conception in an elementary way. A *reference* is a special binary relationship between a conception and a representation used to refer to that conception.

In Figure 28 concepts on three layers of the Frisco framework are metamodeled. Bold lines are used to show boundaries between the layers. The upper part contains the concepts of the fundamental layer. The lower part contains the cognitive and semiotic concepts (i.e. the third layer). The concepts not surrounded by the bold line belong to the layer of actors, actions and actands (i.e. the second layer)[72]. Although these concepts do not belong to the scope of the core ontology, we have included them in the meta model, because they form the important "bridge" through which the relevant cognitive and semiotic concepts have been defined. The meta model is not aimed to be a precise presentation of relationships between the concepts for two reasons. First, the definitions in the Frisco Report (Falkenberg *et al.* 1998) do not provide an unambiguous basis for such a presentation, in particular as to the multiplicity constraints. Second, we are not going to go into such details in our analysis. Based on the meta model we can now make the following general remarks about the Frisco framework.

[72] Note that this layer in the Frisco framework contains many other concepts. They are not included here because they correspond to the context ontology in OntoFrame.

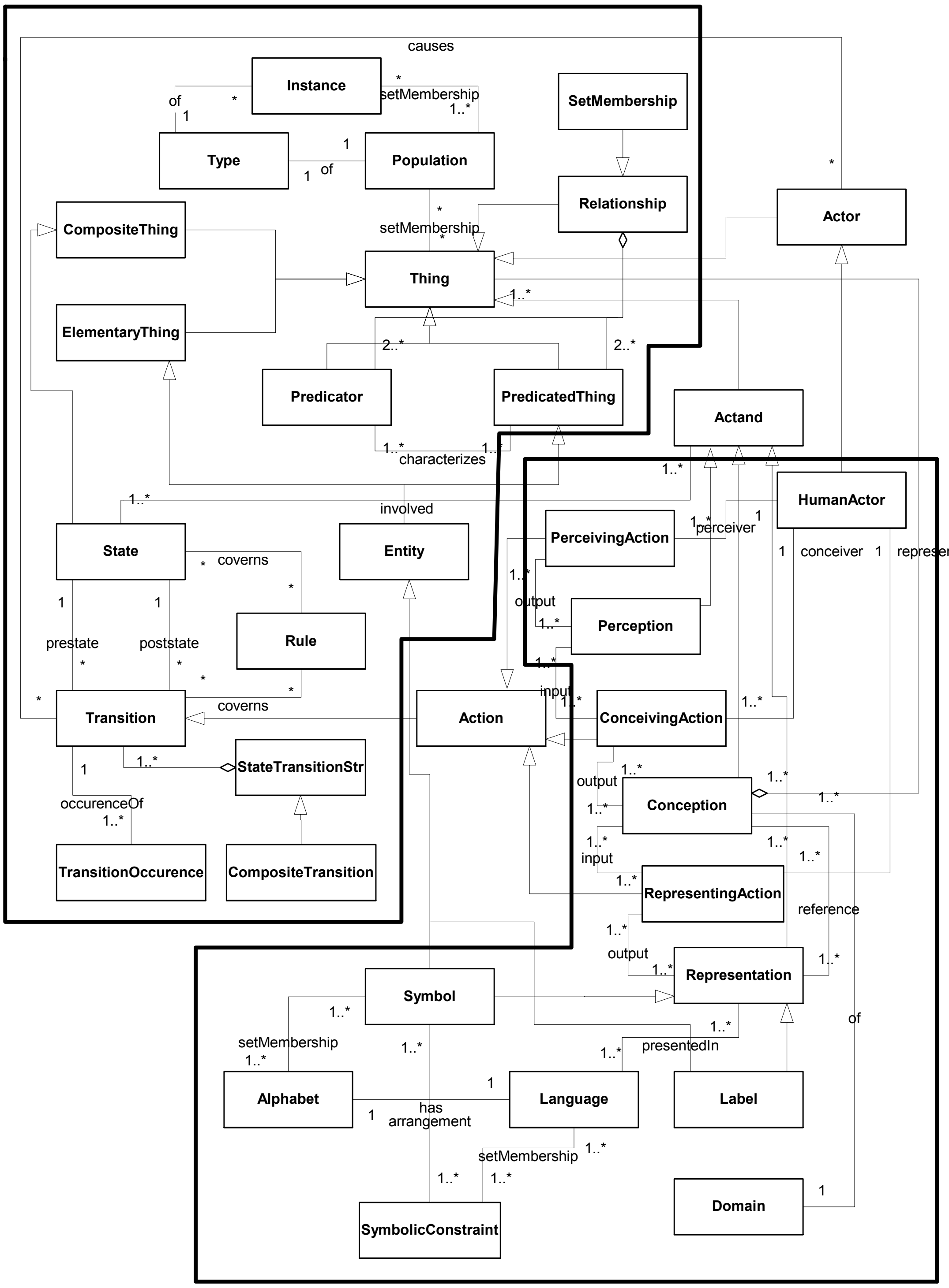


FIGURE 28 Meta model of the relevant part of the Frisco framework

On the fundamental layer, the most focal concept is thing, which is defined via the relation to conception, that is to say, to concern the mental world. Connections between the mental world and the physical world remain vague. The most fundamental concepts (thing, relationships, predicator, predicated thing) are related to one another in the way that is similar to ours. The fundamental layer also contains concepts for states and transitions.

The cognitive and semiotic concepts are structured according to basic mental processes: perceiving, conceiving, and representing. To relate them through the input/output relationships the concepts of action, actor, and actand are defined. In this the aim is to emphasize the human nature of processes and their outputs (perceptions, conceptions, representations). The language-related concepts are defined based on the notion of representation.

C. BWW Model

The BWW (Bunge – Wand – Weber) model has been described and applied in several studies (e.g. Wand 1988a; Wand 1988b; Wand *et al.* 1989; Wand *et al.* 1990a; Wand *et al.* 1990b; Paulson *et al.* 1992; Wand *et al.* 1993; Parsons *et al.* 1993; Wand *et al.* 1995a; Wand *et al.* 1995b; Wand 1996; Green 1997; Weber *et al.* 1996; Weber 1997; Wand *et al.* 1999; Green *et al.* 2000; Parsons *et al.* 2000; Opdahl et al. 2001a; Soffer *et al.* 2001; Rosemann *et al.* 2002). During more than ten years the terminology in, and the contents of, the BWW model have, to some extent, evolved. The model is also rather extensive. Thus, to make a condensed and coherent description of the model is very challenging. Although there are some BNF representations of the BWW constructs (e.g. Wand *et al.* 1993; Weber 1997), it is not easy to transform them into the form of a meta model. Rosemann *et al.* (2002) present a meta model of some of the most essential BWW constructs in the form of an extended ER model, called eERM (Scheer 1998), but the meta model covers only a part of those concepts that are relevant to this study. Nevertheless, we make here an attempt to present a short overview of the model and describe it in a meta model. We base our presentation on Wand *et al.* (1993), Wand *et al.* (1995b), Green *et al.* (2000), and Rosemann *et al.* (2002). First, we present the definitions of those concepts in the representational model (i.e. one of the three models in the BWW model), which are embraced by the scope of the core ontology[73]. We also describe the selected part of the model in the meta model (see Figure 29). At the end of the section we present general remarks of the described model.

The *world* is made up of *things* that possess *properties*. There are *concrete* things, which are called substantial individuals or entities, and *conceptual* things (e.g. mathematical concepts). Domain modelling is based upon someone's view of the existing or possible reality. The notion of a concrete thing applies to anything perceived as a specific object by someone, whether it exists in the physical reality or only in someone's *mind.*

There are no things without properties. Moreover, properties are always attached to things. Properties of concrete things are called *substantial* properties. Properties of conceptual things are called *formal* properties, attributes or predicates. There are two kinds of properties: *intrinsic* properties that depend

[73] Due to the selected scope we have excluded e.g. system-related concepts of the representational model from our consideration.

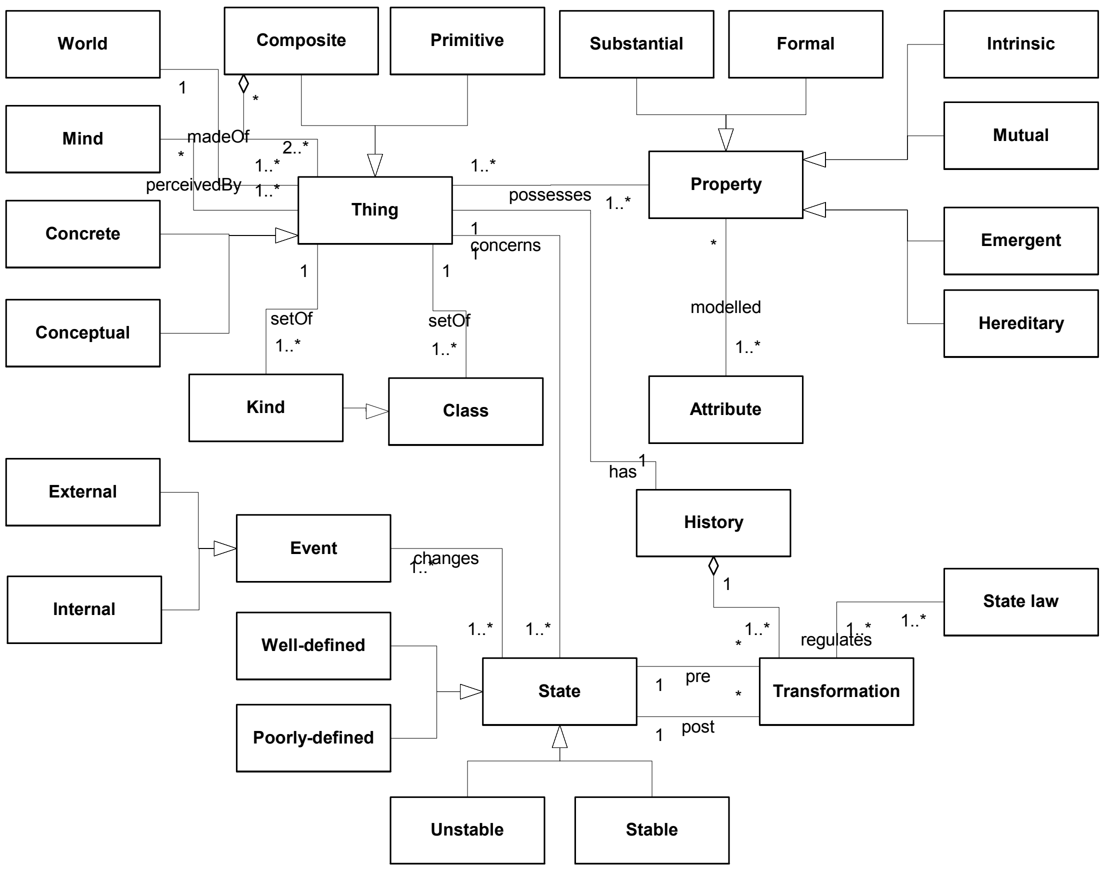


FIGURE 29 Meta model of the relevant part of the BWW model

on one thing only (e.g. the Height of a Person), and *mutual* or relational properties that depend on two or more things (e.g. being a University_Student). Properties themselves cannot have properties. The properties of a thing include laws, which are constraints on other properties and can specify relationships among properties (e.g. a "law" connecting Rank and Salary of an Employee). A *class* is a set of things that can be defined only via their possessing a single property. A sub-class is a set of things that can be defined via their possessing the set of properties in a class plus an additional set of properties. A *kind* is a set of things that can be defined only via their possessing two or more properties.

Things can associate to form *composite* things, whether of composite or primitive things. If things associate to form a thing, then each of the former is a *component* of the latter. A property of a composite thing can be *inherited* (i.e. a property of a component) or be *emergent* (i.e. a property of none of the components). A property that belongs to a component thing is called a *hereditary* property. A composite thing must possess emergent properties.

The properties of a thing exist, whether or not humans are aware of them. Human conceive of things, however, in terms of models of things. Such models are conceptual things. Properties of conceptual things are termed attributes. *Attributes* are characteristics assigned to (models of) things according to human perceptions. For example, people attribute Color to a thing while the real property is the reflection of some wavelength. Every property can be modelled as an attribute but not every property will be described in terms of attributes. The same property might be represented by more than one attribute, and several properties might map into one attribute (for example, the notion of IQ reflects many properties of a human being). Not every attribute has to represent a property (e.g. the Name of a Person).

An attribute is represented as a function from a set of things and a set of 'observation points' (in particular time) into a set of values. This is the basis for defining a model of a thing as a functional schema, which is a set of attribute functions defined over a certain domain. The *state* of a thing comprises the values of the functions in the functional schema at a given time point in a certain domain. Because of the laws, called *state laws*, not all possible combinations of state function values represent valid states of a thing. A lawful state space is the set of states of a thing that comply with the state laws of the thing.

Every change is tied to things and every thing changes. When a thing changes, some of its properties must change. An *event* is a change of a state of a thing. An event space is the set of all possible events that can occur in the thing. An *external* event is a change of state due to the actions of other things. An *internal* event is a change in the state of a thing due to a transformation inside the thing. A *well-defined* event is an event in which the subsequent state can always be predicted given that the prior state is known; otherwise an event is a *poorly defined* event. A *transformation* is a mapping from one state to another state. A lawful transformation defines which events in a thing are lawful. An *unstable state* is a state that will be changed into another state by virtue of the action of transformation in the system; otherwise, the state is a *stable state*. It follows from above that a thing can change state from a stable state only due to an external event. The *history* of a thing means the chronologically ordered states that a thing traverses.

Figure 29 presents the meta model of the concepts and relationships of the selected part of the BWW model. We can make the following remarks on the basis of the meta model. First, the most fundamental concepts are thing and property. There is no isA relationship between them. Both the things and the properties are specialized in several ways. Things are either concrete or conceptual, and either composite or primitive. Properties are substantial or formal, emergent or hereditary, and intrinsic or mutual. In the BWW model there is no basic concept referring to relations between things. A relation is regarded as a special kind of property, that is, mutual property. A class and a kind are defined extensionally. A special feature of the BWW model is that it provides two concepts for characterizing "thing", namely property and attribute. An attribute is a "mental" counterpart to a "physical" property. Besides the aforementioned fundamental concepts, the selected part of the BWW model contains concepts related to states and transformations from one state to another. Via these concepts the BWW model extends to provide a large set of system-related concepts. These are excluded from this presentation.

3.10.3.3 Counterparts and Coverage

To have a more concrete view about how the core ontology, the Frisco framework, and the BWW model compare with one another we present concepts organized according to the structure of the core ontology in Table 11. The table also helps to analyze the coverage and emphases of the three presentations. It should be noted that although some concepts are located in the same rows, it does not mean that the meanings of the concepts are exactly the same. The suggestion made here is that the concepts are, to a large extent, comparable. We are not going to discuss differences between the meanings in details (for the definitions see the preceding sections). Those concepts in the Frisco framework and the BWW model, for which there are no counterparts in the core ontology, are presented in the last row in Table 11.

The Frisco framework addresses mainly the subjective reality (through conceptions) but it does also recognize the physical world, although it refers to it in a "vague" way (i.e. domain, "world"). All the concepts specialized from thing concern the subjective reality. The most fundamental concepts (i.e. thing, relationship, predicator, predicated thing) have direct counterparts in our ontology. The notion of a role is not explicitly defined but substituted with the use of predicator. The Frisco framework does not define the concepts 'point of view' and 'framework'. Neither does it recognize the specific semiotic concepts. The only concept in the intension/extension ontology that is defined in the framework is population.

Because we have used the Frisco framework to elaborate our language ontology, it is natural that there are many similarities between these two presentations as regards this part[74]. It is, however, worth of noting that in the Frisco framework the conceptual basis for the language-related concepts is tightly bound to concepts of a human actor and his mental processes. In contrast, we have derived the concepts directly from the semiotic concepts and delayed the introduction of the "contextual" aspects to the next upper level of OntoFrame.

Also for the state transition ontology the concepts of the core ontology and the Frisco framework are similar to one another. The only exception is the notion of an event, which is missing from the Frisco framework. The concepts of the UoD ontology are not included in the Frisco framework either. The abstraction ontology is covered quite superficially in the Frisco framework. The only concepts defined are: type/instance, composite thing/component and set membership. These are definitely not enough when considering the importance abstraction has in mental processes.

In addition to the concepts having counterparts in the core ontology, the Frisco framework defines a number of concepts that are mostly related to mental processes of a human being and their outcomes. We save the notion of

[74] Actually, we established the language ontology before the Frisco framework was published, but we later noticed needs to refine it according to the Frisco framework.

an entity for the specific use in modeling the UoD (see Chapter 6). The notions of rule, actor, action and actand are given counterparts in the context ontology (Chapter 4).

The BWW model recognizes clearly the physical world, due to its objectivist position, and provides several concepts for characterizing and structuring it. A bridge between the physical world and the "mind" is

TABLE 11 Concepts of the core ontology, the Frisco framework, and the BWW model

Core ontology	**Frisco framework**	**BWW model**
Generic ontology		
physical reality	domain, "world"	world
subjective reality		mind
thing	thing	thing
relationship	relationship	mutual property
property	predicator	intrinsic property, attribute
characterized thing	predicated thing	
role		
point of view		
framework		
Semiotic ontology		
sign		
concept		
referent		
Intension/extension ontology		
predicate		
intension		
extension	population	
population		
basic concept		
derived concept		
abstract concept		conceptual thing
concrete concept		
individual concept		
generic concept		
Language ontology		
language	language	
semantics		
abstract syntax		
concrete syntax		
vocabulary	alphabet	
symbol	symbol	
expression	representation, symbolic construct	
label	label	
proper name		
common noun		

(continues)

TABLE 11 (continues)

Core ontology	Frisco framework	BWW model
State transition ontology		
state	state	state
transition	transition	transformation
event		event
transition structure	state-transition structure	
sequence	sequence	
choice	choice	
concurrence	concurrence	
elementary transition	elementary transition	
composite transition	composite transition	
life cycle		
UoD ontology		
Universe of Discourse		
UoD state		
UoD behavior		history
UoD evolution		
Abstraction ontology		
Classification		
instOf		
type	type	class, kind
instance	instance	
Generalization		
isA		
supertype		
subtype		
Composition		
partOf		
whole	composite	composite
part	component	
Grouping		
memberOf	set membership	
group		
member		
Not included in the core ontology	transition occurence, elementary thing, entity, rule, actor, action, actand, human actor, perceiving action, conceiving action, representing action, perception, conception	concrete thing, primitive thing, substantial property, formal property, hereditary property, emergent property, state law, external event, internal event, well-defined event, unstable state, stable state, poorly defined event

constructed via the notion of an attribute that is a characteristic assigned to things according to human perceptions. The model does not contain the notion of a relationship. Instead, it links two or more things together with the notion of a mutual or relational property.

The BWW model does not address the semiotic ontology, the intension/extension ontology nor the language ontology. Instead, it provides a substantial consideration for states and transformations from one state to another. This area is important for the model due to its system-theoretic basis. Only some of the concepts in the UoD ontology and the abstraction ontology are recognized and defined.

Due to the differences between the approaches of the core ontology and the BWW model, the BWW model contains several concepts that are not included in our core ontology. The BWW model provides, for example, rich sub-categorizations for the notions of property, event and state.

We consider it important to distinguish between two realities, the physical reality and the subjective reality. Likewise, we see it necessary to have specific concepts for identifiable beings (thing), their characteristics (property) and relations (relationship). Conceiving a phenomenon in reality as a thing, a property, or a relationship, depends on the selected point of view, which in turn may be determined by a selected framework. Furthermore, it is important to make the semiotics, as a universal theory, visible in the core ontology. Therefore, we have used the semiotic ontology as a focal point from which the specializations of the notion of a concept, on one hand, and the derivation of language-related concepts, on the other hand, have started. The intension/extension ontology contains rich categorizations of concepts and provides a foundation for deriving specific concepts and constructs of the abstraction ontology. In the core ontology abstraction is strongly emphasized for several reasons. First, abstraction is vital to human conceiving and thinking. Second, abstraction, in all its forms, is the major mechanism with which all the concepts in the extensive OntoFrame have been derived.

The language ontology provides concepts that are fundamental for the presentation of concepts and constructs. The state transition ontology gives means to distinguish between static and dynamic phenomena in reality. The UoD ontology is needed to couple together the phenomena that are relevant from an adopted point of view.

3.10.4 Conclusions

To conclude from the comparative analysis, we can state that the Frisco framework is, to a large extent, similar to the core ontology of OntoFrame as regards the generic ontology, the language ontology and the state transition ontology. In contrast, it provides insufficient support for perceiving, understanding, structuring and presenting phenomena that are addressed by the semiotic ontology, the intension/extension ontology, the UoD ontology, and in particular the abstraction ontology. The BWW model differs from the two others as regards its objectivist position. It provides rich concepts for the domains of the generic ontology and the state transition ontology, but much less support for the other domains. The core ontology of OntoFrame provides the most comprehensive support for those domains, which have above been argued to be important. A large number of concepts included in the core ontology have been specialized from one single concept (thing), thus ensuring coherence and consistency, and structured into component ontologies, thus promoting the modularity of the ontology.

3.11 Summary

In this chapter we defined the core ontology to be the ”heart” of OntoFrame. The core ontology is composed of seven component ontologies: the generic ontology, the semiotic ontology, the intension/extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology. For each component ontology, the concepts and relationships were defined and presented in meta models in our ontology representation language. Following a generative approach, the concepts of the core ontology were specialized from those defined in more generic component ontologies. This yielded a modular structure, which helps the interpretation, use and integration of the concepts and constructs. The related work was widely referred and compared with each component ontology. In addition, an extensive comparative analysis of the two most prominent presentations, the Frisco framework and the BWW model, was made to find out their objectives, ontological positions, basic structures, coverage and emphasizes. The analysis showed that the core ontology provides the most comprehensive and modularized set of concepts for perceiving, structuring and presenting fundamentals in reality.

Our goal has been to engineer the core ontology that is as generic as possible. In addition, the ontology should reflect, in a multifaceted fashion, the diversified phenomena in reality, without going too much into details in any specific domains. Starting with these aims, we first applied a very general view according to which reality means just things. With a slightly more specific viewpoint we recognized properties of things and relationships between things. A conception about whether to have a thing, a property or a relationship depends on the selected point of view. Having built the generic foundation (i.e. the generic ontology), we selected and applied universal theories - semiotics, linguistics and systems theory. Based on these theories we can trust that the corresponding component ontologies remain sufficiently general. To achieve more specific concepts in the essential domains, we finally applied the notions of intension, extension, UoD, and abstraction to derive the rest of the component ontologies in the core ontology.

As far as we know, this kind of approach is quite rare in the literature. The Frisco Group (Falkenberg *et al.* 1998) did use a generative approach to derive specific concepts from elementary concepts. Its outcomes cover, however, only a part of the scope of our core ontology. In the other prominent presentation, the representational model, as a part of the BWW model (Wand *et al.* 1989, Wand *et al.* 1990a), has been engineered based on Bunge's ontology, in which some kind of generative approach has also been applied. The representational model is, however, quite "unstructured" and ignores many of the essential aspects of reality. Although the Frisco framework and the BWW model are two of the most notable presentations, they have not been compared earlier to this degree. Hence, this chapter also contributes to knowledge about similarities in and differences between them.

4 CONTEXT ONTOLOGY

In Chapter 3 we defined the core ontology, which provides the most fundamental concepts, in a modular structure, for perceiving reality. In the core ontology, individual things are related to other things through basic viewpoints and frameworks, thus establishing conceptual constructs with which one can make sense of phenomena in reality. In the semiotic ontology, for instance, things are related to one another through the semiotic relationships. In the state transition ontology things are conceived as states, transitions between states, or events triggering transitions. The points of view and frameworks applied in the core ontology are, however, too simple to provide an adequate support for the understanding of complex phenomena related to human and organizational actions. We need a new, more sophisticated approach and framework. In this chapter we define a contextual approach and an ontology, called the context ontology, which applies the contextual approach.

The general objective of the context ontology is to provide concepts and constructs, which help us understand the nature, purposes and meanings of individual things. The basic idea in the ontology is to aid the conceiving of an individual thing as a part of a whole, called a context. Thus, instead of viewing the UoD as a generic structure of things, the context ontology guides us to see things in special roles or meanings in a context. The context ontology is one of four contextual ontologies, the other ontologies being the layer ontology, the perspective ontology, and the model level ontology (Figure 30). All these ontologies have been derived from the core ontology. The three other ontologies will be presented in the next chapters.

This chapter is organized as follows. First, we define the contextual approach, characterize its application domain and objectives, and establish the theoretical basis for the approach. After a comprehensive literature review and discussion, the contextual approach is anchored on the underlying theories, including semantics, pragmatics, and theories of human and social action. Second, we elaborate the notion of a context and its conceptual structure.

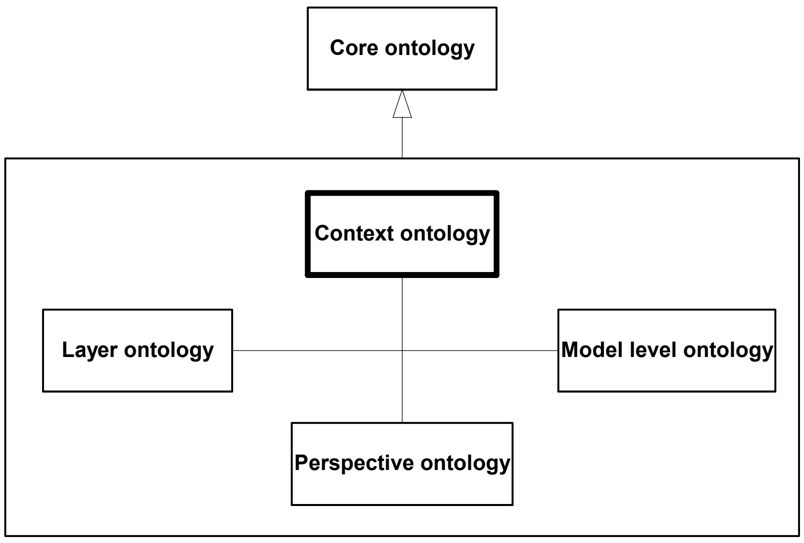


FIGURE 30 Focus of Chapter 4

A context is defined as a construct that is composed of concepts within seven contextual domains. Third, we define key concepts and relationships for all the seven contextual domains and present them in meta models. We also make plenty of references and comparisons to the relevant works. Fourth, we define a large set of inter-domain relationships and consider how implicit relationships between the contextual domains and between the contexts can be inferred from the basic relationships. The chapter ends with a summary and discussions.

4.1 Contextual Approach

In defining component ontologies on lower levels of OntoFrame our aim is to find approaches and sets of concepts that enable us to capture more specific aspects and meanings of things in reality. In this chapter our point of departure is the notion of a context and its potential to support our aim. A context seems to be a suitable notion for many reasons. First, it is a highly universal concept that is known and applied in a large number of disciplines, raging from formal logic and computational linguistic to organizational theory and information systems. Second, it is a common term also in the normal speech. Third, the most common aim of the use of context in various disciplines is to consider a focal thing or an event of interest as a part of the environment (context) in order to understand its nature and meaning (Duranti *et al.* 1992). That is precisely what we wish to achieve with our ontology.

In this section we define the approach, referred to as the contextual approach that is based on the notion of a context. This approach is of high importance, not only to the context ontology, by also to OntoFrame as a whole. Therefore, we want to make a serious attempt to define the approach in a way, which firmly anchors it on the relevant theories and provides an advantageous baseline to establishing the context ontology.

4.1.1 Definition

We start with considering the notion of an approach. Generally speaking, an *approach* is seem something that provides generalized principles, which help us conceive reality, recognize problems and/or find potential solutions in it. Implied from the above, we can distinguish between the conception-oriented approaches, the problem-space oriented approaches, and the solution-space oriented approaches. These approaches are interrelated: the problem-space oriented approaches are based on conception-oriented approaches, and the solution-space oriented approaches are based on the application of some problem-space oriented approaches. Metaphorically put, we can say that an approach sets a path or a road along which goals set for conceiving or problem solving can be approached.

The contextual approach is a conception-oriented approach. A general definition of the contextual approach goes as follows: the *contextual approach* is a conception-oriented approach, which serves the recognition, understanding and specification of the purposes, meanings, and effects of things, through considering them to be contexts and/or parts within contexts. To elaborate the definition, the following issues need to be addressed: (a) an application domain at which the contextual approach is aimed, (b) objectives of the approach, (c) theories underlying the approach, and (d) conceptual contents of the approach. In addition, (e) some examples of outcomes from applying the approach should be provided.

In defining the contextual approach we proceed in a top-down fashion. First, we characterize the application domain (Section 4.1.2) and specify the objectives for the contextual approach (Section 4.1.3). Second, we establish the theoretical foundation for the approach (Section 4.2) from which the content of the approach is then derived (Section 4.3). Third, we elaborate the contents of the context ontology by explicitly defining the contextual concepts and constructs (Sections 4.4-4.6). The definitions form direct outcomes of applying the contextual approach. More evidence of the applicability of the approach will be provided in the following chapters.

4.1.2 Application Domain

The goal of our study is to develop a conceptual foundation for the analysis, design, and implementation of information systems development methods. This goal addresses a number of sub-domains such as information systems, information systems development, and method engineering. These sub-domains are parts of the application domain of the contextual approach. These all embody, to a high degree, human and social action. In the following we characterize the application domain on two levels, first as a generic human and social action, and then more specifically as the analysis, design, and implementation of ISD methods. Human and social action has been subject to a wide array of research in several fields, e.g. in anthropology, philosophy, psychology, linguistics, organizational theories, sociology, economics, politics, law, statistics, library science, cybernetics, systems theory, computer science, and information systems. Here, we content ourselves only with common insights and views.

Human and social action can be characterized in terms of six intrinsic properties. The most significant property of a human being is his/her ability to use a semantically and pragmatically rich language. It enables one to observe, conceive, understand, and describe relevant features of reality. Second, a human being is able to analytically, heuristically or intuitively make conclusions, based on his/her observations, in order to promote his/her understanding and management of the environment. Third, a human being is, to a high degree, oriented towards the goals emerging from the needs of his/her own, on one hand, and from the expectations of the environment, on the other hand. Fourth, a human being is able to learn in several areas including language usage, sensations, norms, etc. This, together with his/her orientation to goals, facilitates the application, construction and reconstruction of norms and rules. Fifth, human beings make their most prominent achievements in cooperation with others, under the goals, norms and values they have jointly established. A prerequisite for cooperation is a common language, understood and cultivated by all members in the society. Sixth, a great deal of human and social action is primarily concerned with information. Information is essential in rendering human and social actions effective.

The view of the application domain of the contextual approach becomes more concrete when relating to information processing. The application domain comprises people developing and applying rules, norms and facilities to collect, process, store, distribute, and utilize information in order to make actions of the others and/or of their own possible or more effective (cf. Nissen 1980). This kind of human and social action is visible at several processing layers. At the root layer (IS), information is collected, processed, transmitted and utilized in order to make “right” decisions on business actions. At the next layer (ISD), information is collected, processed, transmitted and utilized in order to make “right” decisions on designing and implementing IS’s. At the third layer (ME), information is collected, processed, transmitted and utilized in order to make “right” selections and adjustments of ISD methods. Finally, at the highest layer (RW), information is collected, processed, transmitted and utilized in order to engineer ME methods. For all four layers it is important that the nature and meaning of information processed and of actions performed are adequately understood.

4.1.3 Objectives

The contextual approach is a conception-oriented approach, which aims at the following objectives:

1. *The approach should be natural.*
   The approach should reflect an intuitive view, held by a human being, on the structure and aspects of reality.
2. *The approach should be easy and flexible to apply.*
   The approach should be easy to learn and use, and flexible by its conceptual structure, in order to enhance its applicability for different purposes.
3. *The approach should have a sound theoretical basis.*
   The approach should be based on sound theories on information processing as well as on human and social action.
4. *The approach should cover essential aspects of a context.*
   The approach should guide us to conceive, structure and represent the most essential aspects of the contexts.
5. *The approach should help perceive the nature, meaning and effect of individual things in a totality.*
   This means, for instance, that the meaning of data can only be understood if knowledge is available of those situations in which the data is created, processed and/or utilized. Likewise, it is hard to understand the essence of action if the action is not connected, at least, to its objectives, to objects that are involved, and to subjects conducting the action (cf. Kuutti 1991). It seems equally evident that the ultimate intentions of persons can only be revealed by observing their behavior in real situations.

The contextual approach is not intended for the studies of specific features or structures of facilities, persons or any other elementary things alone. Instead, its purpose is to bring the things together and provide a framework to interrelate them, in order to help us find out their purposes and meanings.

4.2 Theoretical Basis

The purpose of this section is to establish the theoretical basis for the notion of context. We approach this along three paths: (a) we make a short survey of the literature to find out those disciplines for which context is an essential research issue, (b) we make a review of those theories that pursue the same contextual objectives as we do, and (c) we search for studies which, although lacking any theory, work with concepts which can be considered to be essential parts of a context. After that we are ready to make a synthesis and complete the definitions of the context and the contextual approach in Section 4.3.

4.2.1 On the Notion of a Context

The notion of a context[75] plays an important role in many disciplines, such as in formal logic (e.g. Costa 1999), knowledge representation and reasoning (Brezillon *et al.* 1998; Sowa, 2000; Ghidini *et al.* 1999), machine learning (Matwin *et al.* 1996), pragmatics (Levinson 1983), computational linguistics (Clark *et al.* 1981; Berthouzoz 1999), sociological linguistic (Halliday 1978), problem solving (Motschnig-Pitrik *et al.* 2001), organisational theory (Weick 1995), sociology (Layder 1993), neurology, cognitive psychology (Kokinov 1999), and information systems (Kyng *et al.* 1997). Due to the large variety of research fields, it is no surprise that there is no general agreement on a unique, shared notion of a context. Rather the interpretation of the notion itself is context-sensitive. Linguists may see it as a psychological construct, a subset of a hearer's assumptions about the world that is used in interpreting an utterance. To an organisational theorist context is a social environment in which actions are taken. To a sociologist, a context is provided by macro-social forms, such as gender, national ethos, and economic maturity of a society (Berztiss 1999). Even within the same discipline, conceptions of context may vary a lot. In philosophy, for instance, it is questioned, whether context is internal, part of the state of the mind, or external, part of the state of reality. Is it explicitly represented in the human mind or just implicitly (Kokinov 1999)? Because of this large variety of interpretations, it would be better to speak of "family-resemblance" concept (Penco 1999, 269).

Some of the confusion results from an ambiguity in the English word 'context'. According to Webster's Encyclopedic Unabridged Dictionary (Webster 1989), context means (1) "the parts of a written or spoken statement that precede or follow a specific word or passage, usually influencing its meaning or effect", and (2) "the set of circumstances or facts that surround a particular event, situation, etc.". Hence, the notion of a context can refer to the text, to the information contained in the text, to the thing that the information is about, or the possible uses of the text, the information, or the thing itself (cf. Sowa 2000). The ambiguity results from which of these aspects happen to be the central focus.

In spite of inconsistencies and diverging interpretations, a context is of vital importance in many disciplines, especially in that sense in which we are interested about it, that is to say in revealing meanings of things. As Klemke (1999, 481) states, psychologist perform memory test to analyze the effect of a context on the remembrance of words (Srinivas 1997). Researchers from machine learning study the effects of a context on the automatic learning of concepts (Matwin *et al.* 1996). Organizational research people use communication models to investigate the role of a context in information product evaluation (Murphy 1996) and cognitive scientists stress the importance of a context for expertise (Raccah 1997). In artificial intelligence, the

[75] In Latin *contexere* means weaving or joining together.

notion of a context was introduced in a logic framework by McCarthy as early as in 1971 in his Turing Award talk (Massacci 1996).

Due to its importance and universality, we adopt a context as the most essential concept of our contextual framework. Also, the way a context is used as the environment through which the meanings of its parts are explained encourages us to the use of the concept (Duranti *et al.* 1992). A context is characterized with the following generic features: (a) It is determined and shaped by one or more focal parts of which making sense is important. (b) It is composed of parts, which all have specific roles of their own. (c) It is a totality in which each part gets its meaning through the position it holds in the whole and through the relationships it has with the other parts. In the next sections, our aim is to elaborate these general characterizations into the precise definitions of the context and the contextual approach.

4.2.2 Relevant Theories

In this section we make a review of those basic theories that pursue the same contextual objectives as we do. From the viewpoint of our study, the most promising theories are those, which have their focus on human and social action, and for which features of information and data processing are essential. In general, the semiotics as a theory of signs meets these requirements. Peirce (1991) divided semiotics into three branches: syntactics, semantics, and pragmatics[76]. Semiotics itself is, however, too general and superficial in its treatment of contextual aspects. Stamper (1973, 1996) has extended the set of three semiotic branches into the semiotic ladder, which builds a bridge between the physical world and the social word and helps to distinguish between the different conceptions of meaning. The steps in the ladder are: physical world, empirics, syntactics, semantics, pragmatics, and social world. In Table 12 for each step, essential things and conceptions of meanings are presented (cf. Falkenberg *et al.* 1998, 140-145).

Elements in the physical world are physical signals and marks. They have almost nothing to do with meaning. In the empirics, the considerations are focused on a process of encoding of messages, transmission of them over a channel, disrupted by noise and entropy, and finally of decoding messages back. "Meaning" in this environment stands for the equivalence of codes, and so the view of meaning is highly technical and narrow. Syntactic "meaning" can be manifested through transformations of symbolic forms to other forms according to the rules. Not until on the semantic step of the semiotic ladder are meanings discussed in the real sense. As it is known, there are several theories of semantic meaning (Alston 1980; Lyytinen 1985; Holm *et al.* 1995). Most of them maintain a view of a dictionary-like or universal meaning of words that ignores effects an individual interpreter has on the meaning. More close to the subjective meaning can be reached in pragmatics, which considers a

[76] The words syntactics (or syntax), semantics, and pragmatics were introduced by Charles Morris (1964) in his presentation of Peirce's three branches of semiotics.

TABLE 12 Steps in the semiotic ladder

Steps of the ladder	**Things**	**Meaning**
Physical world	signals, marks, traces,…	equivalence of signals
Empirics	codes, entropy, channel,…	equivalence of codes
Syntactics	signs, formal structures, production rules,…	through transformation rules
Semantics	meaning, objective reality,...	mapping from syntactic structures onto features of real world[77]
Pragmatics	intentions, communication, speakers, utterances, contexts	defined in terms of the actual social consequences which stem from communication
Social world	beliefs, expectations, norms, attitudes, commitments	defined in terms of social norms involved

relationship between the signs as meaningful utterance and the behaviour of responsible agents[78]. This view interlinks information, expression, message, or whatever, to a context in which that something is expressed and interpreted by someone, at the concrete time and place. The most in-depth consideration of meaning is possible when conducted in conjunction with the social world. The social world embraces norms of many kinds, ways of behaving, sets of values, shared models of reality, common attitudes, organisational cultures, etc.

As said above, we are interested in theories which focus on information and its meaning, as well as on the understanding of human and social action. Thus our concern is related to the upper part of the semiotic ladder, i.e., semantics, pragmatics and social world. Compared to the meanings Webster Dictionary provides for a context, we can see that the first two (i.e. semantics and pragmatics) are more or less related to the first conception, and the social world is related to the second conception in Webster (1989).

Next, we shortly describe theories of semantics, pragmatics and social world with the aim to recognize their essential principles and concepts that could elaborate our conceptions of the contextual approach and the context. We also describe some approaches based on the theories.

77 Note that this view is maintained in the objectivistic position. Another view, favoured by the constructivist position, considers meaning as being "constructed and continuously tested and repaired through people using syntactic structures to organise their co-ordinated actions" (Falkenberg *et al.* 1998, 143).

78 The semantics-pragmatics distinction is commonly discussed in linguistics and in the philosophy of language. However, it is not altogether clear; see, for example, the discussion in Levinson (1983, Chapter 1). For present purposes, we say that semantics is concerned with context-free meaning, while pragmatics is concerned with context-dependent meaning.

4.2.3 Semantics

Semantics is the study of meaning. Due to a variety of semantic theories, there is a spectrum of "meanings" of meaning in the linguistic literature. Alston (1980) classifies the theories of meaning into three groups, which he calls referential, ideational and behaviour theories. Lyytinen (1985) defines five "language views" that have resulted in different conceptions of meanings. The views are: the Fregean core, the Chomskyan grammar, Piage's schema, the Skinnerian response, and Ordinary speaking. Holm *et al.* (1995) distinguish between the referential theory, the 'traditional' theory, the ideational theory, the stimulus-response theory, and meaning in use. Generally speaking, meaning can be considered as a function from signs to reality (Stamper 1992b). We apply here the classification of semantic principles presented by Stamper (1996). Stamper distinguishes between the objectivist principle, the mentalistic principle, and the constructivist principle.

According to the *objectivist* meaning, an observer 'sees' the objective connection between the sign and the referent that is independent of the observer. There is only one reality, independently of any observer and interpreter. Meaning is what a sign refers to, or stands for (Lyons 1981), in other words, meaning is the relation between the sign and what is referred to. According to the *mentalistic* meaning, there is no one (physical) reality, because reality is conceived and perceived solely by the senses of human beings. Meaning of the sign is the idea or concept, associated with it in the mind of anyone who knows it (Lyons 1981). According to the *constructivist* meaning, a community establishes and alters the relationship between the sign and the referent. There is one reality but that is perceived and conceived differently. The meaning of the sign is the way it is used. Of the three semantic principles, first two clearly belong to semantics, while a major part of the constructivist principle concerns the issues that we regard as belonging to pragmatics. This is how we deal with them below.

Next, we consider two semantic approaches based on the objectivist principle. These are: case grammar and conceptual graphs. In both of them, one can see the purpose "to create a mapping from natural language sentences into a formal specification of a grammar" (cf. the Chomskyan grammar view in Lyytinen (1985)) and "to create a formal language such that every fact in the world corresponds to a structure in the formal language" (cf. the Fregean core view in Lyytinen (1985)).

Case Grammar

Fillmore (1968) introduced a case grammar as a representation of a universal semantics of a natural language. He distinguishes between the surface level cases corresponding to grammatical functions and the deep level cases (called semantic cases) corresponding to underlying roles that the sentence participants play with respect to the main verb. The case grammar is founded on a set of semantic cases and a set of rules, which can provide the semantics for simple sentences describing actions.

According to Fillmore (1968) "the case notions comprise a set of universal, presumably innate, concepts which identify certain types of judgments human beings are capable of making about the events which are going on around them, judgments about such matters as who did, who it happened to, and what got changed"(ibid p. 24). The sentence in its basic structure consists of a verb and one or more noun phrases, each associated with the verb in a particular case relationship. The notion of case is a language element that is more stable than surface-level grammatical terms such as subject and object. For this reason, the case grammar is sometimes said to reveal the "deep structure" of a sentence. Fillmore (1968, 24-25) defines six cases:

- *Agentive*. The case of the typically animate perceived instigator of the action identified by the verb.
- *Instrumental*. The case of the inanimate force or object causally involved in the action or state identified by the verb.
- *Dative*. The case of the animate being affected by the state or action identified by the verb.
- *Factitive*. The case of the object or being resulting from the action of state identified by the verb, or understood as a part of the meaning of the verb.
- *Locative*. The case, which identifies the location or spatial orientation of the state or action identified by the verb.
- *Objective*. The case, which identifies the things, which are affected by the action or state identified by the verb.

In his article, Fillmore (1968) did not preclude a possibility that "additional cases will be needed" (ibid p. 25). So, the set of cases is open ended. None of the cases can be interpreted as matched by the surface-level relations, subject and object, in any particular language. Consequently, the same word can correspond to different cases in different sentences. In the basic structure of sentences, Fillmore distinguishes between what he calls the 'proposition', a tenseless set of relationships involving verbs and nouns, and what he calls the 'modality' constituent. This latter includes such modalities on the sentence-as-a-whole as negation, tense, mood, and aspect.

Fillmore's case grammar has been criticised (e.g. Platt 1971; Nijholt 1988), especially as to the notion of case itself and the criteria on which cases are identified. Likewise, there are many presentations, which suggest different notions of case and different sets of cases (e.g. Schank 1973; Wilks 1977; Simmons 1973; Dik 1989). Nevertheless, the grammar has been widely applied, including in the fields of information systems and computer science. For instance, Rolland and Proix (1992) propose a natural language approach to requirements engineering. The approach adapts the notion of case to make it applicable, not only to words, but also to clauses in sentences. The set of cases includes: owner, owned, actor, target, constrained*, constraint*, localization*, action*, and object, in which * denotes the cases that are applicable to clauses, too. Rolland and Achour (1998) present an approach to guide the construction of use case specifications, which is based on a set of linguistic patterns derived from case grammars by Fillmore (1968), Dik (1989), and Simmons (1973).

Conceptual Graphs

Conceptual graphs form a knowledge representation language based on linguistics, psychology, and philosophy (Sowa 1984, 69). They are an extension of Peirce's existential graphs (Sowa 2000, 476). Besides Peirce's primitives, conceptual graphs provide means of representing case relations, generalized quantifiers, indexicals, and other aspects of natural languages. They were first proposed as a mechanism for representing database semantics (Sowa 1976). Later, the emphasis has been on the AI field, and this is where they have received the most attention.

In a conceptual graph, the boxes are called concepts and the circles are called conceptual relations. Sowa (1984, 2000) presents a set of primitive concepts and relations. The set can be extended with user-defined ones. Examples of concepts are (Sowa 1984): ACT (an event with an animate agent), ARRIVE (a mobile entity arrives at a place), PLACE (a role played by a stationary entity), STATE (has duration, as opposed to events), and TOOL (an entity that plays the role of an instrument for some act). Examples of relations are (Sowa 1984): agent (links an actor to an act), destination (links an act to an entity towards which the act is directed), instrument (links a tool to an act), location (links something to a place), object (links an act to an entity, which is acted upon), and point-in-time (links something to a time at which that something occurs). In addition, Sowa (2000) provides the concept of context. It stands for a nested conceptual graph that describes the referent.

Conceptual graphs are widely applied, also in the IS field. To give just some examples, Bezivin *et al.* (2001) use the conceptual graphs to clarify metamodeling layers and OMG/MDA architecture. Kayed *et al.* (2002) extract ontological concepts for tendering conceptual structures. de Moor *et al.* (2001) apply conceptual graphs in workflows. Moulin and Creasy (1992) extend the conceptual graph approach for data conceptual modelling.

4.2.4 Pragmatics

The modern usage of the term 'pragmatics' is attributable to the philosopher Charles Morris (1964) who distinguished three branches of inquiry within the semiotics: syntactics, semantics, and pragmatics. His conception about the pragmatics was very broad, covering psychological, biological and sociological phenomena, which occur in the functioning of signs. Thus, it would include disciplines which are now known as psycholinguistics, sociolinguistics, neurolinguistics and many more besides. Since Morris' introduction, the term has been treated and defined in many ways (see Levinson 1983). Our intention here is first to generally characterize the scope of pragmatics, and then bring up those issues in pragmatics which could benefit us in elaborating the contextual approach. In addition, we select one theory, the speech act theory, for considerations in more detail.

Generally speaking, pragmatics is the study of language use. Assuming that semantics is concerned with the statements of truth conditions, we can say that pragmatics deals with all those aspects of meaning that are not captured in a truth-functional semantics. One of the most favoured definitions in the linguistic literature is as follows: "pragmatics is the study of the ability of language users to pair sentences with the contexts in which they would be appropriate" (Levinson 1983, 23). This definition forms a first indicator of the significance of context to the pragmatics.

Instead of trying to define analytically the concept of pragmatics, we can enumerate issues that are addressed in pragmatics. According to Levinson (1983, 27), pragmatics is the study of deixis, implicature, presuppositions, speech acts and aspects of discourse structure. There are also other suggestions for the list. Especially the relation to sociolinguistics (e.g. conversational structure) is vague. Nevertheless, deixis is, in most cases, included in pragmatics.

We conclude that pragmatics is the study of the relations between language and context. In such a study one of the most essential issues deals with deixis. Deixis concern the ways in which "languages encode or grammaticalize features of the context of expressions or speech events, and thus also concern ways in which the interpretation of expressions depends on the analysis of that context of expressions" (Levinson 1983, 54). Traditional categories of deixis are person, place and time deixis[79].

Person deixis concerns the encoding of the role of participants in the speech event. The category 'first person' is the grammaticalization of the speaker's reference to herself, 'second person' the encoding of the speaker's reference to one or more addresses, etc. Such participant-roles are encoded in pronouns in a language. Time deixis concerns the encoding of temporal points and periods relative to the time at which an expression was presented. This kind of deixis is grammaticalized in deictic adverbs of time like now, then, yesterday and this year. Place deixis concerns the encoding of spatial locations relative to a location of the participants in the speech event. It is grammaticalized in demonstratives like this and that, and in deictic adverbs of place like here and there.

Above, deixis was mainly considered from the grammatical point of view. Although we are not interested in the ways the deixis are organized in language, their explicit appearance in the expressions makes us convinced that

---

79 Besides the traditional categories considered above, two more deixis aspects have been lately distinguished (Levinson 1983): discource deixis and social deixis. The former concerns the encoding of reference to the portions of the unfolding discourse in which the expression is located, and the latter concerns the encoding of social distinctions that are relative to participant-roles (cf. honorifics). These kinds of deixis aspects go beyond our interests.

they are also conceptually of vital importance. That is to say, the concepts underlying the deixis are good candidates for contextual concepts.

Speech Act Theory

Speech act theory is the systematic study of linguistic regularities and the meaning of utterance. It is concerned with the pragmatic use of language. The speech act theory originated with Austin (1962) and was further elaborated by Searle (Searle 1969; Searle 1979; Searle *et al.* 1985). The basic notion is that people do things with words. Words change the world, rather than merely describe it. Uttering a sentence is the performance of an act, called a speech act.

By uttering a sentence, four concurrent acts occur: propositional act, illocutionary act, utterance act, and perlocutionary act (Searle *et al.* 1985). A propositional act expresses the propositional contents of a message. An illocutionary act is performed when a speaker utters a sentence in an appropriate context with certain intentions. A propositional act always occurs as part of an illocutionary act. An utterance act refers to the simple uttering of a sentence. A perlocutionary act means producing effects on the feelings, attitudes and behaviour of the hearer.

The most important type of speech act, especially for us, is that of the illocutionary act. It is composed of three constituents: context, content, and illocutionary force. All these constituents are essential for constructing and understanding the meaning of a sentence. Content refers to the propositional content of the message. Context is defined in terms of speaker, hearer, time, place, and the possible world. The first four define a context in which a speaker utters something to a hearer sometimes and somewhere. The possible world refers to the residual features of the context. It is something more than the "actual world", and it enables to talk about "what could be". An illocutionary force covers several issues. Here we are interested in the concept of commitment. According to the speech act theory, commitments are created through communication. Depending on types of illocutionary acts, different kinds of commitments can be distinguished: assertives, directives, commissives, declaratives, and expressives.

Along the application of the speech act theory to the IS research the prevailing language perspective changed from a referential one to communication oriented one. The speech act based approaches, recently called language/action approaches, have been explored and elaborated (see e.g. Proceedings of Language/Action Conferences), but also criticized, extensively. Proponents of the approach are, for instance, Auramäki *et al.* (1988) and Auramäki *et al.* (1992a) who suggest methods and principles for analyzing offices as systems of communicative action according to the SAMPO (Speech-Act based office Modeling aPprOach) approach. De Cindio *et al.* (1986) developed a coordinator, called CHAOS, based on the language/action approach. Dietz (1992, 1994, 1999, 2003) presents the DEMO approach and framework to model open active systems and business processes. The approach makes a distinction between subjects, which are the active elements of the system, and objects to be acted upon. Woo *et al.* (1992) present an approach to facilitate the automation of semi-structured and recurring negotiations in organizations. Janson *et al.* (1995) use the speech act theory to compare IS development tools and methods.

The speech act theory has been criticized as a philosophical and linguistic theory as such and due to problems with a rationalistic design of work (Ljungberg *et al.* 1996). Wand *et al.* (1995a, 295) argue that the speech act theory lacks a comprehensive overall picture of how actions fit and relate to each other. They also state that the theory focuses on unidirectional speech act performance and is restricted to spoken discourse. In spite of the criticism, the speech act theory provides views of the meaning and related concepts that are beneficial for our study.

4.2.5 Theories of Human and Social Action

The complexities of organisational practice in information systems put demands on research and knowledge. There are many conceptual approaches aiming to describe and explain these complex phenomena. Some of the approaches emphasize the action concept. According to them, it is difficult, or even impossible, to create good scientific descriptions and explanations about organizational practice without acknowledging actions. Although such approaches may have differing theoretical perspectives, they have a unifying interest in the action concept and its explanatory power. Examples of such approaches are activity theory (Engeström 1987, 1999), actor-network theory (Latour 1999; Monteiro 2000), and structuration theory (Giddens 1984). There are also other approaches, which have theoretical influences of more or less explicit action orientation, such as social phenomenology, symbolic interactionism, ethnomethodology, soft systems theory, critical social theory, hermeneutics, social semiotics, socio-pragmatism, situated cognition theory, practice theory and affordance theory. Along with this interest in action comes also an interest in many related issues, such as knowledge, language, communication, social interaction, social institutions, coordination, artefacts, power and values.

A prominent representative of approaches pertaining to social and organizational issues is the activity theory. The *activity theory* presents highly general propositions of the nature of human activity incorporating several psychological, educational, cultural and developmental approaches (Leont'ev 1978; Vygotsky 1978). It is a philosophical framework for studying different forms of human praxis as developmental processes, at both individual and social levels interlinked at the same time (cf. Kuutti 1991, 530). According to the theory, there exists a fundamental type of context, called an activity. Activity is a minimal meaningful context for individual actions. Relations within an activity are direct ones but are mediated by various artifacts such as instruments, signs, procedures, machines, methods, laws, forms of work organization, etc. These artifacts have been created and transformed by humans during the development of the activity itself (Kuutti 1991, 531).

Based on the theory, Engeström (1987, 1999) developed an applicable model of the systemic structure of human activity. The fundamental concepts of the model are (see Figure 31): subject, object, tool (mediating artifact), rules, community, division of labor, as well as outcome from the activity. The concepts are interrelated in terms of mediating: the relationship between subject and object is mediated by tools, the relationship between subject and community is mediated by rules, and the relationship between object and community is mediated by the division of labor. An activity consists of actions or chains of actions, which in turn consist of operations. Actions are related to goals.

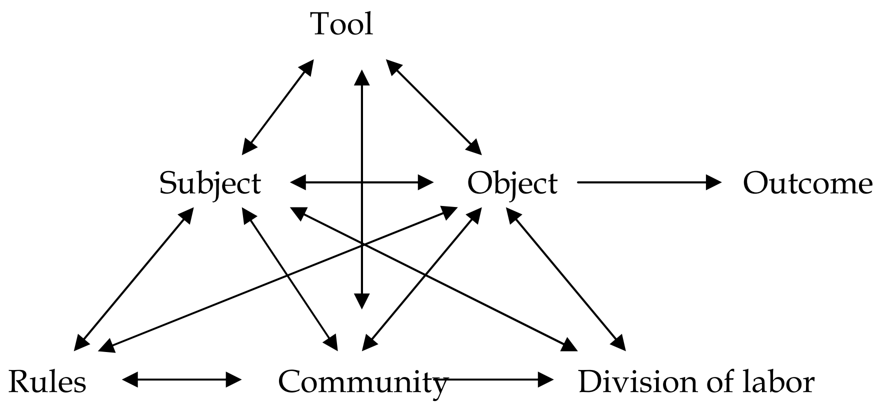


FIGURE 31 Engeström's activity model (Engeström 1999)

The activity theory has been applied and elaborated in various sub-fields of information systems, e.g. in user interface design (e.g. Bodger 1987; Nardi 1996), information systems development (e.g. Korpela *et al.* 2000; Korpela *et al.* 2002), computer-supported collaborative work (e.g. Kuutti 1991; Kuutti 1994; Bardram 1998), product concept design (e.g. Tuikka 2002), knowledge management (e.g. Boer *et al.* 2002), and method engineering (e.g. Kaasboll *et al.* 1996).

4.2.6 Context-Related Approaches

Context is used as a key concept also in other fields, although without any explicitly defined context-based theories. Here we make a short review of the studies which aim to specify and understand the meaning of things through the notion of a context.

In databases, Motschnig-Pitrik (1999, 2000) proposes a context mechanism for object-oriented database languages, extending the functionality of views. Individual views are interpreted as context for drawing entities or objects visible from these views and intentionally neglecting others. Contexts are associated with relativized naming, authorization, and channels for change propagation. Contexts are objects in their own right, and they can be classified, associated with properties, enclosed in contexts and dealt with like ordinary objects.

Enterprise modeling aims to capture essential enterprise knowledge, in order to provide a clear, unambiguous picture of how the enterprise functions currently and what are the requirements and the reasons for change (Loucopoulos *et al.* 1998; Loucopoulos 2000; Kirikova 2000). An enterprise knowledge model can serve as a means of understanding and communication between the different participants. The model covers e.g. goals, actors, activities, physical and informational objects, business rules, and their interrelations within a totality, which could be called a context.

Workflow means "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (Workflow Management Coalition 1999, 8). Workflow management system (WfMS) is a system that "defines, creates and manages the execution of workflows through the use of software.." (ibid p. 9). To analyze, design and compare WfMS's, metamodels (e.g. Rosemann *et al.* 1997; Lei *et al.* 1997) and reference models (zur Muhlen 1999) have been engineered. Based on them, workflow systems address persons, organizations, positions, roles, tasks, resources, objects, and their relationships in a complex manner (Mentzas *et al.* 2001; Chiu *et al.* 1999).

User modeling has produced new theories and methods to analyze and model computer users in short and long-term interactions. A user model is an explicit representation of properties of individual users or user classes. It allows the system to adapt its performance to user needs and preferences. Methods for personalizing human-computer interaction based on user models have been successfully developed, applied and evaluated in a number of domains, such as information filtering, e-commerce, adaptive natural language and hypermedia presentation, and tutoring systems. Recently user modelling has also addressed problems of personalized interaction in mobile, ubiquitous and context-aware computing (e.g. Abecker *et al.* 2000). A user context can be composed of an environment context, a personal context, a task context, a social context, and a spatiotemporal context (Myrhaug 2001)

Process modeling aims at systematic analysis, design and implementation of processes. Some of the approaches look at processes through the notion of a context. In the NATURE project (Rolland *et al.* 1995; NATURE Team 1996) a process theory for modeling and engineering the requirements engineering process was crafted. According to the approach, a requirements engineer is commonly in a situation where his/her reaction depends on both the situation and the intention he/she has in mind, that is to say, it depends on the context he/she is placed in. A context is the association of a given situation and the decision, which can be taken on it. A situation is defined as being a part of the product it makes sense to make decision on. There are three kinds of contexts: executable contexts, choice contexts, and plan contexts. At the most detailed level, the execution of the requirements engineering process can be seen as a set of transformations performed on the product under development, each transformation resulting from the execution of a deterministic action. The process theory pertaining to NATURE has been later applied and elaborated in many studies (e.g. Rolland *et al.* 1996; Grosz *et al.* 1997; Rolland *et al.* 1999; Rolland *et al.* 2000).

For information systems architectures Zachman (1987) introduced a framework that was later extended by Sowa and Zachman (1992). It contains two dimensions, views and aspects, and is described by a matrix with rows and columns. The views are determined through the perspectives of different stakeholders. The stakeholders having particular views on an information system are a planner, an owner, a designer, a builder, and a component sub-contractor. The aspects of an information system are data, function, network, people, time, and motivation, thus realizing the set of interrogatives presented originally by Zachman (1987): "Why", "Who", "What", "Where", "When" and "How"[80]. In the framework, presented in the form of matrix, essential concepts and conceptual relations are named for each of 30 cells. The framework has been applied and refined in many studies. Short (1991) has used the framework with six aspects to propose a framework for a classification of ISD methods. Evernden (1996) has extended the framework into the Information FrameWork that is aimed at managing information. Martin *et al.* (2000) have formalized the structure of the framework based on the notion of a frame. Garner *et al.* (1999) have enhanced the framework to elicit and manage the complexity of IS requirements.

4.2.7 Elicitation of Contextual Domains

In the previous sections, we have reviewed theories and approaches that share our aim to promote the understanding of the meaning of things in a whole. The theories and approaches for the review were selected from the topmost steps of the semiotic ladder (i.e. semantics, pragmatics and social world), on one hand, and from a large set of literature, which, yet lacking in-depth theories, aim at "contextual thinking". To conclude from the review we argue that the basic contextual domains are related to purpose, actor, action, object, facility, location, and time. The arrangement of these domains into a context can be illustrated by the following contextual view, also called the seven S's view:

> *"For Some purpose, Somebody does Something for Someone with Some means, Sometimes and Somewhere"*

Next, we justify our argument by summarizing the review. For the summary we select seven major theories or approaches from which we extract the most essential concepts and locate them in those contextual domains to which they primarily belong. The summary is presented in Table 13.

80 There is a large set of studies, which also use the same set of interrogatives ('5Ws and H') for various purposes; e.g. Bull (1989), Curtis *et al.* (1992), Zultner (1993), Couger *et al.* (1993), Krogstie *et al.* (1996), Fitzgerald *et al.* (1998), Söderström *et al.* (2002), and Zhu (2002).

In the case grammar by Fillmore (1968), we can identify the following correspondences between the cases and the contextual domains. An action is expressed by a main verb, an actor corresponds to agentive (i.e. an animate instigator of the action), and an object stands for objective (i.e. a thing affected by the action). Further, a location means locative (i.e. the location or spatial orientation). The cases of dative and factitive are not so clear. We associate dative to the object domain because it is defined as an animate affected by the action. Because factitive can be understood as part of the meaning of the verb (Fillmore 1968), we regard it as part of the object domain. In addition to the named cases, the case grammar contains the modality constituents that reflect time (tense) and purpose (mood or aspect).

In pragmatics (Levinson 1983) several obvious counterparts to the contextual domains can be found. Person deixis, place deixis and time deixis correspond to actor, location and time, respectively. A verb mostly expresses action in the context. In addition, there are other essential concepts that can be classified into the domains: e.g. intention (purpose), speaker and hearer (actor), performative act (action), utterance (object), as well as coding and receiving time (time) which can be seen as specialized concepts, each having been derived from one of the seven generic contextual domains.

In speech act theory (Searle 1969; Searle 1979; Searle *et al.* 1985), illocution within an illocutionary act corresponds to purpose. A speaker and a hearer are actors who are concerned with speech acts (action). The concept of context in speech act theory is composed of time, place and the possible world. The first two have their counterparts among the contextual domains.

In activity theory (Engeström 1987) an activity is regarded as the fundamental type of context. Individual actions and operations get their meanings within an activity. Subject refers to an individual or a group, whose agency is chosen as the point of view. Object refers to the raw material or problem space at which the activity is directed and which is molded and transformed into an outcome. Tool is an intermediating artifact, physical or symbolic, external or internal, which is used to transform the object. Subject, object and tool have their counterparts within the domains. The other concepts of the activity model, namely rules, community, division and outcome, are more specialized concepts or constructs in the contextual domains, and therefore they are not included in the table. Note that there are various derivatives of the original activity theory (e.g. Kuutti 1994; Korpela *et al.* 2000), which may have somewhat different counterparts in contextual domains.

In enterprise modeling, especially in the one based on the EKD approach (Loucopoulos *et al.* 1998), the concepts of goal, actor, activity, object and temporal event are defined and applied. Also a large set of relationships between the concepts are introduced to address the structure and behavior of an enterprise.

In the NATURE approach (NATURE Team 1996; Grosz *et al.* 1997) context is the "central generic" notion for representing ways-of-working. A context is composed of decisions in a certain situation. A situation is a part of the product

TABLE 13 Contextual concepts in the reviewed theories and approaches

Domains	**Case grammar (Fillmore 1968)**	**Pragmatics (Levinson 1983)**	**Speech act theory (Searle 1979)**	**Activity theory (Engeström 1987)**	**EKD Approach (Loucopoulos *et al.* 1998)**	**NATURE (Rolland *et al.* 1995)**	**Sowa *et al.* (1992)**
Context		context	context	activity		context	
Purpose			illocution	goal	goal	intention	ends
Actor	agentive	person deixis	speaker, hearer	subject	actor		agent
Action	main verb, factitive	verb	speech act	action, operation	activity	action	function
Object	objective, dative			object	object	product	entity
Facility	instrumental			tool			agent
Location	locative	place deixis	place				node
Time		time deixis	time		temporal event		time

(object). A decision reflects a choice a requirements engineer can make and refers to an intention (purpose). An intention expresses what the engineer wants to achieve. An executable context implements a decision, meaning that its intention is realized by an action (action). An action changes a product part (object).

In the framework for information systems architecture (Sowa *et al.* 1992) the aspect dimension is based on six interrogatives 'why' (motive), 'who' (people), 'how' (function), 'what' (data), 'where' (network) and 'when' (time). On a general level, they correspond to the following contextual domains: purpose, actor, action, object, location, and time, respectively. In Table 13 the generic terms for each aspect are used instead of the aspect names. Note that location is seen as a node of the network, and agent can also be a software component.

4.2.8 Summary

The purpose of this section was to build the theoretical foundation for the contextual approach. For this purpose, we started with outlining a general conception of a context, and then explored disciplines and approaches for which the purposes, meanings and effects of things, as well as the notion of a context are important. On the basis of the semiotic ladder, we confined ourselves to consider theories and approaches on the three topmost steps of the ladder (i.e. semantics, pragmatics and social world). In addition, we analyzed a set of context-related approaches that, although not building on any specific theories, share our aims at understanding and specifying the meaning of things through the notion of a context. The comparative review of this literature gave us justification for the importance of context, as well as for the division of contextual issues into seven domains: purpose, actor, action, object, facility, location, and time. In the next section we will elaborate the contextual approach and the contents of those domains.

4.3 Elaborating the Notion of a Context and Contextual Domains

In this section we first give the elaborated definition of a context and outline the contents of the contextual domains. Second, we define the contextual framework and discuss different variations of contexts with the notion of a contextual role. Third, we consider whether there are domains that are compulsory or particularly essential to perceiving the UoD as a context. Finally, we present classifications of contexts.

In Section 4.2.1 we characterized a context with the following words: (a) It is determined and shaped by one or more focal parts of which making sense is important. (b) It is composed of parts, which all have specific roles of their own. (c) It is a totality in which each part gets its meaning through the position it holds in the whole and through the relationships it has with the other parts.

After establishing the theoretical basis of the contextual approach, we can now present a more elaborated definition. It is composed of two parts, the teleological part and the structural part:

> A *context* is a conceptual or intellectual construct that can help us understand, analyze, and design natures, meanings, and effects of more elementary things in the concerned environment or circumstance.
>
> A *context* is a whole, which is determined by the focal thing(s) of which making sense is important. It is composed of interrelated things, each of which represents a certain contextual domain.

The contextual domains are: purpose, action, actor, object, facility, location, and time. In the following the contents of the domains are outlined:

- *Purpose domain* consists of those concepts and constructs which refer, directly or indirectly, to goals, motives, or intentions of someone or some thing. They may also express reasons for why someone exists, why something has been done, why someone is used, etc. in a context.
- *Actor domain* consists of those concepts and constructs, which refer to individuals, groups, positions, or organizations. Actors have an active role in a context.
- *Action domain* consists of those concepts and constructs which refer to functions, activities, tasks, or operations carried out in a context.
- *Object domain* consists of those concepts and constructs which refer to something which an action is targeted to. The object can be material or informational.
- *Facility domain* consists of those concepts and constructs which refer to means by which something can be done or is done. The facility can be a tool or a resource.
- *Location domain* consists of those concepts and constructs which refer to parts of space occupied by someone or something. The location can be physical, like a room or building, or logical, like a site in a communication network.
- *Time domain* consists of those concepts and constructs which refer to temporal aspects in a context.

To concretize a view of the relationship between the contextual approach and the core ontology we present Figure 32. According to the core ontology, and in particular on the basis of the UoD ontology (Section 3.8), reality is seen as a universe of discourse (UoD), which is composed of concepts. According to the contextual approach, a UoD is seen as a context, which is composed of concepts from seven contextual domains. In Figure 32 the outermost rectangles stand for the contextual domains, and the innermost rectangles correspond to the generic concepts that are here used as the representatives of the other *contextual concepts* of the concerned domains. The contextual concepts are inter-related to one another through *contextual relationships,* including intra-domain, inter-domain and inter-context relationships. In Figure 32 inter-domain relationships are only depicted.

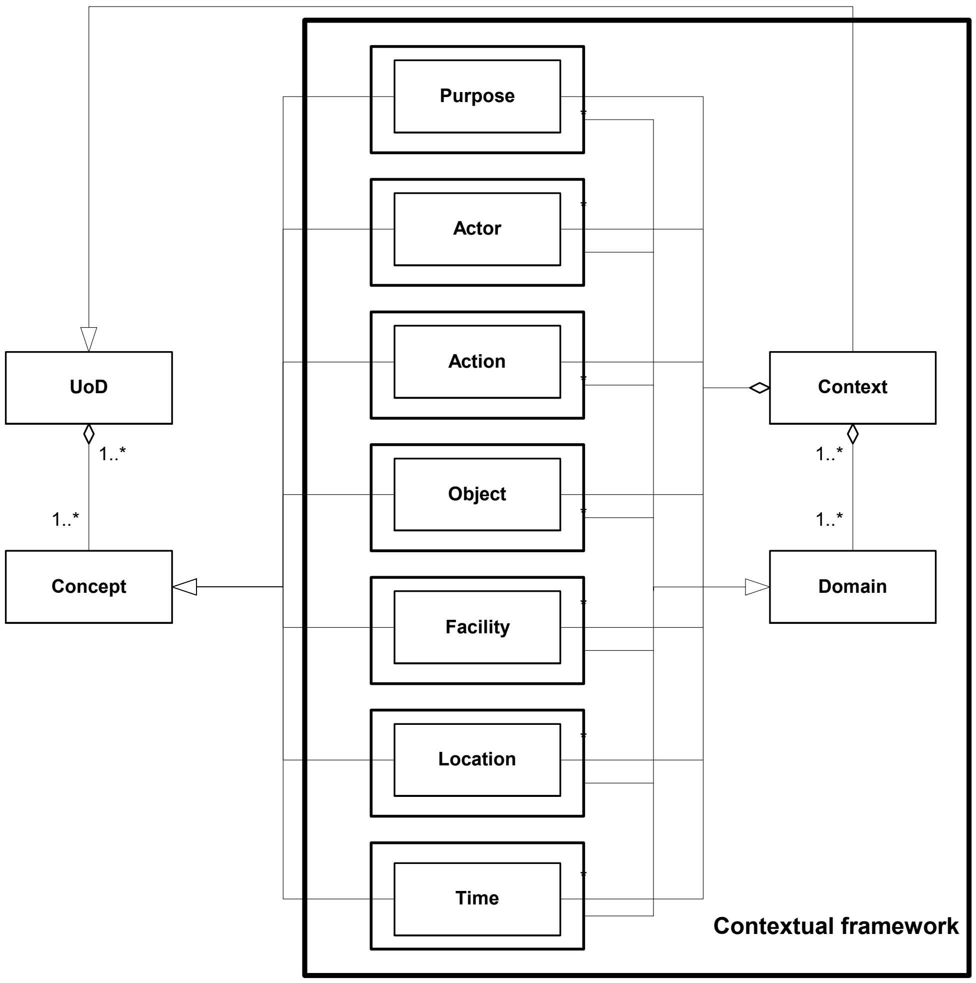


FIGURE 32 Contextual framework

In Section 3.3 (the generic ontology) we defined a framework to mean a thing that guides a human being to select the points of view that are the most appropriate for the case or the problem at hand. Here we specialize the notion and define the contextual framework as follows: The *contextual framework* is a framework, which is composed of contextual concepts related with one another through contextual relationships, and which is used to conceive things as contexts and/or within contexts. Furthermore, we can derive the definition of the context ontology: The *context ontology* provides concepts and constructs for conceiving, understanding, structuring, and representing things as contexts and/or within contexts, in order to promote our understanding of the nature, purposes, and meanings of the things.

Next, we move on to consider the notions of contextual role and role constraint, which further clarify our conceptions about a context and the contextual concepts.

A context is a whole composed of things. Each thing has a specific role in a context. For instance, a librarian is an actor in his/her searching for a copy of a book, asked by a customer, and a ladder is a facility, by the help of which a librarian tries to reach the upper shelf. A role, which a thing plays when being part of a context, is called a *contextual role.* There are certain constraints as to what things can play in which of the contextual roles. These are called *role constraints.* In the following, we consider the role constraints.

First, the same contextual role can be held by one or more things. For instance, in a communication context a sender sends a message to a receiver. A sender and a receiver are both actors. Likewise, in the context where John is collecting and analyzing samples of insects, and in the context where goods are transmitted from one location to another, there are two things from the same domain. Note that this does not mean that their specific roles would be the same.

Second, the same thing cannot be in the different contextual roles in the same context. For instance, it is not possible that a thing is an actor and an object in the same context[81]. Instead, it is quite possible that a thing is in different roles in different contexts. For instance, a stool is a facility for the librarian but an object of manufacturing in a factory. Likewise, Mary acts as a librarian (actor) and visits a hairdresser (as an object).

Third, it is not possible for any thing to act in all the contextual roles. For instance, a thing belonging to the action domain cannot be in the role of actor in the same or different context. In fact, there are two domains the concepts of which can, only to a very limited extent, participate in the roles of the other domains. These are purpose and action. Among the contextual domains the following intersections, presented in pairs, are allowed: actor – object; object - purpose; object – action; object - facility; object – location; location – facility; time – object[82].

Next, we consider whether some domains are more essential than others, perhaps even to such a degree, that without things of those domain(s), the UoD cannot be interpreted as a context at all. Phrased in terms of the abstraction ontology, it is a question about the essentiality in the composition (cf. Section 3.9.2.3): Is a context a whole with essential partOf relationships with contextual parts? If yes, what is the "nucleus" of the context?

We distinguish between three different views of the "nucleus" of a context, based on three approaches. The approaches are: the semantic approach,

[81] In fact, there are contexts in which this general constraint does not hold. For instance, a person can think about himself/herself or hit himself/herself, meaning that the person is an actor and an object in the same context. These kinds of cases are, however, exceptional.

[82] The time system of one context can be an object of creation, integration, revision, etc. in another context.

the pragmatic approach, and the approach based on the activity theory. A characteristic of the semantic approach is its aim to reveal the meaning of an expression through the analysis of its deep structure (e.g. Chomsky 1966; Fillmore 1968). For this purpose many semantic approaches concentrate on the main verbs. The verbs commonly correspond to concepts in the action domain. Besides the verbs, the subjects (mostly corresponding to the actors) and the objects (mostly corresponding to the objects) are also considered. Adjectives and adverbs (viz. purpose, facility, time, and location) are not seen so important.

According to the pragmatic approach, one focuses on revealing the meaning of a linguistic expression through looking at the situations in which the expression is created, transmitted and interpreted. Therefore, the most important part of a context is, as a matter of course, the object (i.e. expression), but knowledge on the actors (speaker, hearer), the (linguistic or instrumental) actions and the purposes (intention) are also needed. Concepts of the location domain and the time domain are not seen as important. Concepts of the facility domain are also insignificant in the pragmatic approach.

For the approach based on the activity theory, it is important to get a comprehensive and rich picture of an activity (viz. context and action). Thus, besides the action domain, which constitutes the very core of a context, the actor (subject), object (object) and facility (tool) domains are focal to the approach. The purpose, location and time domains are on the "outer level".

The views of the three approaches are illustrated in Figure 33. In the figure, for each view the contextual domains are divided onto three layers, of which the inmost one stands for the "nucleus" of a context. It is important to note that a context from the viewpoint of the semantic approach is interpreted through the contents of a linguistic expression. In contrast, a context from the viewpoint of the pragmatic approach is a situation in which expressions are uttered and used. For the approach based on the activity theory, a context is a situation in which objects of any kind are handled or dealt with otherwise. The arrows in the figure denote the signifies relationships between the linguistic objects and their conceptual referents.

Implied from the above, we can conclude that there is no universally fixed domain or set of domains, which should always be included in a context. A context itself is a choice (Dilley 1999). However, depending on the purpose for which the contextual framework is to be used, a context should contain at least concepts from the action domain (the semantic and activity theoretic approaches) and from the object domain (the pragmatic approach). Concepts from the actor domain may also prove essential. It is important to notice that to obtain a deep understanding of the meaning of some part in a context, other domains should also be involved, one way or another. Further, as for any whole that is composed of parts, the intension of a context must also contain so-called emergent predicates, to ensure right interpretation of the features of the UoD. In this chapter, we introduce the concepts of all the domains without giving any preferences for the domains.

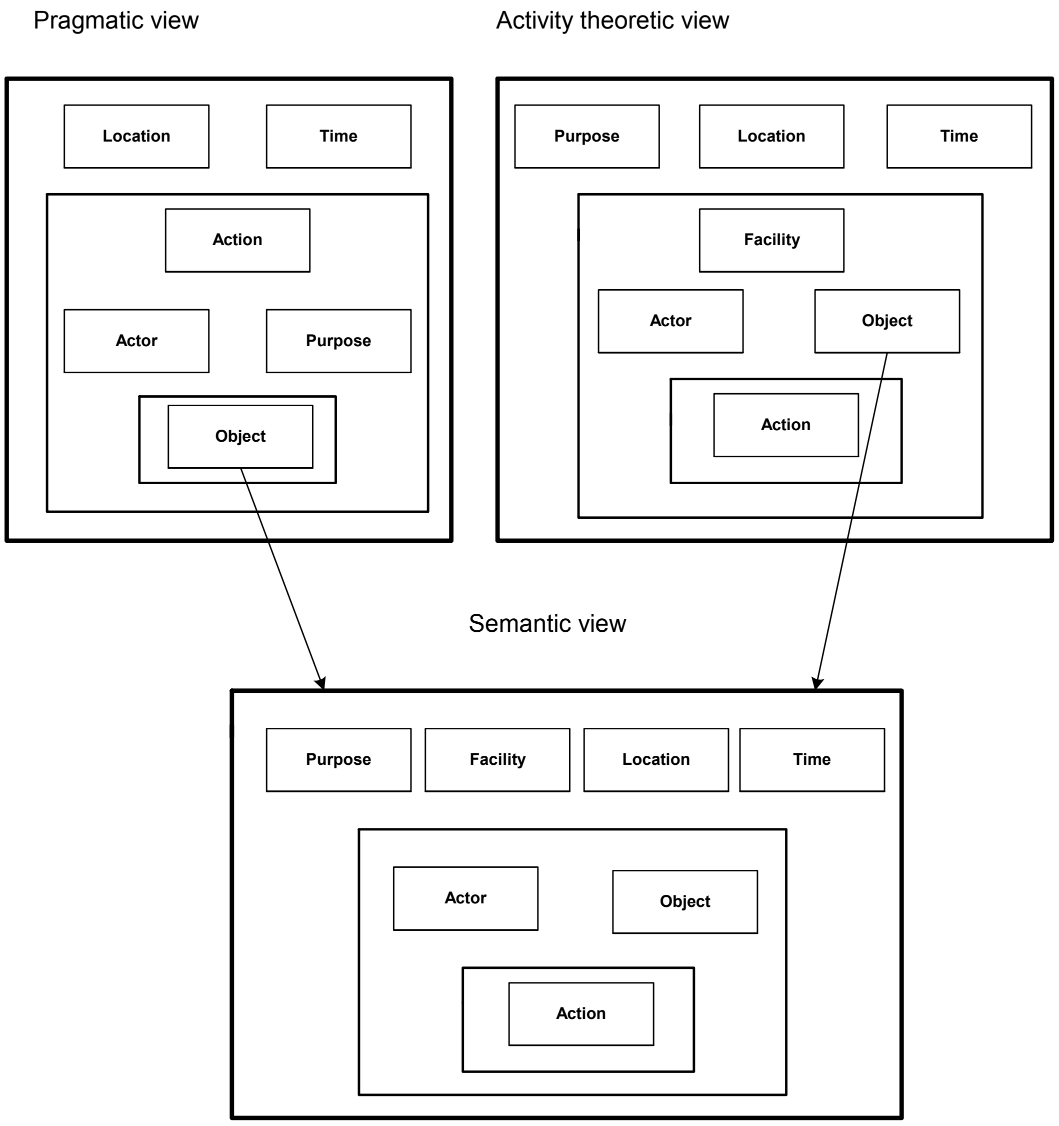


FIGURE 33 Three views on the "nucleus" of the context

At the end of this section, we present some classifications of the contexts. Depending on which of the contextual domains is the most essential, we can distinguish between the purpose-centered contexts (cf. i* model (Yu *et al.* 1995)), the actor-centered contexts, the action-centered contexts (cf. activity theory (Engeström 1987), the framework of activity analysis (Korpela *et al.* 2000)), the object-centered contexts, the facility-centered contexts, and the location-centered contexts. Based on the role the action and time domains have in the consideration, we can identify the static contexts and the dynamic contexts.

Furthermore, we can use the inter-context relationships to categorize the contexts. Assume a situation where a thing is an output from an action in one context and an input to another context. Extending the classification of Kuutti (1991, 536-537) we can now say that there are at least six different kinds of contexts:

- *Goal-producing context*. One context produces something which is used as a goal statement or a requirement in another context.
- *Actor-producing context*. One context ″produces″ objects (e.g. more skilled persons) which act as actors in another context.
- *Rule-producing context*. One context produces objects which are used as rules for another context (e.g. a method engineering context vs. an ISD context).
- *Object-producing context*. One context produces objects (e.g. services) which are utilized in another context (cf. an IS context vs. a business system context).
- *Facility-producing context*. One context produces objects which are utilized as tools or resources in another context (cf. an ISD context producing software vs. a business context).
- *Location-producing context.* One context produces objects which are used as locations in another context.

4.4 Contextual Domains

Having established the contextual approach and the contextual framework, we next define the concepts and constructs within each contextual domain, specializing from the notion of a concept defined in the core ontology. After that we specify inter-domain and inter-context relationships. The context ontology embraces all the concepts and constructs in the contextual domains, as well as the inter-domain and inter-context relationships. In each sub-section, we also discuss the relevant literature to demonstrate an extent to and a manner in which the concepts and constructs are recognized and interpreted in the literature.

4.4.1 Purpose Domain

The *purpose domain* embraces all those concepts and constructs that refer to goals, motives, or intentions of someone or something. The concepts are also used to express reasons for which something exists or is done, made, used, etc. (Webster 1989). They may show a direction toward which it is due to proceed, or a state that needs to be attained or avoid (cf. Loucopoulos *et al.* 1998; Sutcliffe 2000). They can also exhibit reasons to use or apply a facility, a time (system), or a location. The concepts are commonly named with terms such as objective, goal, intention, target, end, reason, aim, etc. We use *purpose* as the general term in this domain[83].

[83] Purpose is one of the key concepts in philosophy. Aristotle used the word *telos*, which is the goal or final cause of an action. Peirce argued that purpose is the *Thirdness* that relates some mind or mindlike entity (first), which directs the course of a process (second) toward some goal (third) (Sowa 2000, 265)

The purpose domain is important to our UoD, in which conscious human beings are conducting purposeful acts to attain desirable states. Explicit capturing of purposes is essential because it enables us to represent, not only "what" information, but also "why" information" (cf. Koubarakis *et al.* 2000, 144). However, we are not claiming that every act or event in the UoD is purposeful. There are lots of human and social activities, like chattering in a corridor, that are not pre-determined. There are a large variety of occurrences that are mere incidences.

Purpose can be expressed or can manifest oneself in many ways. On one hand, it can be regarded as an objective or *goal* (of e.g. an actor or action) meaning a desired state of affairs. It can also be related to an object, a facility, a location or a time (system), meaning the purpose, which they are aimed at. On the other hand, purpose can be expressed indirectly through a reason for something or someone. A *reason* is a basis or cause for some action, fact, event etc. (Webster 1989). A reason can be a requirement, a problem, a strength/a weakness and/or an opportunity/a threat. Between a goal and a reason there is the *dueTo relationship* meaning that a reason gives an explanation, a justification or a basis for setting a goal. In the following we define the sub-concepts of goal and reason. The meta model of the purpose domain is presented in Figure 34.

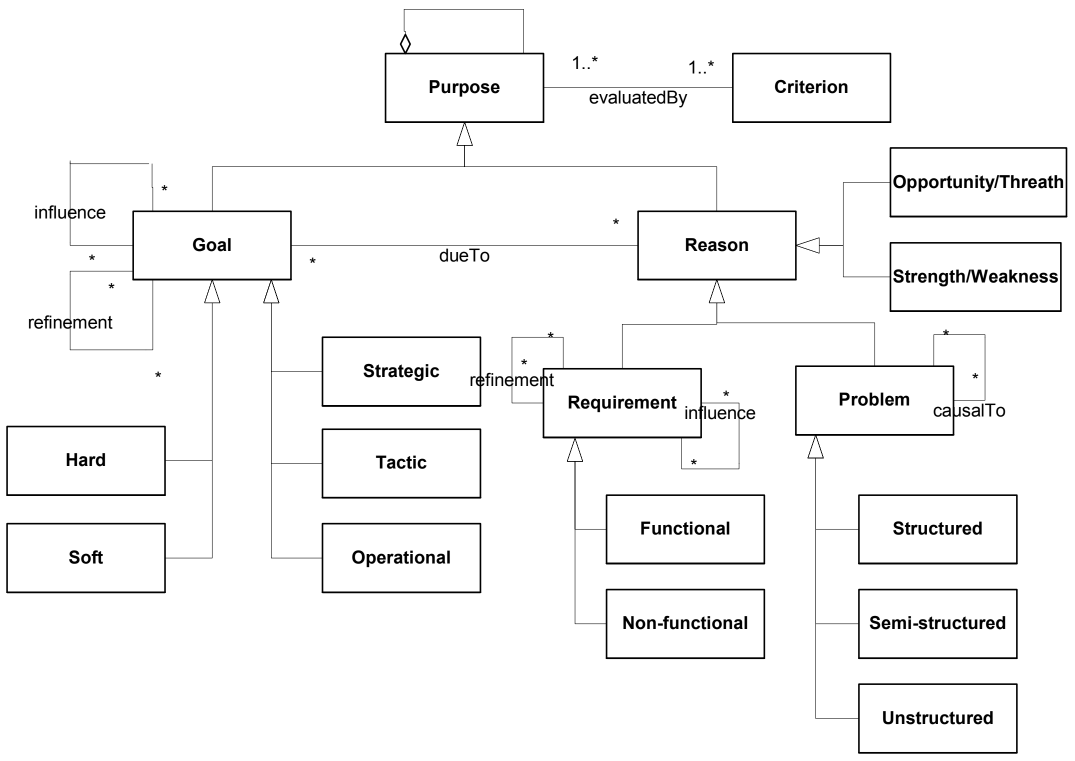


FIGURE 34 Meta model of the purpose domain

Since a goal can be considered as a desired state (Loucopoulos *et al.* 1998, 9; Koubarakis *et al.* 2000, 144), we can specialize the goals based on their lifespan. *Strategic goals* are kinds of missions, answering questions such as "What is the direction of an enterprise in the future". Their spans are generally 5 – 10 years.

Tactic goals show how to attain strategic goals. They are defined by some measurable factors within a shorter time frame. *Operative goals* are generally determined as concrete requirements that are to be fulfilled by a specified point of time. An example of the operative goal is "improvement of delivery time with 10% next year". The more concrete the goals are, the more close to operational rules they are in dictating how to fulfill the goals (Wangler *et al.* 1993, 190). The goals can also be categorized based on whether it is possible to define clear-cut criteria for the assessment of the fulfillment of goals. *Hard goals* have pre-specified criteria, and *soft goals* have not (Lin *et al.* 1999). A *criterion* is a standard of judgment presented as an established rule or principle for evaluating some thing. Further, the goals can be classified based on kinds of contexts they appear. There are business goals, information system goals, project management goals, etc. (Kueng *et al.* 1996, 100).

In some cases, purposes are expressed in terms of requirements. *Requirements* mean something that are necessary and needed. They are statements about the future (NATURE Team 1996, 525). Actually, the goals and the requirements are two sides of a coin: some of the stated requirements can be accepted to be goals to which actors want to commit. Also for the requirements there are many categorizations. One of them divides the requirements into the functional requirements and the non-functional requirements. A *functional requirement* can be achieved by performing a sequence of operations (cf. Lee *et al.* 2001, 125). A *non-functional requirement* is defined in terms of constraints, to qualify the functional requirement related to it. Non-functional requirements can involve e.g. performance, safety, quality, maintainability, portability, usability, reliability, confidentiality, security, and accuracy.

Instead of directly referring to a desirable state, a purpose can also be expressed through an indirect reference to problems that should be solved (Berztiss *et al.* 1995, 189). A *problem* is the distance or a mismatch between the prevailing state and the state reflected by the goal (cf. Goldkuhl *et al.* 1988; Jayaratna 1994, 242). To reach the goal, the distance should be eliminated or at least reduced. Associating the problems to the goals expresses reasons, or rationale, for decisions or actions towards the goals (cf. Ramesh *et al.* 1994, 296). The problems are commonly divided into structured, semi-structured and unstructured problems (e.g. Simon 1960; Gorry *et al.* 1971)[84]. *Structured problems* are those that are routine, and can be solved using standard solution techniques. *Semi-structured* and *unstructured problems* (sometimes called wicked problems (Rittel *et al.* 1984; Hevner *et al.* 2004, 81)), however, do not usually fit a standard mold, and are generally solved by examining different scenarios, and asking "what if" type questions.

Other expressions for the reasons, of not so concrete kind, are strengths, weaknesses, opportunities and threats related to something for which goals are set or are to be set (cf. SWOT-analysis, e.g. Johnson *et al.* 1989). *Strength* means something in which one is good, something that is regarded as an advantage

84 Checkland (1981) divides the problems into well-structured and ill-structured problems.

and thus increasing the possibilities to gain something better. *Weakness* means something in which one is poor, something that could or should be improved or avoided. *Opportunity* is a situation or condition favourable for attainment of a goal (Webster 1989). *Threat* is a situation or condition that is a risk for attainment of a goal. Strengths and weaknesses are internal factors, while opportunity and threat are external to the UoD.

Next we shall define intra-domain relationships in the purpose domain, first those relating the goals and then those related to the requirements and the problems.

A general goal is refined into more concrete goals. The *refinement relationship* between the goals establishes goal hierarchies, meaning that a goal can be reached when the goals below it (so-called sub-goals) in the hierarchy are fulfilled. Sub-goals are means ("how") to more general goals ("what"), for which the goals at still upper levels express "why". Consequently, the relationship can be called goal/means –relationship (e.g. Lindland *et al.* 1994), goal operationalisation, or goal satisfying –relationship, relating a satisfied goal and a satisfier goal (Kavakli *et al.* 1999, 192; Jarke *et al.* 1992, 25). In some cases, a sub-goal may contribute to the achievement of two or more general goals. In this case, the resulting structure is a goal graph, rather than a goal hierarchy. To specify which sub-goals are contributing in each case, AND and OR operators are used in the specifications. The refinement relationship is irreflexive, non-symmetric and transitive.

To indicate that the achievement of a goal has some influence on the achievement of another goal, the *influence relationship*[85] is used. Influence can be positive or negative (cf. Berztiss *et al.* 1995, 189; Loucopoulos *et al.* 1998, 10; Kavakli *et al.* 1999, 192). A positive influence between two goals means that the achievement of one goal assists the achievement of another goal. A negative influence means that the achievement of one goal hampers, jeopardizes or obstructs the achievement of another goal. Based on the kind of influence, the relationship may be referred to as the conflict relationship or the support relationship (Kavakli *et al.* 1999, 192; Lee *et al.* 2001, 128). [86]

As mentioned above, the goals and the requirements are two sides of a coin. Therefore, also the relationships between the requirements are similar to those between the goals. Consequently, a requirement can influence on another requirement, and a requirement can be a refinement of another requirement. The relationships between the problems manifest causality, and they can be analyzed with the aid of problem matrices (e.g. Lundeberg *et al.* 1981) or problem graphs. The *causalTo relationship* between two problems means that the appearance of one problem is at least a partial reason for the occurrence of the other problem.

---

[85] Liu *et al.* (2002, 40) call this relationship the contribution link.

[86] Liu *et al.* (2002, 40) define also the relationships of correlation and dependency.

4.4.2 Actor Domain

The *actor domain* consists of all those concepts and constructs that refer to human and active parts in a context[87]. Among other things actors perform, own, communicate, borrow, send, receive objects in the contexts. They are responsible for and/or responsive to triggering and causing changes in the states of objects in the same context or in other contexts. They are aware of their intentions and able to react to fulfill their goals. Their actions and reactions can be routine or intuitively caused.

Depending on a viewpoint, an *actor* is seen as a human actor or as an administrative actor. A *human actor* can be an individual person or a group of persons. A *person* is a human being, characterized by his/her consciousness, emotions, personality, beliefs, desires, intentions, social relationships, and behavior patterns conditioned by his/her culture (cf. Bratman 1987; Padgham *et al.* 1997). A person may be a member of none or several *groups.* An administrative actor is a position or a set of positions. A *position* is a post of employment occupied by a human actor. It is a set of addressable role expectations with the following properties: (a) it is occupied by a human actor, (b) it is to take care of given responsibilities, and (c) it has limited communication possibilities with other positions (cf. van Aken 1982)[88]. The *occupiedBy relationship* between a position and a human actor is antisymmetric, irreflexive, and intransitive (see Figure 35).

A position is occupied by zero or many human actors. If an actor occupies more than one position, it should be ensured that the positions have no opposing roles, resulting in role conflicts. For each position, specific qualifications in terms of skills, demands on education and experience, etc. are specified. Also many other specific constraints are related to the assignments but they are not discussed here.

An *organizational role,* shortly a role[89], is a collection of responsibilities, stipulated in an operational or structural manner. In the former case, a role is composed of tasks that a human actor occupying the position with that role has to perform. These kinds ofroles are called the process roles (Workflow Management Coalition 1999). In the latter case, a role is charged with

87 Note that our notion of actor cannot be a non-human thing. In this respect, our approach differs from many others (e.g. agent in Sowa (2000, 330), and actor in Ramackers (1994, 227) and Falkenberg *et al.* (1998)). Krogstie (1995, 10-11) distinguishes between a social actor, which can be an individual or an organization, and a technical actor that is a computational actor (hardware or software) or some other device (e.g. a clock). We regard a technical actor as a facility.

88 Zur Muchlen (1999, 5, 11) defines also the concept of position type (e.g. secretary), because for a certain position type there may be several positions (instances) and a specific position (instance) is occupied by a person. This viewpoint is common in the workflow systems.

89 We defined the notion of a role to be a part of the generic ontology. Here, a role means an organizational role. We use the shorter term when there is no risk for confusion.

responsibilities for some objects. The role of a database administrator, for instance, is charged with responsibilities, which focus on databases of the enterprise. A role can be played by many persons through the position(s) they hold. A role also summarises a set of skills or capabilities necessary to discharge the responsibilities required by the role.

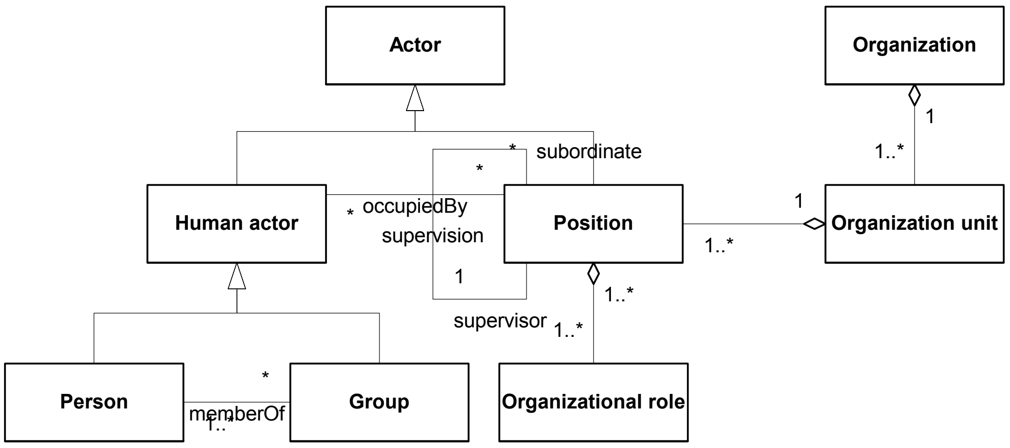


FIGURE 35 Meta model of the actor domain

The positions are related, directly or indirectly, with each other. An indirect relationship between the positions is established through relationships to and between other concepts. For instance, an intentional dependency (Kavakli *et al.* 1999, 193, called a goal dependency in Loucopoulos *et al.* 1998) reflects the fact that the achievement of a goal that a position is about is dependent on the achievement of a goal of another position. These kinds of indirect relationships are discussed in Section 4.6. Here, we next define the most essential direct relationship between the positions, namely the supervision relationship.

The *supervision relationship* involves two positions in which one is a supervisor to another that is called a subordinate. A supervisor position has responsibility and authority to make decisions upon the positions subordinate to it, and those occupying the subordinate positions have responsibility for reporting on one's work and results to those occupying the supervisor position. Responsibility is the obligation owed by subordinates to their supervisors for exercising authority delegated to them in a way to accomplish results expected (cf. Koontz *et al.* 1972). It is assigned to the position at the time the position is created or modified. Authority is the degree of discretion in positions conferring actors occupying these positions the right to use their judgment in decision making (Koontz *et al.* 1972). The supervision relationship is irreflexive and antisymmetric. The transitiveness of the relationship depends on the type of organizational structure and policy.

An *organization* is an administrative arrangement or structure established for some purposes, manifesting the division of labor into actions and the coordination of actions to accomplish the work. It can be permanent and formal, established with immutable regulations, procedures and rules. Or it may be temporally set up, like a project organization, for specific and often short-range purposes. Further, an organization may be informal and continuously evolving as those driven by the social relations emerged through informal interactions between individuals.

An organization is an extremely multifarious and complex concept. It involves human, social, juridical, economical, technical, cultural, etc. aspects. An organizational structure is reflected by many kinds of artifacts. An artifact is an object made by people, usually with skills, for subsequent use (Baskerville 1996, 10). It has a physical persistence but not necessarily physical embodiment. Artifacts represent different rules and protocols by which the members of the organization may choose to behave. These are, for instance, organization charts, personnel policies, workplace divisions, and union agreements (Baskerville 1996, 12). Human behavior quite commonly conflicts with its artifacts. For example, in many organizations the CEO's secretary wields real power more than authorized by superiors (Baskerville 1996, 10). The interaction between the structure and the human and organizational behavior occur through social processes; changing the structure influences the behavior and vice versa. Here we are not able to model an organization with all its special characteristics but view the notion through the elementary artifacts.

An organization can be enterprise-wide or established for some part of the enterprise. In the latter case, we use the term 'organizational unit'. Hence, we have organizational units of Marketing, Financing, and Data Administration. An *organizational unit* is composed of positions with the established supervision relationships, and an organization is composed of organizational units. The supervision relationships between the organizational units are derived from the ones between the positions in the concerned units. The organizational units with their positions and supervision relationships establish an organizational structure.

There are a large variety of organizational structures. Traditionally, organizational structures are divided into autocratic, participative, and egalitarian organizations, corresponding to the well-known theories of X, Y and Z (McGregor 1960; Ouchi 1981). Looking from a more technical viewpoint, we can distinguish between hierarchical and matrix-like organizations, as well as some hybrid forms. In a hierarchical organization structure, for each subordinate there is only one superior, except for the one that has the power over all the other positions. The positions are organized strictly according to the main functions, which justifies the use of the term 'functional organization'. In a matrix –like organization, for one subordinate there can be several superiors with different authorities.

4.4.3 Action Domain

The *action domain* comprises all those concepts and constructs that refer to deeds or events in a context, that is, to state transitions in reality. *Action* is used as the generic concept to refer to things belonging to the action domain. Actions can be autonomous or cooperative. They can mean highly abstract work like studies in mathematics, or at the other extreme, physical execution of a step-by-step procedure with detailed routines. They can be momentary or last hours, days or even years. The most essential characteristic of actions is that they somehow change the world, that is to say, execute state transitions (cf. Section 3.7), in either physical or mental sense. The former are called causal actions and the latter knowledge-producing actions (Koubarakis *et al.* 2002, 304-305). The knowledge-producing actions are perceptual or communicative actions. Although it is difficult to describe actions independently from related contextual domains[90], we here consider the concepts and relationships of the action domain separately from the other domains. Later in Section 4.5 the concepts of the action domain will be related to concepts of the other domains.

The actions can be classified according to several criteria. There are a large number of action structures, which an action is a part of. We divide the action structures into specific structures and generic structures. The specific structures are the management – execution (Mgmt-Exec) structure and the problem solving structure. The generic structures comprise the decomposition structure, the control structure and the temporal structure. An essential notion related to all the action structures is a rule. In the following, we will first discuss the specific structures, then the notion of a rule and finally the generic action structures (see the meta model of the action domain in Figure 36).

It is a commonplace to distinguish between the management actions and the execution actions[91]. The management – execution structure is composed of one or more management actions and those execution actions that implement prescriptions provided by the management actions. The management – execution structure is one of the cornerstones in systems theories (e.g. Mesarovic *et al.* 1970), which are concerned with control and coordination, as well as of organizational theories (Weick 1995). The dichotomy is so far-reaching that the kinds of actions can be considered to mold conceptions of the contexts in which they are. Consequently, we can talk about the management contexts and the execution contexts. The *management actions* aim to provide execution actions with prescriptions and resources. This means planning,

90 In some approaches (e.g. Loucopoulos *et al.* 1998; Kavakli *et al.* 1999, 191), processes as individual occurrences of actions are perceived as composites of four key components: the roles, the activities, the objects, and the rules.

91 For instance, Kerola and Järvinen (1975) distinguish between eight functions in ideal-seeking purposeful systems, including logistic functions, managerial or control functions, and supporting functions. Iivari (1989a) defines the conversion function (involves changes in quality, quantity, place and/or time of objects) and the development/rearrangement function (reorganizing authority relationships, reallocation of action into positions, etc.). van Slooten *et al.* (1993, 179) divide the processes into three categories: primary processes (transform inputs to outputs which are useful for the environment), regulative processes (regulate primary processes, like policy making, planning, control), and maintenance processes (obtain and maintain the means (e.g. staff, machines) of the organization). Verrijn-Stuart (1995, 271) presents a functional categorization into (a) productive activities (primary or core business) and (2) control, coordination and supportive activities.

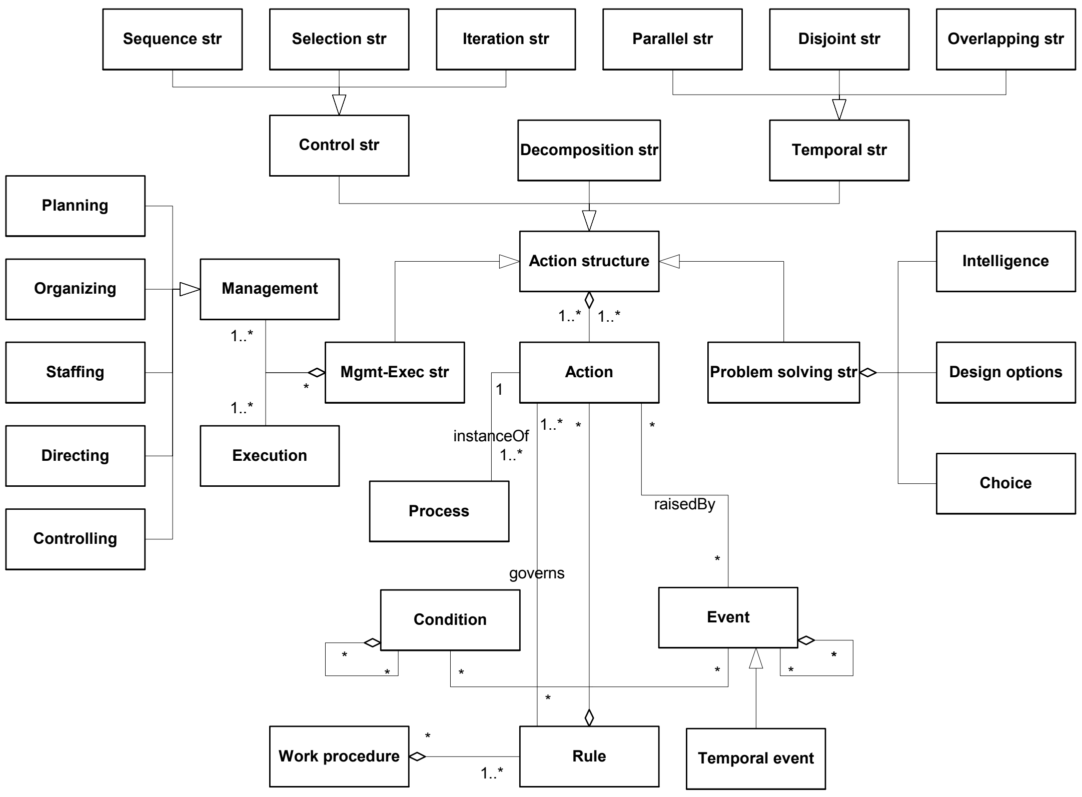


FIGURE 36 Meta model of the action domain

organizing, staffing, directing and controlling the actions of the execution contexts, in order to ensure the achievement of goals and constraints given from the environment (cf. Cleland *et al.* 1972; Sisk 1973). The management actions deploy procedures, practices, technologies, and know-how to make their courses of action effective. The purpose of the *execution actions* is to implement the prescriptions with the given resources.

The management actions are further divided into planning, organizing, staffing, directing and controlling actions (e.g. Thayer 1987). *Planning* consists of all those management actions that lead to the creation, assessment, and selection of alternative future courses of action and the program for carrying out the actions. It involves the definition of objectives and constraints as well as the development of strategies, policies, and procedures to achieve the objectives. Strategies define long-range goals and incorporate methods to obtain those goals. Policies are concerned with predetermined management decisions about e.g. personnel recruitment. Procedures establish customary ways of handling future actions and thereby allow little if any discretion.

Organizing contains all those management actions that result in the design of a formal organization structure of actions and authority relationships. It determines and decomposes actions required to achieve the objectives and arranges these actions into logical groups called roles and positions. The essential part of organizing is the creation of position titles, the descriptions of each organizational position, and the definition of the scopes, duties, authorities, and relationships for each position. *Staffing* consists of all those management actions required to fulfill and sustain filled positions that were established by organizing. This includes selecting candidates for the positions, making formal assignments, and training or otherwise developing both candidates and incumbents to accomplish their tasks effectively.

Directing consists of all those management actions dealing with the interpersonal aspects through which the personnel come to understand and contribute to the achievement of organizational goals. Once selected, assignments are clarified to the subordinates. The actors are guided, motivated, led and stimulated towards improved performance. *Controlling* is all those management actions that ensure the actual work goes according to the plans. It measures performance against the goals and the plans, shows when a deviation occurs, and by putting in motion corrective or complementary actions, helps ensure the accomplishment of plans. Controlling means observing and comparing the performance against the set standards.

As the PSC model (Kerola 1980) suggests, the pair of management and execution actions can be recognized in different contents and forms in organizations, depending on a point of view. As a matter of fact, the pair may also be recognized within a management action as well as within an execution action. Thus, there is a hierarchy of management and execution actions in which upper structures are concerned with long run policies and objectives and lower structures are implementing short run goals. A control lifespan means a time period for which plans are made and during which their execution is controlled. According to the control lifespan, the actions can be categorized into strategic level, tactic level and operational level actions (cf. strategic planning, management control, and operational control in Anthony (1965)).

The other kind of the specific action structure is the problem solving structure. It is based on the stages of problem solving (Simon 1960). Consequently, the *problem solving structure* is composed of three kinds of actions: intelligence, design options, and choice. *Intelligence* means actions that search the environment for conditions calling for a decision. They collect information based on which a decision can be made. *Design* consists of actions of inventing, shaping and specifying alternatives for possible courses of action. If the available information is found insufficient, the problem solver may choose to go back to the intelligence stage before making any further move. *Choice* means the evaluation and comparison of each alternative and the selection among them. The choice is complicated by multi-preferences, conflicting interests, and uncertainty. If needed, more information is collected, more alternatives are specified and/or specifications are further refined or revised. Hence, the stages constitute an iterative rather than a sequential process.

The action structures are enforced by rules. A *rule* is a principle or regulation governing a conduct, action, procedure, arrangement, etc (Webster 1989). It is composed of four parts (Herbst 1995, 187; Herbst *et al.* 1994, 29), event, condition, thenAction, and elseAction, leading to the well-known ECAA structure. An *event* is an instantaneous happening in the context, or in its environment, that is significant for the behaviour of the context. It means anything that has happened, happens, or can happen. An event has no duration. A *condition* is a prerequisite for triggering an action. A *thenAction* is an action that is done when the event occurs and if the condition is true. An *elseAction* is an action that is done when the event occurs but the condition is not true.

The parts of a rule are inter-related in many ways. First, both an event and a condition can be decomposed into more elementary parts with the logical operators AND and OR (cf. Herbst 1995, 188-189). Also thenAction and elseAction can be at any decomposition level. The execution of an action may raise a new event, which in favorable circumstances (e.g. the pre-defined conditions are true) leads to the execution of new actions. This is the way the rules enforce the implementation of the control structure of actions. For instance, consider the following rule: When a damage is reported and if the information about the damage is available, then a claims handler in the insurance company registers the damage provisionally and raises the event 'Damage-provisional-registered'; else he sends the damage form to the policy holder and raises the event 'Damage-Form-Sent' (Herbst *et al.* 1997, 123).

The rules can be classified in many ways (Krogstie 1995). A *dynamic rule* restricts the allowable transitions between the pre-states and the post-states. A *static rule* restricts the allowable states. An *analytic rule* is a rule that cannot be broken by an inter-subjectively agreed definition of the terms used in the rule (e.g. the age of a person is never below 0). An *empirical rule* is a rule that cannot be broken according to shared explicit knowledge (e.g. no one can travel faster than the speed of light). Analytic and empirical rules are rules of necessity, in other words they must always be satisfied. *Deontic rules* are socially agreed among the persons. Thus, a deontic rule can be violated without redefining the terms in the rule. The deontic rules can be classified into obligations, recommendations, permissions, discouragements, and probibitions (Krogstie *et al.* 1994).

As said above, the execution of an action causes one or more state transitions[92]. Every state transition is a potential event triggering another action. We can distinguish between three kinds of events (cf. Brinkkemper 1990, 54, Loucopoulos *et al.* 1998, 32): internal, external and temporal events. An *internal event* is an occurrence happening inside the context (e.g. A stock item passes the re-order level). An *external event* is an occurrence happening in the environment of the context (e.g. A client phones for information on a particular stock item). A *temporal event* is an occurrence having time as its impulse (e.g. On 31st of December at 12.00 a list of the total inventory has to be printed). A temporal event can be internal or external.

In practice many kinds of exceptions are experienced. An exception is " a case to which a rule, general principle, etc. does not apply" (Webster 1989). Exceptions can be classified according to their relations to "normal" cases

92 Note that there are state transitions which are not necessarily caused by actions; e.g. catching fire as a result from lightning.

(Auramäki *et al.* 1989, 172-175). Here, it is not possible discuss the exceptions in more detail. Next, we move on to define the generic action structures that are the decomposition structure, the control structure and the temporal structure.

In the *decomposition structure,* actions are divided into sub-actions, these further into sub-sub-actions, etc. Sub-actions may be functions, activities, tasks, operations, etc. Decomposition aims at reaching the level of elementary actions, where it is not possible or necessary to further decompose. The level of elementary actions depends on a point of view. Functional decomposition yields a hierarchical structure, which reveals the parts of each action, not their selection criteria neither the order in which they are to be performed.

A more specific view on the relationships between the actions can be obtained by looking at the control structure. The *control structure* indicates the way in which the actions are logically related to each other and the order in which they are to be executed. The control structures are: sequence, selection, and iteration[93]. The *sequence relationship* between two actions act_1 and act_2 means that after selecting the action act_1 the action act_2 is next to be selected. This implies that act_1 logically precedes act_2. For example, in order to go in, a door must be opened, and sending a submitted paper to referees requires that the paper is first received by the program chairman. In information processing the sequence relationship is commonly a manifestation of the need of the action act_2 to have at least part of the output from action act_1 as its input. The sequence relationship is antisymmetric and irreflexive. The relationship can be indirect or direct. It is direct if there is no other action act_3 between the actions act_1 and act_2 in the sequence order. The direct sequence relationship is not transitive, whereas the indirect sequence relationship is transitive.

The *selection relationship* means that after selecting the action act_1 there is a set of alternative actions act_2,.., act_n from which one action (or a certain number of actions) is to be selected. For example, after receiving reviews from the referees, an acceptance letter or a rejection letter is sent to the author(s). A selection is made based on contextual criteria. The criteria can be quite complex. They may prescribe to select one, two or several alternative actions. If several can be selected, the criteria can state which of them are mandatory or prioritized.

The *iteration relationship* means that after selecting the action act_1 the same action is selected once more. The selection is repeated until the stated conditions become true. For instance, a referee writes a review report for all the papers he/she has got for review. There can be different reasons for iterations: revision of an object due to better knowledge ("re-do"); iteration for the same object, each time on a more detailed level ("refine"); performing the action, each

93 The terms 'sequence', 'selection' and 'iteration' are adopted here due to the tradition in programming languages (e.g. Hoare *et al.* 1973), although some of them are a bit problematic. For instance, the term 'sequence' gives an impression of a temporal relationship according to which one action cannot start before the other action has finished. The temporal aspect is not, however, involved here.

time for different objects, due to, for instance, lack of processing capacity in terms of manpower, facilities, or space ("repeat") (cf. Goldkuhl *et al.* 1993).

Note that a part of the control structure can also contain an inner control structure. For instance, the sequence relationship can associate two actions, which in fact are complex actions including compositions of actions with, say, another sequence relationship or an iteration relationship. Likewise, among a set of alternative actions, there can be an inner control structure specifying, for instance, in which logical order selected actions should be executed.

The third generic action structure is the temporal structure. The *temporal structure* is like the control structure but with temporal conditions and events. A temporal condition means a condition, which contains at least one temporal expression. A *temporal event* is a time-driven event. In the temporal structure, the actions are bound to the time axis, with either absolute or comparative terms. An absolute term signifies a time unit (time point or time interval) in some time system; e.g. at 10.00 pm. Conditions and events are expressed in comparative terms if they signify, for instance, the beginning or ending events of some other actions; e.g. act_2 should not start before the end of the action act_1. Hybrid terms contain absolute and comparative parts: e.g. if an order is not assembled within 20 days after the order is registered, remind the responsible clerk (Herbst 1995, 188),

The temporal structures are specified using temporal constructs[94], such as during, starts, finishes, before, overlaps, meets, and equal. Constructs are used to specify the relationships between the starting and/or ending events, or between the durations of the actions. With these constructs, overlapping, parallel, disjoint (non-parallel) and overlapping executions of actions can be distinguished. Two actions are said to be *overlapping* if the durations of their executions overlap; i.e. the action act_2 starts before the action act_1 ends. The actions are (strictly) *parallel* if the durations are equal or the duration of one action is included in the duration of the other action. Two actions are said to be *disjoint* if their durations do not overlap. The actions are strictly sequential, if the action act_2 starts exactly after the action act_1 ends; i.e. there is no elementary time unit between the ending event and the starting event. More temporal concepts and relationships will be defined in Section 4.4.7.

The generic action structures can be positioned in a continuum to reflect an extent to which the relationships between actions are constrained. Consider the action structures in Figure 37. In the decomposition structure the relationships between the actions are specified on the most general level. The action *act* is decomposed into four parts and one part further into three subparts. No other relationships, except the partOf relationships, are specified. On the next general level (the control structure), the parts are also interrelated by logical relationships (an arrow describes a sequence, a box containing parts describes a selection, and an arrow returning to the part stands for an iteration). On the most specific level (the temporal structure) the parts are bound to the

[94] The temporal constructs will be specified in Section 4.4.7.

time axis, in this case with comparative terms (i.e. without explicit references to time units).

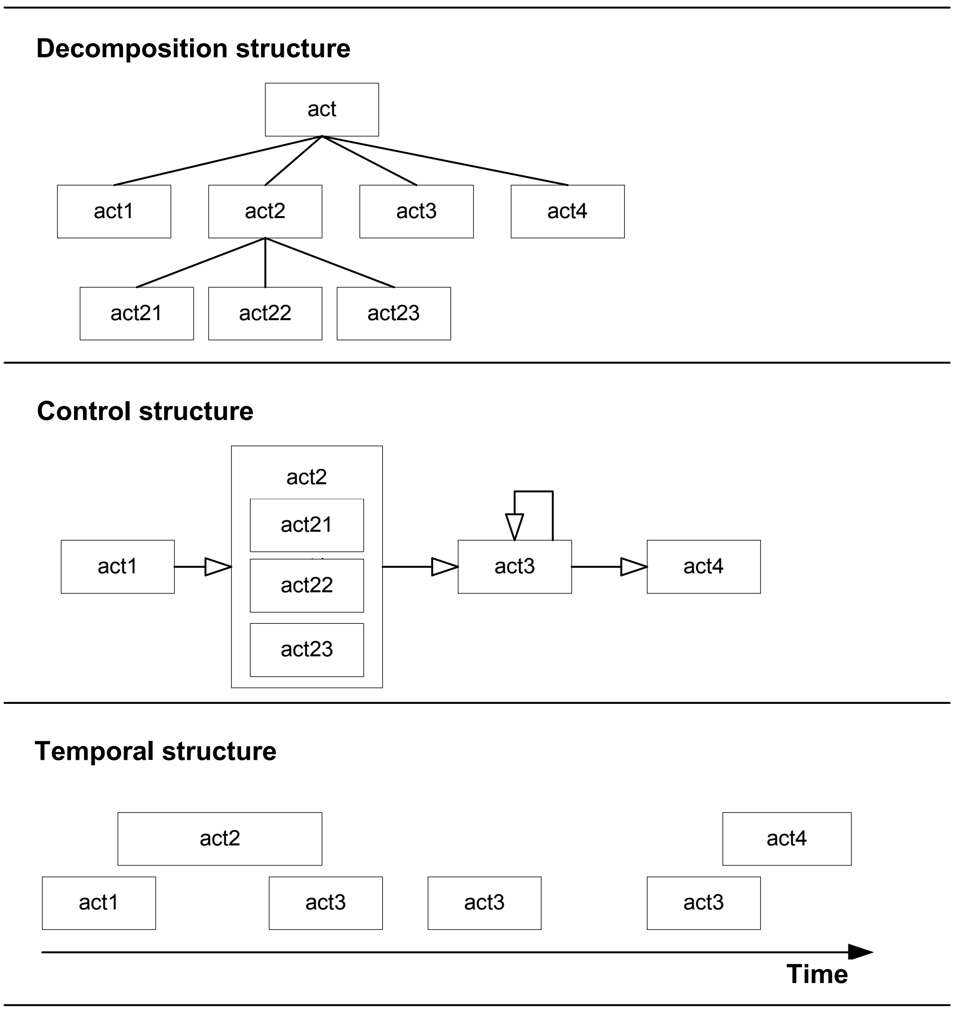


FIGURE 37 The action *act* seen from the viewpoints of three generic structures

An aggregate of related rules constitutes a *work procedure* (cf. Iivari 1989a, 333), which prescribes how the course of action should proceed. Depending on the knowledge of, and a variety of, actions, work procedures may be defined at different levels of detail. Hoc (1988) identifies three kinds of modes of prescriptions: declarative, functional and procedural. Declarative prescriptions express the desired state of the UoD, often in terms of expected outcomes (i.e. "What"). Procedural prescriptions give more explicit, often step-by-step, guidance for the course of actions (i.e. "How"). An intermediate type of prescriptions, termed the functional prescriptions, establishes relations between the actions and the positions but does not have the status of prescriptions for operational procedures.

The considerations above have dealt with the concepts and relationships in the action domain at the type level. Due to the importance of the actions in the context ontology and a large variety of instance-level occurrences of actions, we next consider the actions at the instance level.

An instance of an action is called a *process*. For one action, several process enactments can occur. A process may have sub-processes, which may have sub-sub-processes, etc. A process can occasionally terminate and after a while resume. Another process can be enacted, one after another, as a new iteration of the same action. Consequently, a variety of instance-level structures, which the process enacted from the certain action may have, is very large but all of them have to comply with the structures and rules specified for the action. To illustrate the relationships between an action and its processes as well as a variety of processes of an action, we present the process p1 of the action *act* on the decomposition, control and temporal levels in Figure 38. We can see that in the process p1 two sub-sub-processes p121 and p122, corresponding to the sub-actions act21 and act22, are enacted. The sub-process p13 is executed with iteration, leading to the enactment of three sub-sub-processes p131, p132 and p133.

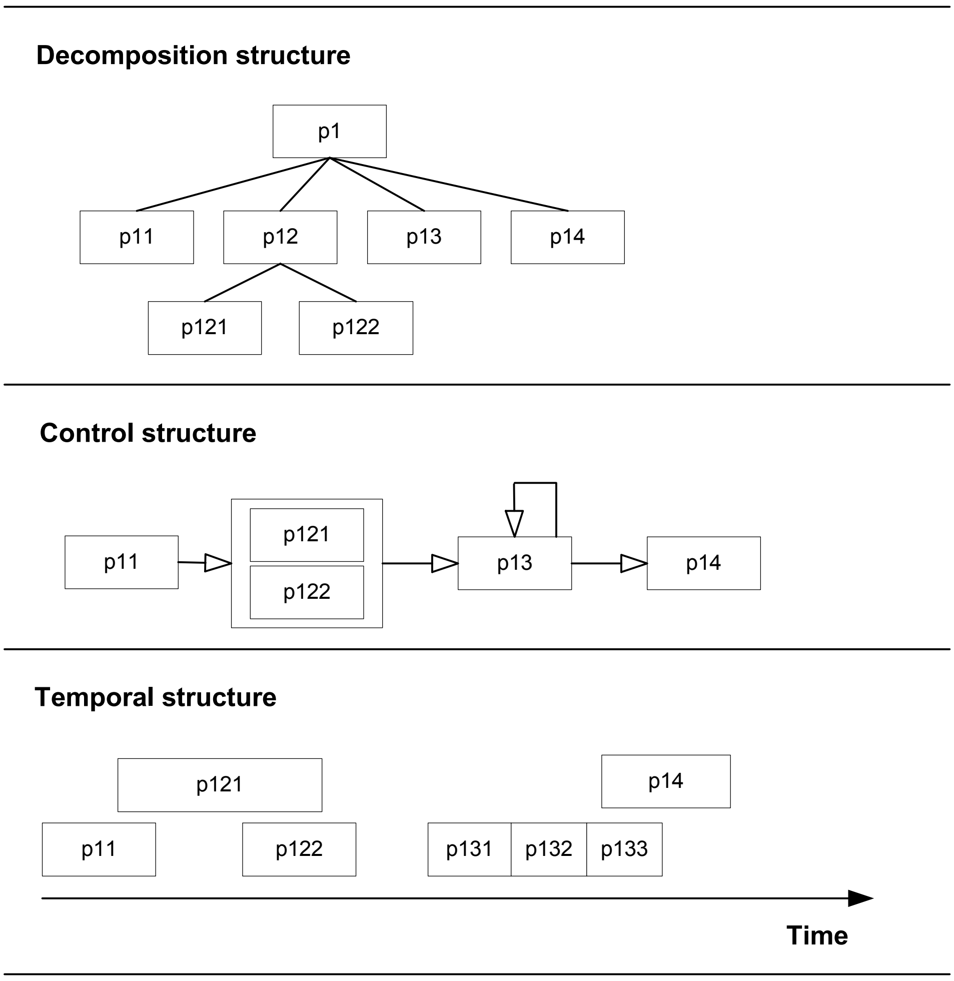


FIGURE 38 The process p1 of the action *act* seen on three levels of action structures

Due to the fact that action is such a focal notion to information processing, even more classifications for actions are presented in the IS literature. Most of them can be categorized based on the contextual domains. Hence, depending on whether the actor of an action is a human being, a computerized tool or both, the actions can be classified into manual, automatic or semi-automatic actions. Manual actions are totally performed by human beings, whereas automatic actions are executed by a computerized tool (e.g. auto pilot system in an air plane). Linguistic actions (cf. speech acts in Searle *et al.* (1979)) deal with linguistic objects, whereas instrumental actions concern physical objects with no representational function. Personal actions are conducted by individuals, while collaborative actions presume group work. Based on the time domain, we can distinguish between non-recurring actions and recurring actions. Further, we can recognize centralized actions and decentralized actions.

4.4.4 Object Domain

The *object domain* contains all those concepts and constructs that refer to something, which an action is directed to. It can be a message, a decision, an argumentation, a list of problems, a program code, CASE tool graphics on the screen, a workstation, etc. In general, an object can be a conception in a human mind, the data represented in some carrier, or physical material such as timber, a machine or a house (cf. the semiotic realms). Also a person as a physical thing can be an object as is the case when a person is in the barber. In the literature an object is called an information/material set (Olle *et al.* 1988a), material/ information (Iivari 1989a), a dataset (Harmsen 1997), data (van Swede *et al.* 1993), a document (Ang *et al.* 1993), a resource (Freeman *et al.* 1994), and an actand (Falkenberg *et al.* 1998). We use 'object' as the generic term to signify any concept in the object domain.

Based on the nature of the objects we can distinguish between material objects and informational objects. *Material objects* do not carry or present any information, whereas *informational objects* do. For us, objects of special interest are in the form of data or information. We call them data objects, or *linguistic objects,* and information objects or *conceptual objects,* respectively[95]. *Service* is something tangible or intangible, composed of material and/or informational objects, made or given for someone from which it benefits.

Linguistic objects can be classified according to languages in which they are presented. They can be *formal, semi-formal* or *informal* (cf. the language ontology in Section 3.6). Informational objects can be classified based on the intentions by which the objects are provided and used. Our aim here is to develop a simple (lattice) taxonomy that serves as a basis to elaborate further the object domain. The taxonomy has benefited from the classifications of Stamper (1973, 1975, 1978a), Searle *et al.* (1985), and Lee (1983). Next we define

[95] This division is in accordance with the well-know distinction between a datum as "a representation of information" and information as "an interpretation of a datum" (cf. Langefors 1971).

the concepts included in the taxonomy (see Figure 39 for the meta model of the object domain).

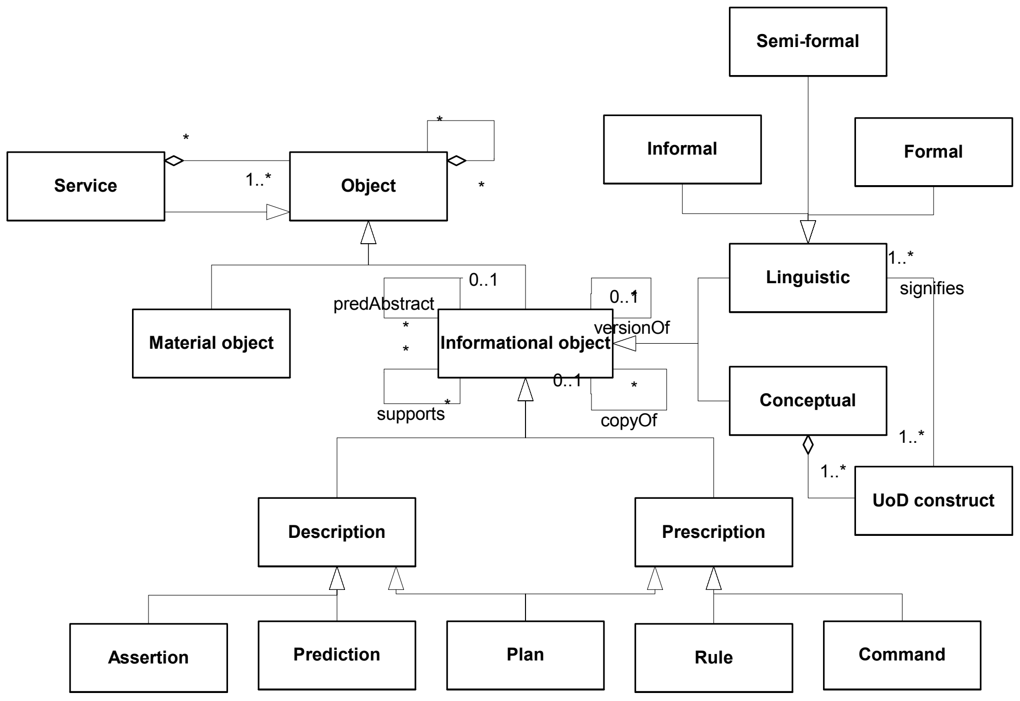


FIGURE 39 Meta model of the object domain

Stamper (1978a) distinguishes between the denotative and affective modes of a language. Accordingly, there are informational objects, which merely signify objects in the UoD, and those which affect upon the human feelings. We constrain ourselves to discuss mainly the informational objects in the denotative mode. These informational objects can be descriptive or prescriptive. A descriptive object, called a *description,* is a representation of information about a slice of the UoD (the actual or possible world). It provides a picture of reality to enable an actor as a planner or as a decision maker to take actions even if he/she is at some distance from the "world" in question. A prescriptive object, called a *prescription,* is a representation of the established practice or an authoritative regulation for action. It is information that says what must or ought to be done. It can be in the form of an order or an instruction, a rule or a regulation, a recommendation or an advice.

A distinction between a description and a prescription can be illustrated and elaborated by the direction of fit (Searle *et al.* 1985). The direction of fit discloses how the propositional content of an informational object relates to reality. Three main cases are: (a) The propositional content fits a state of affairs in reality; (b) States of affairs are changed to fit the propositional content; (c) The propositional content induces the intended alteration in the state of affairs. The two first cases correspond to the descriptions and the prescriptions, respectively. The third case stands for declarations, which are not distinguished in our ontology.

An informational object can be descriptive in various ways. An *assertion* is a description, which asserts that a certain state has existed or exists, or a certain event has occurred or occurs. The propositional content of the assertion fits a past or existing state of affairs. A *prediction* is a description of a future possible world with the assertion that the course of events in the actual world will eventually lead to this state (cf. Lee 1983).

Also the prescriptions can be specialized. First, a prescription can be expressed in different forms. Most commonly a prescription can be reduced into the ECAA form: If Event and Condition, then Action$_1$; else Action$_2$ (Herbst 1995). As defined in Section 4.4.3, a prescription with at least two parts ((E or C) and A) is called a *rule*. A prescription with neither an event part nor a condition part is called a *command.* Second, the prescriptions are different from one another in regard to their stability. On one hand, we have prescriptions that, to a high degree, remain unchanged. A good example of these is the rule which forbids Muslims from eating pork. On the other hand, there are prescriptions that are subject to periodic or occasional changes (e.g. the tax legislation). Third, we can distinguish between first-order and second-order prescriptions (Stamper 1978a). Second-order prescriptions are used to modify first-order prescriptions in certain contexts.

An informational object may possess aspects of several intentional subtypes. For instance, a *plan* is, on one hand, a description about what is intended. It can also be regarded as a kind of prediction, which is augmented with intentions of action. It is assumed that the future possible world described in the plan would not normally come out, except for the intended actions (cf. Lee 1983). On the other hand, no plan is prepared without considering its implementation. Consequently, it contains a requirement to act in order to change the states of affairs. As regard to the states in which a plan can appear during the courses of action, we can distinguish possible, probable, proposed, and approved plans (cf. Glasson 1986, 272).

There are many important relationships between the objects. Except the abstraction relationships, most of the relationships are type-specific. In the following we consider five generic relationships between the informational objects, which are: the versionOf relationship, the copyOf relationship, the supports relationship, the predAbstract, the signifies relationship, and the partOf relationship.

An object is often produced gradually through several iterations. A *version* is a result of an iterative or phased action toward the final outcome. It can be a preliminary or tentative object, the final product itself, or something between them. The *versionOf relationship* holds between two objects obj$_1$ and obj$_2$, if properties of, and experience from, the object obj$_1$ have influenced the creation of another object obj$_2$ intended for the same purposes and if the objects refer to the same UoD (cf. the *is_derived_from* relationship in Katz (1990)). In some cases, the objects are considered versions albeit the purpose has evolved. The creation can get its start from "scratch" or it can be based on updating, elaborating or improving the earlier version. The versionOf relationship is irreflexive, antisymmetric, and transitive. The versions can establish a version tree, in which several alternative versions are made from one object and one of the versions is selected for further development. The process of producing versions is iterative.

We may have several copies from an object. A characteristic of the *copyOf relationship* is that the original object and a copy object are exactly, or to an acceptable extent, similar. Depending on the nature of the original object, there may be some or even millions of copies (cf. an e-mail message forwarded as a copy to thousands of users through a public mailing list). If the copies are equal to the original object, there is no need to maintain the knowledge about which of the objects is original. In this case the copyOf relationship is reflexive, symmetric, and transitive. Otherwise, the relationship is irreflexive, antisymmetric and intransitive.

The *supports relationship* involves two informational objects, obj_1 and obj_2, such that the information "carried" by the object obj_1 is needed to produce the object obj_2. For instance, to place an order, it is necessary to know what the current quantity-on-hand of the product is, who the potential suppliers are, what the prices of the products are, what the delay of supplying is, etc. The supports relationship is irreflexive, antisymmetric and intransitive. We can distinguish between two subtypes of the supports relationships. Depending on the role in which the information conveyed by the object obj_1 plays in producing the object obj_2, the relationship can be the data supports relationship or the control supports relationship. In the *data supports relationship,* obj_1 means "raw data" that is converted into obj_2. In the *control supports relationship*, obj_1 means "control data", i.e. rules, policies, principles or other prescriptions, according to which the object obj_2 is to be produced. The versionOf relationship is a subtype of the data supports relationship.

The *predAbstract relationship* between two informational objects means that one object is more abstract that the other object in terms of predicate abstraction (see Section 3.9.3) and both of the objects signify the same thing(s) in the UoD. For instance, a document containing the description of functional properties of a machine is more abstract, in terms of predicate abstraction based on the realization criterion, than another document, which contains a diagram of electric wiring. The predAbstract relationship is irreflexive and antisymmetric. In a hierarchical system of predicate abstraction levels, the relationship is also transitive.

The *signifies relationship* defines the conceptual meaning of the linguistic object in terms of UoD constructs, which the object signifies. The relationship is a specialization of the relationship between a sign and a concept, defined in the semiotic ontology (Section 3.4)[96]. The *UoD construct* means any conceptual construct in the same or different context[97]. Above we distinguished between five kinds of informational objects: assertion, prediction, plan, rule and command. They reflect various illocutionary forces (Searle 1969, 1979) or "pragmatic meanings". It would be possible to specialize the signifies

96 We use the term 'signifies' for the relationship here in spite of this specialization.

97 The notion of a UoD construct will be discussed more closely in Section 6.3.3.

relationship into five special relationships between a specific linguistic object and a UoD construct, one for each of the object kinds. This is not done here.

The partOf relationship in the object domain means that the object is composed of two or more other objects that are called object parts (cf. product elements or sub-products in Schmitt (1993, 238)).

4.4.5 Facility Domain

The *facility domain* contains all those concepts and constructs that refer to the means by which something can be accomplished, i.e. something, which makes an action possible, more efficient or effective. We distinguish between two main kinds of *facilities:* tools and resources (see the meta model of the facility domain in Figure 40).

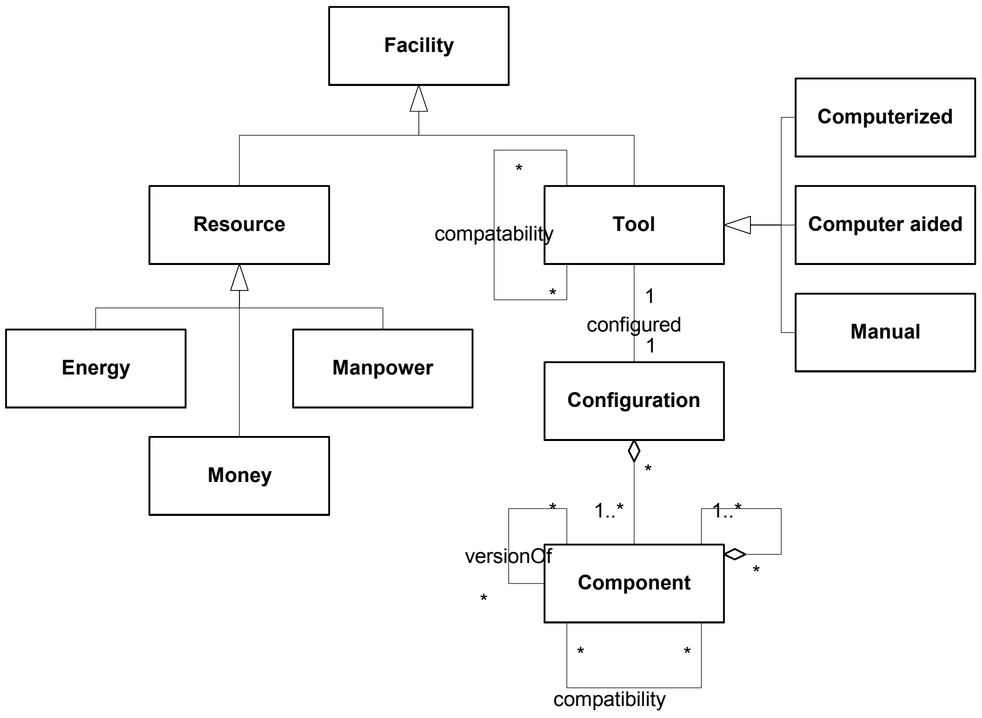


FIGURE 40 Meta model of the facility domain

A *tool* is a thing that is designed, built, installed, etc. to serve in a specific action affording a convenience, efficiency or effectiveness. A tool may be, for instance, a simple and concrete instrument held in hand and used for cutting or hitting. Or, it may be a highly complicated computer system supporting an engineer in his/her controlling a nuclear power station. In contrast to an actor, a tool has a supportive role in the context: it is activated or taken "into hand" when needed by a skilled actor. In a technical context, a tool may have a major role in action (cf. an auto pilot in the air plane), but there also it is assumed to be under the control of a human being[98]. Tools can be *manual, computer aided,* or *computerized.*

[98] By this standpoint, we strongly support McGregor (1960) who points out a difference between a human being and a machine: "The distinctive potential contribution of the

A *resource* is a kind of source of supply, support, or aid. It can be money, energy, capital, goods, manpower, etc. (Barros 1991, 539). The resources are not interesting in terms of pieces, e.g. in individual coins or men, but rather in terms of amount. When a resource is used, it is consumed, and when consuming, the amount of the resource diminishes. Thus, a resource is a thing, about which the main concern is how much it is available (cf. Liu *et al.* 2002, 39). In this respect, the notion of a resource sharply differs from the notion of a tool.

There are a great number of relationships between the concepts within the facility domain, representing functional, structural and other kinds of connections. As we do not want to emphasize too much technical aspects of a context, we content ourselves with defining some examples of the relationships among the computer aided and computerized tools. These are: compatibility, versionOf, and configuration.

For being operative and useful, the tools should be compatible**.** Two tools are *compatible* if their interfaces are structurally and functionally interoperable. The compatibility relationship is reflexive and sometimes symmetric. Compatibility can be considered from several viewpoints. For instance, the basic reference model of Open Systems Interconnection (OSI), developed by ISO (1984), establishes an architecture with seven compatible layers. On each layer, a particular viewpoint is applied to establish compatibility for data communication between components of information processing systems. In this case compatibility is not symmetric. Besides on data, compatibility can be based on presentation, control, and process of the tools (Thomas *et al.* 1992).

Tools are commonly composed of one or more components. Components develop through consecutive *versions*. Only some versions of a component are compatible with certain versions of the other component. To manage compositions of components of different versions, the notion of a configuration is used. A *configuration* is a whole that is composed of the components with compatible versions. For instance, a software house must manage customized configurations of software for specific platforms or for other special requirements of customers (see Pohl (1994) for discussions of upwards and downwards compatibility).

4.4.6 Location Domain

The *location domain* contains all those concepts and constructs that refer to parts of space occupied by someone or something. A *location* can be physical or logical. A *physical location,* such as a room, a building or a city, is a spatial thing, which is placed in a region of space and which can, through its spatial attachment, provide a place for some other thing (e.g. a person in a building). We distinguish between spatial things and the space they are placed in (Borgo *et al.* 1996; Bittner 1999). A *spatial thing* may be a building, a street, a river or the

human being in contrast to the machine, at every level of an organization, stems from his capacity to think, to plan, to exercise judgment, to be creative, to direct and to control his own behavior.

like, that is to say, some thing that is necessary or beneficial to localize. A *region* is a part or division of space[99]. A *point* is the elementary unit in space specified by a single coordinate with reference to a system of two or three geographical dimensions. An *area* is any particular extent of space specified with at least two coordinates. A *geographical dimension* means any dimension within which space can be specified. A *geographical system* is a system of two or three geographical dimensions. A spatial thing is associated with a region by the *placedIn relationship.* Distinguishing a spatial thing from a region it is placed in is important to enable recognizing that something is moved across space (Shanaham 1995). Depending on the match between the granularities of the spatial things and the regions of space, the placedIn relationship can denote an exact place, a part place or a rough place (Bittner *et al.* 2002).

A *logical location,* like a site within a computer network, is a space that is not attached to any geographical point or area. We make a clear distinction between a location as a space and the contents of the location. The latter can, depending on a viewpoint, be an actor, an object, or a facility. Next, we define relationships in the location domain (see the meta model of the location domain in Figure 41).

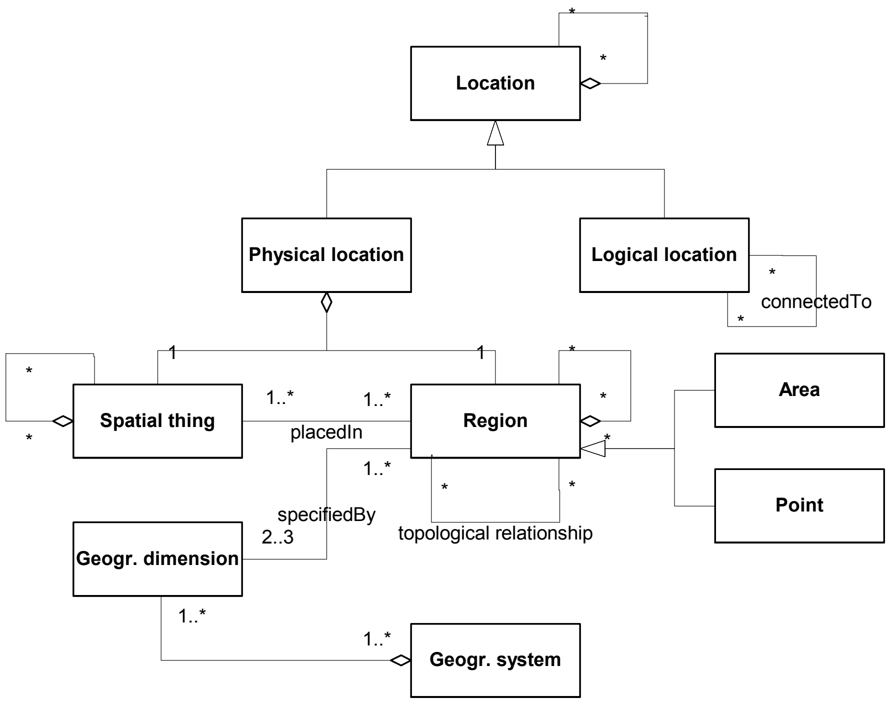


FIGURE 41 Meta model of the location domain

With the expansion of geographical information systems (GIS's) and other location-aware applications, e.g. in 3G networks, it has become more and more necessary to collect, model, store and analyze information originating from

99 There are many alternative ways to divide space. Sowa (1995, 670), for example, defines three elementary "geographical features" with which space can be divided: area, line, and point.

maps, digital images, satellites, roads, transportation networks, etc. A prerequisite for these is that the relationships between the locations are strictly defined. There is a special theory of parts, called mereology (e.g. Varzi 1996), which defines wholes and parts and their location-based relationships. The most essential relationships are the partOf relationship, also called the parthood relationship, and the topological relationships. The geographical partOf relationship associates both the spatial things and the regions of spaces. Cities contain buildings, which comprise apartments, which consist of rooms, etc. Granularity of the regions depends on the scale and metrics relevant from the adopted viewpoint[100]. Regions are measured as areas along the geographical dimensions or abstracted as points in spatial space[101].

Geometrical dimensions enable to establish various topological relationships between the regions. A *topological relationship* between two regions states how the regions are related in terms of geographical points or areas along two or three geometric dimensions. Depending on the spatial theory applied, various primitive relationships have been defined. Randell's theory (e.g. Randell *et al.* 1989), based upon Clarke's (1981) calculus of individuals, for instance, assumes the primitive dyadic relationship 'connects', which means that the connected regions share a common point. Based on this primitive, other relationships like disconnected, externally connected, partial overlap, being a tangential part of the other, etc. can be defined. Based on different spatial theories, relationships such as separates, contains, surrounds, above, below, to the side of, etc. can be defined (cf. Stamper 1978a, 67). The topological relationships are used in topological operations, such as interpolation and proximity analysis, to make inferences.

For the logical locations, e.g. terminal sites, we propose different relationships. For instance, in an electronic mail system, it is necessary to maintain knowledge about all possible sites and connections between them. The *connectedTo* relationship means that two sites can communicate with each other through exchange of messages.

4.4.7 Time Domain

The *time domain* contains all those concepts and constructs that refer to the temporal aspects of the UoD. *Time* is indefinite, unlimited duration in which something is considered as happening in the past, present, or future (Webster 1989). Most of our knowledge is founded in time and expressed in terms of time units. The two fundamental *time units* are the time point and the time interval. The *time point* is the primitive as an indivisible point on the time continuum (e.g. Kahn *et al.* 1977). The *time interval* is an abstraction of time points,

100 There is a specific set of literature that deals with different classifications in this respect (see for an overview in Freundschuh *et al.* 1997).

101 The approaches in topology literature differ from one another in their view of what are the primitives. Some regard the regions as primitives (Clarke 1981), whereas others operate with more elementary particulars like lines and points.

manifesting duration of something. Note that a point in time is relative, dependent on the chosen point of view. For instance, a day is, in one context, a time point, and, alternatively, a time interval in some other contexts.

There are many different time theories in the literature. Based on Hayes (1995) we distinguish between (a) instant-based theories (e.g. McDermott 1982, Shoham 1987), which operate with time points, (b) interval-based theories (e.g. Allen 1984), which are founded on intervals, and (c) temporal theories (e.g. Bochman 1990), which treat both time points and intervals as independent primitives. Here we follow the instant-based approach, because it is more natural to regard an interval as being composed of time points. Next we consider the sub-concepts of, and the relationships between, the time units. The meta model of the time domain is presented in Figure 42.

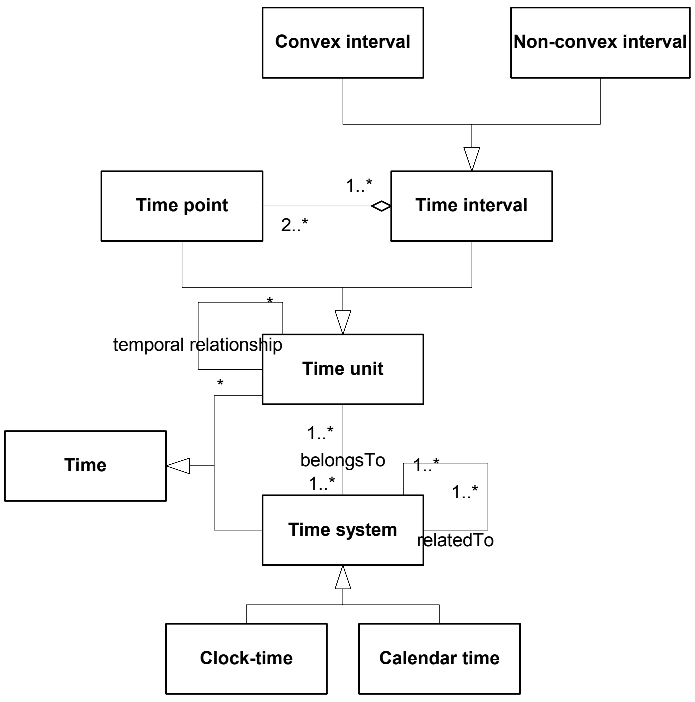


FIGURE 42 Meta model of the time domain

There are many kinds of time intervals. First, there are convex time intervals and non-convex time intervals (Zhou *et al.* 2000). A *convex time interval* is an interval that consists of continuous time points (e.g. January 3, 2002). A *non-convex time interval* is an interval with some "holes" (e.g. Wednesday). Further, we can distinguish the *regular non-convex time intervals*. "Every Wednesday in September", for instance, consists of 4 or 5 connected intervals, each of which represents a Wednesday.

Time expressions can be definite or indefinite (Oberweis *et al.* 1988). To a definite expression, only one predicator applies whereas for an indefinite expression, like "something happens before or during the time unit tim_1", several predicators combined with OR-operators are used. Time expressions can make absolute or comparative references to time units. Absolute references contain only concepts in the time domain, whereas the interpretation of comparative references, such as "something happens after the action act_1 is accomplished", applies concepts in the other contextual domains (e.g. in the action domain), too.

It is common to all human and social behavior that time is perceived in relation to some socially constructed time system. A *time system* is a totally ordered set of time units (Clifford *et al.* 1988). It is always based on a certain time resolution or granularity, which is selected to suit the purposes in hand. In physics, one of the billion fractions of a second is a relevant time unit, whereas in paleontological research, units of millions of years are used to express eras, in which prehistoric organisms have lived and evolved. The most universal time systems are Julian and Gregorian calendars. A more customized time system is a fiscal time containing fiscal year and fiscal month, which may vary in different countries. Another universal time system is the clock time system that enables reckoning and measuring time through the natural and prominent cycles of day and night. For organizational contexts, a composition of the common calendar date with the clock time system suffices. Because time is also said to be "the system of those sequential relations that any event has to another" (Webster 1989), we consider time as a generic concept generalized from the concepts of time unit and time system.

The relationships between the time systems may grow to very complicated constructs. To define a *relatedTo relationship* between the time systems usually requires that some elementary time system is established; e.g. a continuous number line with real numbers. The time systems form hierarchies in which a time point in one time system at a higher level (e.g. Calendar-Year) is seen a time interval from the perspective of another time system on a lower level (a Calendar-Month). It is much more difficult, however, to relate the systems containing weeks or other not so straightforward time units to the other time systems. For our study, the relationships between the time systems are not an essential issue and therefore we do not discuss it more.

A large set of *temporal relationships* can be defined for the time points and the time intervals. Three binary relationships defined for the time points are: *before*, *after* and *equal-point*. Following Allen's time theory (Allen 1984), we can further define a minimal set of relationships between the time intervals (intv)[102]:

- *during* ($intv_1$, $intv_2$). $intv_1$ is fully contained with $intv_2$.
- *starts* ($intv_1$, $intv_2$). $intv_1$ shares the same beginning as $intv_2$, but ends before $intv_2$ ends.
- *finishes* ($intv_1$, $intv_2$). $intv_1$ shares the same end as $intv_2$, but begins after $intv_2$ begins.
- *before* ($intv_1$, $intv_2$). $intv_1$ is before $intv_2$, and they do not overlap in any way.
- *overlaps* ($intv_1$, $intv_2$). $intv_1$ starts before $intv_2$ and they overlap.

102 There are also other suggestions (e.g. De *et al.* 1982) that contain even 28 temporal relationships.

- *meets* ($intv_1$, $intv_2$). $intv_1$ is before $intv_2$, and there is no period between them.
- *equal* ($intv_1$, $intv_2$). $intv_1$ and $intv_2$ are the same interval.

There is also a large set of axioms that define the semantics of the relationships (Allen 1984). It can be proved, for instance, that the overlapping intervals have a common sub-interval and the temporal relationships are transitive. In addition, there are a large variety of functions on the time points and the time intervals (e.g. Year_of, Hour_of, Starting_point, Co_Start, etc.). We are not able to consider them here in more detail.

4.5 Inter-Domain Relationships

Until now we have defined only those contextual relationships which associate concepts in the same contextual domain. There is, however, a large set of contextual relationships that relate concepts in different domains. For example, an actor carries out an action, an object is an input to an action, and a facility is situated in a location. We call them the *inter-domain relationships.*

Figure 43 presents an overview of the inter-domain relationships. The space is divided into seven areas corresponding to seven contextual domains, with the action domain being in the centre. In each sub-areas we present the concerned generic concepts to be related with the inter-domain relationships. In the following we define the inter-domain relationships, starting from those that involve the concepts of the purpose domain, and then continue, domain-by-domain, with defining the relationships that concern concepts within the other domains.

A. Purpose

expressedBy(Actor,Purpose)
The relationship means that an actor has expressed a goal, a requirement, a problem, or the like concerning the context as a whole or some of its part (e.g. actions, objects, locations), in the same or different context.

motivatedBy(HumanActor,Purpose)
When associated with a human actor, a purpose means a subjective or inter-subjective motive or an inner drive that makes a person or a group to do something or behave as he/she/it does. A purpose may be a far-reaching or near-reaching goal, idealistic or realistic, nevertheless it is something for which a human actor is ready and willing to struggle. Individuals may have their own “selfish” goals, for instance, to advance their personal career, which differ from the goals of the organizational unit they are working for.

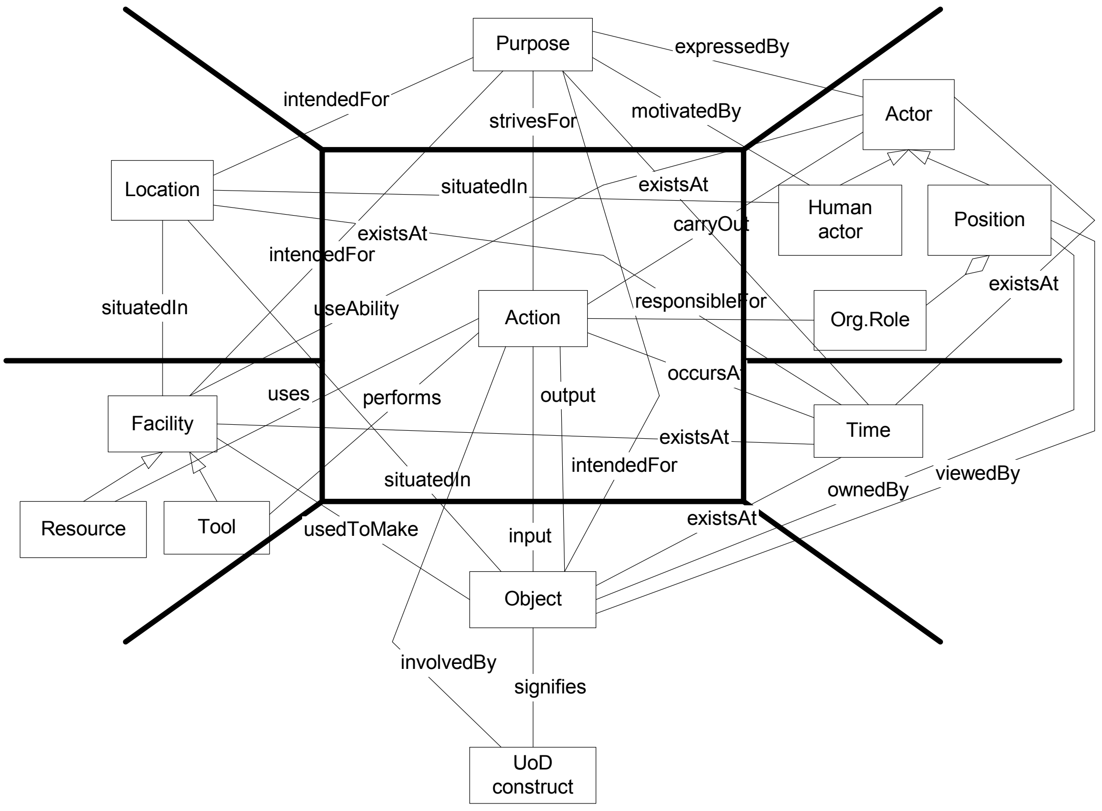


FIGURE 43 Overview of inter-domain relationships

strivesFor(Action,Purpose)
When associated with an action, a purpose means a goal for which an action strives. A goal can be strategic, policy-level, or highly operative and concrete. Unlike for the objects, here a purpose through the strivesFor relationship particularly means a goal for a process, not for its outcome.

intendedFor(Object,Purpose)
When associated with an object, a purpose means a goal or a reason for which an object has been made, is made, or is to be made. For instance, an IS should be used to support invoicing, and customer data should be correct and timely enough to serve the customer relationship management.

intendedFor(Facility,Purpose)
When associated with a facility, a purpose means a goals or reason for which a facility has been made/acquired, is made/acquired, or is to be made/acquired. Commonly used purposes of a facility are: to make an action more efficient and easier, to make a service more available, etc.

intendedFor(Location,Purpose)
When associated with a location, a purpose means a goal or reason for which a location has been made/acquired, is made/acquired, or is to be made/ acquired.

existsAt(Purpose,Time)
The relationship defines time for the existence of a purpose.

B. Actor

carryOut(Actor,Action)
The relationship means that an actor carries out an action. The relationship is considered general enough to cover all kinds of functioning and behavior in various temporal and modal forms. Actor is here used as a generic concept meaning a human actor or a position occupied by a human actor.

responsibleFor(Org.Role,Action)
The relationship defines the functional contents of an organizational role. Depending on a level of detail on which an action is expressed, the role specification can include a general outline of responsibilities for an action, or a composition of operations in the form of decomposition, control or temporal structures. There are a large variety of principles and rules guiding and constraining the composition of actions to make organizational roles and the composition of organizational roles to establish positions. A position should, for instance, facilitate some job autonomy, variety and rotation of work among a set of positions. Positions should also constitute natural units of work, and be clearly defined and organized (Mumford *et al.* 1979; Hedberg 1980; Steenis 1990).

ownedBy(Object,Actor)
The relationship defines that an actor is an "owner" of an object, and therefore, he/she has some responsibilities for and authority over an object. Depending on organizational agreements and policies, an actor takes care of an object, can utilize an object, or can grant privileges to some other actors for using or modifying an object (e.g. granting permissions to access to data), etc.

viewedBy(Object,Actor)
The relationship defines that an object presents views, insights, opinions, etc. of a certain actor. If associated with a person or a group of persons, an object represents a subjective or inter-subjective view, whereas if associated with a position, an object reflects an organizational view or a so-called 'official' view. Through this relationship it is possible to present differences between and conflicts among the views (see views in data bases (e.g. Motschnig-Pitrik 2000), in the socio-technical theory (Mumford *et al.* 1979), in the Softsystems theory (Checkland 1981), and in the political theories (e.g. Robey 1984; Ciborra 1998)).

useAbility(Actor,Facility)
The relationship defines that it is possible for an actor to use a facility. If a facility is a tool, this means that he/she has the required skills and rights for its use. If a facility is a resource, the relationship means that he/she is authorized to use or make decisions on the acquirement and use of the resource.

situatedIn(HumanActor,Location)
The relationship defines that a human actor (a person or a group) is situated in a location.

existsAt(Actor,Time)
The relationship defines the time for the existence of an actor. Through this relationship the lifespan of an actor (i.e. a person, a group, a position, or an organizational unit) is established.

C. Action

The inter-domain relationships between the actions and the objects may be highly complex. In the literature, two major approaches have been applied in modeling action structures connected to the objects. In the implicit approach, the input and output relationships are used as such (e.g. Ramackers 1994; Sowa *et al.* 1992) or the particular concept of a flow is used (e.g. Olle *et al.* 1988a; Harmsen 1997). In the explicit approach, a special construct is defined for the action structure. This construct can be a whole (e.g. Communication in Falkenberg *et al.* (1998), Transaction in Verrijn-Stuart *et al.* (1992, 486), Business exchange in Sowa *et al.* (1992)), or an objectified relationship (e.g. Organizational channel in Iivari (1989a)). Heym and Österle (1992a, 233) define input usage and output usage constructs to enable the expression of the purpose (output create, output modify) for which the usage occurs. In this study we contend ourselves with a simple approach, which is based on defining the separate input and output relationships between the actions and the objects.

input(Object,Action)
The relationship defines that an object is used as an input to an action. An object is called an input object, and the corresponding action is called a consumer or a destination.

output(Action,Object)
The relationship defines that an action produces an object as its output. An action is called a provider or a source, and an object is called an output object.

involvedBy(UoD-construct,Action)
The relationship defines that a UoD construct is involved by an action through informational objects that signify a UoD construct. Involving may mean creating, modifying, utilizing, or deleting informational objects.

performs(Tool,Action)
The relationship defines that an action is performed by a tool.

uses(Action,Resource)
The relationship defines that an action uses certain resources. The level of detail in which resources are referred to may vary a lot.

occursAt(Action,Time)
The relationship defines when an action is done, has been done or will be done.

D. Object

usedToMake(Facility,Object)
The relationship defines that a certain facility, a tool or a resource, is used to produce an object. The relationship could be specialized according to a kind of facility, a nature of the use, amount (of certain resources) used, etc.

situatedIn(Object,Location)
The relationship defines that an object is situated in a certain location.

existsAt(Object,Time)
The relationship defines when an object exists or has existed.

E. Facility

situatedIn(Facility,Location)
The relationship defines that a facility is situated in a given location.

existsAt(Facility,Time)
The relationship defines when a facility exists or has existed.

F. Location

existsAt(Location,Time)
The relationship defines when a location exists or has existed.

To have an overall picture of the intra-domain relationships and the inter-domain relationships defined above, we present the relationships in Table 14. The intra-domain relationships, containing all but the generalization (isA) and composition (partOf) relationships, are presented in the diagonal, and the inter-domain relationships are located in the cells below the diagonal. The abbreviations used in the expressions are assumed to be self-describing.

4.6 Implicit Relationships

Based on the intra-domain and inter-domain relationships defined above, a large set of *implicit contextual relationships* can be derived. In this section, we first give examples of these relationships and then illustrate the implicit inter-context relationships with examples. The implicit relationships are not included, on the individual level, in the context ontology.

TABLE 14 Intra-domain and inter-domain relationships

	Purpose	Actor	Action	Object	Facility	Location	Time

Purpose	influence (goal,goal)
	refinement (goal,goal)
	dueTo(goal,reason)
	causalTo (prob,prob)
	refinement(req,req)
	influence(req,req)						

Actor	expressedBy (pur,actr) motivatedBy (physActr,pur)	occupiedBy (pos,humActr) supervision (pos,pos) memberOf (per, group)					
Action	strivesFor (actn,pur)	carryout (actr,actn) responsibleFor (role,actn)	governs (rule,actn) raisedBy (event,actn) instOf(proc,actn)				

Object	intendedFor (obj,pur)	ownedBy (obj,actr)
		viewedBy (obj,actr)
			input(obj,actn)
			output(actn,obj)
			involvedBy (UoD,actn)
				versioOf(obj,obj)
				copyOf(obj,obj)
				supports(obj,obj)
				predAbstract (obj,obj)
				signifies(lobj,UoD)			

(continues)

TABLE 14 (continues)

	Purpose	**Actor**	**Action**	**Object**	**Facility**	**Location**	**Time**
Facility	intendedFor (fac,pur)	useAbility(actr,fac)	uses(actn,res) performs (tool,actn)	usedToMake (fac,obj)	configured (conf,tool) versionOf (com,com) compatability (com,com) compatability (tool,tool)		
Location	intendedFor (loc,pur)	situatedIn (humActr,loc)		situatedIn (obj,loc)	situatedIn (fac,loc)	placedIn (spatThing,regi) topologicalRelationship(regi,regi) connectedTo (logLoc,logLoc)	
Purpose	existsAt(pur,tim)	existsAt(actr,tim)	occursAt (actn,tim)	existsAt (obj,tim)	existsAt(fac,tim)	existsAt(loc,tim)	temporal relationship (tim,tim) belongsTo (tim,tsys) relatedTo (tsys,tsys)

The organizational role has been previously defined as a collection of responsibilities and authorities. Based on the relationships that a role has to an action (responsibleFor), an action has to a purpose (strivesFor), and an action has to an object (input, output) and to time (occursAt), we can derive (implicit) dependencies between the roles. An intentional dependency between the roles reflects the fact that the achievement of a goal that one role brings about is dependent on the achievement of a goal of another role (Kavakli *et al.* 1999, 193). A coordination dependency expresses the need for one role to wait for completion of another role's responsibilities before it can complete its own (Loucopoulos *et al.* 1998, 19-20; Kavakli *et al.* 1999, 193). A resource dependency illustrates the need for one role to use a resource that can be provided by another role. For instance, a construction service requires material that is under the supervision of warehousing services (Loucopoulos *et al.* 1998, 19; Yu *et al.* 1995).

Further, an actor is a tool user, if he/she carries out an action, which is partly performed by a tool. An information provider is an actor who carries out actions that produce informational objects that are used as input to actions carried out by another actor. A facility provider is an actor who carries out actions to produce an object that is used as a tool by a tool user. A schedule for the actions can be derived from a set of the occursAt relationships.

Implicit relationships can also associate the contexts. For instance, there is a managing relationship between two contexts if the actions of the management-execution structure are divided in such a way that the management actions belong to one context and the execution actions belong to the other. A context is a provider if its actions produce objects for the use of another context. For instance, an information system is a provider context for a business context. We can also distinguish between a signifying context and a signified context. The former context produces and/or uses informational objects that signify some things in the latter context. Further, the temporal relationships between two contexts follow from the temporal relationships between the corresponding actions, and the topological relationships between two contexts follow from the topological relationships between the locations.

Abstraction among the parts of the contexts also results in abstraction relationships between the contexts, as the following examples show. By classification individual contexts are abstracted into a context type. By the inverse process, a context is instantiated. Consider the following example:

```
[Context: Cxt || [Actor: Secretary ], [Action: Stores ],
[Object: Document ] ]
```

By instantiating any of the three contextual concepts (Secretary, Stores, Document), a more concrete context is achieved:

```
[Context: Cxt1 || [Actor: Mary ], [Action: Stores ], [Object:
Document ] ]
```

```
[Context: Cxt2 || [Actor: Secretary ], [Action: Stores ],
[Object: Document#123] ]
```

Hence, we have derived the implicit inter-context relationships instanceOf (Cxt1, Cxt) and instanceOf (Cxt2,Cxt).

Let us next consider generalization. Assume the contexts Cxt3 and Cxt4 below:

```
[Context: Cxt3 || [Actor: Secretary ], [Action: Stores ],
[Object: Document ] ]
[Context: Cxt4 || [Actor: Secretary ], [Action: Stores ],
[Object: DesignDocument ] ]
```

Generalizing any of the three contextual concepts in the contexts Cxt3 and Cxt4 results in a more generalized context as follows:

```
[Context:  Cxt || [Actor: Person], [Action: Stores ],
[Object: Document ] ]
```

Hence, we have derived the implicit relationships isA (Cxt3,Cxt) and isA (Cxt4,Cxt).

Abstraction among the contextual concepts does not always imply the corresponding abstraction among the contexts. Consider the following example about grouping:

```
[Context:  Cxt5 || [Actor: Committee], [Action: Makes]
[Object: Decision]]
```

Let us assume that there is the relationship memberOf (Representative, Committee). From this it does not necessarily follow that

```
[Context: Cxt6 || [Actor: Representative], [Action: Makes]
[Object: Decision]],
```

because decisions are made by the committee as a collective unit, not by individual representatives. There are, however, cases, in which grouping among the contextual concepts indeed implies a grouping relationship among the contexts. For example, if

```
[Context: Cxt7 || [Actor: Committee], [Action: TravelsTo]
[Location: London]],
```

then it is quite possible to derive a context in which an individual representative travels to London. Note, however, that this does not necessarily hold for instantiated contexts, because it is not certain that every representative of the committee visits London.

4.7 Summary and Discussions

The purpose of this chapter was to present the context ontology, as the first component among the so-called contextual ontologies. The context ontology provides the concepts and constructs to conceive, understand, structure and represent things in reality as contexts and/or within contexts. For engineering the ontology, we first specified a particular approach, called the contextual approach. We characterized the application domain at which this approach was aimed, defined the objectives, searched for theories underlying the approach, and crafted the notion of a context. A context and the contextual approach are vital, not only to the context ontology, but also to the whole ontological framework. Therefore, we made a serious attempt to construct a solid theoretical and conceptual basis for that.

The contextual approach is a conception-oriented approach, which helps us understand and specify purposes, meanings, and effects of things, through considering them as parts of a context. It is a kind of abstraction mechanism by which we can reveal what is relevant for what we aim to explain and exclude all other that does not offer the requested explaining power (Sharfstein 1989). The notion of a context is widely used in e.g. formal logic, pragmatics, computational linguistics, sociological linguistics, organizational theory, cognitive psychology, and information systems. We were particularly interested in context-related theories that are situated on the three topmost levels of the semiotic ladder (Stamper 1973; Stamper 1996). They are semantics, pragmatics and theories of human and social world. Each of these theories was considered and several approaches based on them were described and evaluated for applicability. In addition, we reviewed a large variety of studies that use the notion of a context, although without any explicitly defined context-related theories. These studies are related to data bases, enterprise modeling, workflow management, user modeling, process modeling, and information systems architecture.

Building on this theoretic basis, we defined a context to be a complex construct that is composed of concepts from seven contextual domains. The domains are: purpose, actor, action, object, facility, location, and time. Although there is no universally fixed aggregate of domains, which should always be included in the context, we recognized domains that commonly form the so-called "nucleus" of the context. Depending on a selected point of view, these domains are the action domain, the object domain, and the actor domain.

The context ontology is composed of concepts and constructs that are related with one another through a number of intra-domain, inter-domain and inter-context relationships. For each contextual domain, we defined the concepts and relationships and presented them in meta models. We also made plenty of references to the relevant literature and compared existing suggestions with ours. In addition, we defined a large set of inter-domain relationships and considered what kinds of implicit relationships between the domains and between the contexts can be inferred and how.

The context ontology provides a means to substantially detail the view of reality that was formed with the core ontology in Chapter 3. Things that were seen in the core ontology to be elementary and "instrumental" get now particular contextual meanings. The context ontology has been derived, in a transparent fashion, from the core ontology by specializing and elaborating its concepts and constructs.

As the context ontology has a focal role in OntoFrame, we next consider the ontology with the quality criteria given in Section 1.3. The contextual approach and the context ontology are rooted on universal theories, thus contributing to naturalness and comprehensiveness. Most of the terms we use are common and self-evident without wordy definitions. The context ontology is comprehensive, assessed in terms of the coverage of the contextual domains and the features derived from the theories for. The comprehensiveness can also be argued with comparisons to the corresponding artifacts in the literature (e.g. Olive 1983; Zachman 1987; Iivari 1989a; Sowa *et al.* 1992; Olle *et al.* 1988a; van Swede *et al.* 1993; Freeman *et al.* 1994; Harmsen 1997). This kind of comparison will be presented in Section 5.6.

The context ontology can be easily extended specializing existing concepts and constructs. Due to the large size of the ontology we have been obliged to exclude several specific areas that can be, if necessary, equipped with new concepts and constructs and integrate into the context ontology. Such areas may concern, for instance, tools (e.g. computer architecture, communication network) and organizations (e.g. group working dynamics, informal organizational forms).

The context ontology is applicable to comparative analyses, as will be demonstrated in Section 5.6 (cf. the analytical intention of use). The ontology is also of vital importance in engineering more specific component ontologies of OntoFrame in the next chapters (cf. the constructive intention of use). The notion of a context with its seven contextual domains becomes clearly visible in all the lower-level component ontologies.

5 LAYER ONTOLOGY

In the previous chapter we defined the context ontology, which provides a comprehensive set of concepts and constructs to conceive, understand, structure, and represent the structure and behavior of the contexts in general. Applying the context ontology we can make sense of the meanings of the things by considering them to be a context or part of a context. In this chapter we will focus on more specialized contexts, namely on contexts the purpose of which is solely information processing. We define the *layer ontology*, which provides concepts and constructs to conceive, understand, structure and represent static and dynamic features of information processing at four layers. The ontology is derived from the context ontology by specialization (Figure 44).

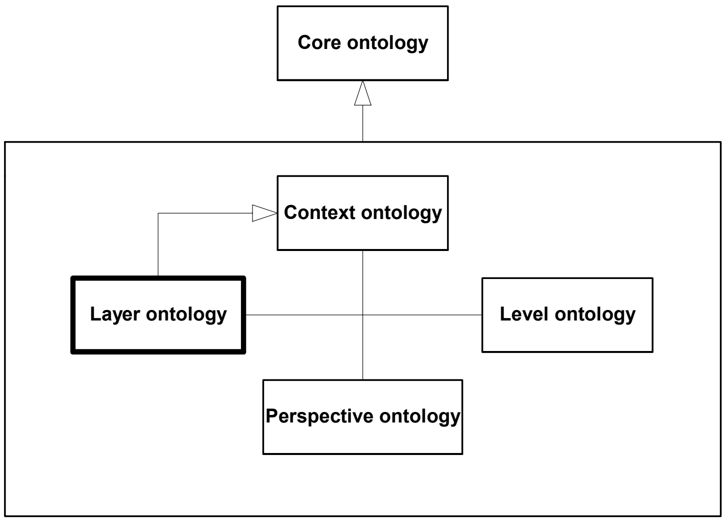


FIGURE 44 Focus of Chapter 5

The layer ontology is composed of two parts. The first part provides concepts and constructs related to information processing in general. The second part of the ontology shows how information processing is structured and related onto four layers according to a predefined system of layers.

The chapter is organized into four sections. In Section 5.1 we define the basic concepts pertaining to information and information processing. We also distinguish between the information system, the object system, the utilizing system, and the controlled system, define them and discuss relationships between them. In Section 5.2 we recognize the primary actions and the development actions in information processing. Deriving from this dichotomy we define the system of four processing layers. The layers are: information system, information system development, method engineering, and research work. Each layer is characterized from the teleological, functional and structural viewpoints. We also discuss the contents of, and the relationships between, the contexts positioned at these layers. In Section 5.3 we specialize the notions of utilizing system and object system to concern each of the processing layers. The chapter concludes with a summary.

5.1 Information Processing

To enable the considerations of information and information processing in various forms and on various layers we need special concepts and constructs. We start this section by defining the notions of knowledge, data, information, and information processing. Then we distinguish between four contexts, which are related to information processing. Because it is commonplace in the IS field to regard those contexts as systems, we call them the information system (IS), the object system (OS), the utilizing system (US) and the controlled system (CS). After defining them and discussing the relationships between them, we make a comparative analysis of the relevant literature. Figure 45 presents the meta model of the concepts and relationships that will be defined in this section. It also shows how the concepts are related to the generic concepts defined in the context ontology. This meta model is the first part of the layer ontology.

5.1.1 Basic Concepts

Human and social actions are based on expertise and its accumulation mainly through communication. Expertise is *knowledge*, which is a relative stable and sufficiently consistent set of (conceptual) informational objects owned by single human actors (cf. Falkenberg *et al.* 1998, 66). There are two kinds of knowledge: explicit knowledge and tacit knowledge (Nonaka *et al.* 1995). *Explicit knowledge* can be articulated in a natural or formal language, which makes it 'easy' to transmit knowledge between people (cf 'shared knowledge' in Falkenberg *et al.* 1998, 71). *Tacit knowledge* is a body of knowledge that is embedded in personal experience and therefore cannot be (easily) represented externally. It shows up only in the actions of the person having that knowledge.

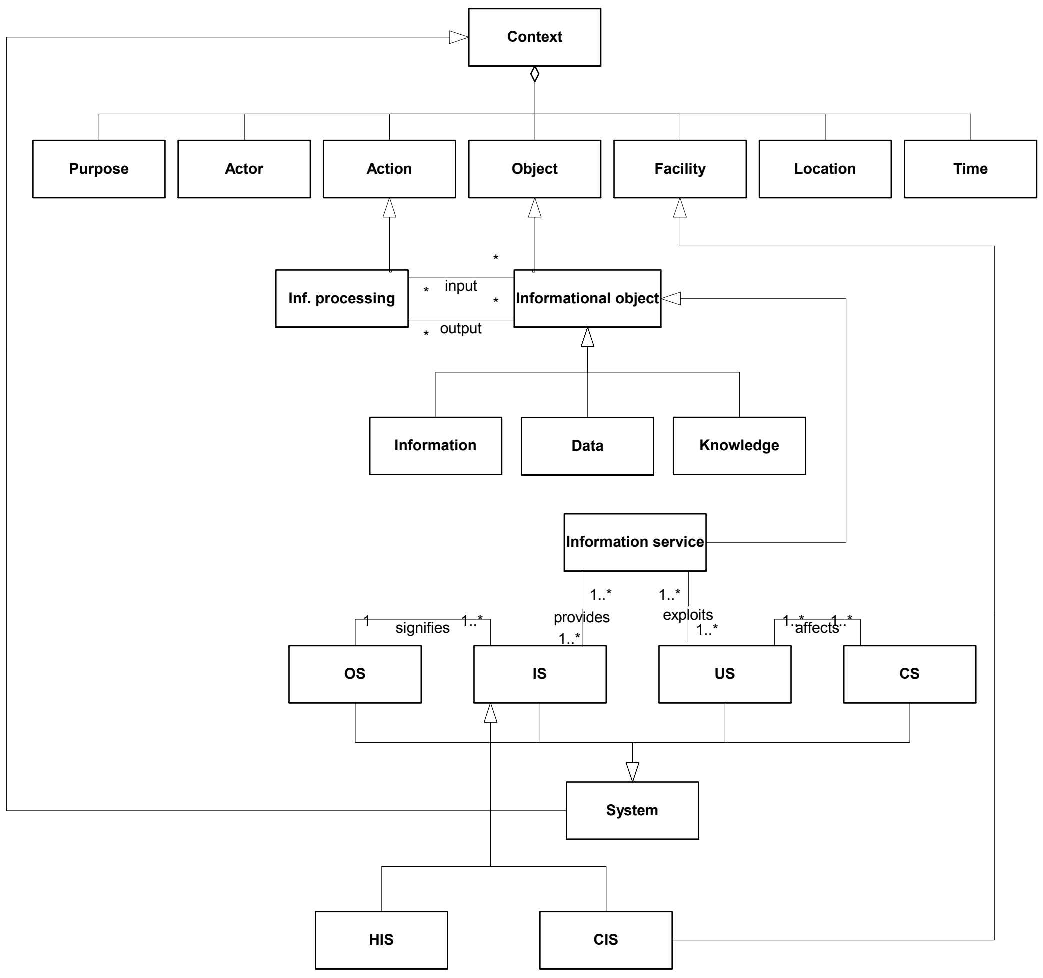


FIGURE 45 Meta model of information processing related concepts and relationships

Knowledge represented in a language is called *data* (Falkenberg *et al.* 1998, 66). *Information* is the knowledge increment brought about by receiving data, by observing reality, or by inner thought processes by which a person organizes, compares, assesses his/her knowledge (cf. Falkenberg *et al.* 1998, 68)[103]. Besides that, information is an increment into the body of knowledge, it is commonly assumed (hoped) to be usable and profitable (cf. correct, timely, etc.). Notice that also knowledge that increases the reliability of some conception, already possessed by a human being, is regarded as an increment, and thus as a piece of information.

Information processing means action(s) by which informational objects are collected, stored, processed, disseminated and interpreted. The informational objects can be in the linguistic or conceptual form (cf. Section 4.4.4). The generic

103 Our definition of information extends the one given in Falkenberg *et al.* (1998). Among the IS researchers conceptions about information differ greatly. For instance, Stamper (1992b; 1999) regards norms and attitudes as essential parts to information. Hirschheim *et al.* (1995, 14) contrast information with a speech act conveying intentions and argue that "items of information are meanings that are intended to influence people in some way".

notions used above to enumerate information processing actions can be further specialized into a large variety of more specific actions, such as discovering, procuring, interviewing, recording, maintaining, editing, transforming, translating, converting, modeling, ordering, eliminating, decomposing, integrating, deriving, abstracting, concretizing, reviewing, verifying, validating, etc. We will return to some of these in the next chapters when defining more specific ontologies.

We can distinguish between four kinds of contexts which are closely related to information processing: (a) those, which information is about, (b) those collecting, storing, processing and disseminating information, (c) those utilizing information, and (d) those, which are controlled and possibly changed on the basis of the disseminated and utilized information. In the IS field, these contexts are commonly discussed in terms of systems (e.g. Langefors *et al.* 1975; Welke 1977; van Griethuysen 1982; Essink 1988; Hirschheim *et al.* 1995). Therefore, we next define the generic notion of a system and then discuss these contexts as systems.

'System' comes from the Greek term 'Syn histanai' which means 'to put together'. A system, as conceived in the general system theory (cf. Klir 1969; Ackoff 1971; von Bertalanffy 1974), is defined as "a set of elements in interrelations among themselves and with the environment" (von Bertalanffy 1974, 17). The definition highlights the three most essential concepts in the systems theory: element, relation, and environment. An element itself can be a system, and so can an environment as well. A relation between elements stands for any structural, functional or behavioral relationship. Besides the elements, a system is characterized by so-called emergent predicates (or "systemic properties", Falkenberg *et al.* 1998, 60) that concern a system as a whole.

In our terminology, a *system* is defined to mean a conceptual construct through which phenomena in reality can be conceived as a whole (system), contained in the environment, characterized by emergent predicates, and composed of parts (elements). A system is a kind of system-theoretic abstraction from a context. It does not help reveal roles or functions the elements have in a system, unlike the notion of context does. In the following, we use the term 'system' in situations where it is commonplace. To emphasize the contextual nature of the UoD, we still signify it via the term 'context'.

Based on the above definitions, we can now use the following terms about the four kinds of contexts related to information processing: (a) the object system (OS), (b) the information system (IS), (c) the utilizing system (US), and (d) the controlled system (CS) (see Figure 45). In the following sub-sections, we discuss and define these notions. Moreover, we consider complicated relationships between the systems and the role that the controlled system plays in relation to the other systems. We also compare our notions to those presented in the literature.

5.1.2 Information System

Information system has remained a vague and controversial notion in the IS literature (cf. Avison *et al.* 1995b; Carvalho 1999; El-Sayed 1999). Traditionally, the information system has been regarded as a computer-based system, composed of hardware and software, storing, processing and transmitting formal data (e.g. Hicks 1993; Falkenberg *et al.* 1992a; Ein-Dor *et al.* 1993). Those favoring opposite conceptions emphasize the organizational and social nature of the information system embracing human information processing (e.g. Ahituv *et al.* 1990; Jayaratna 1994; Hirschheim *et al.* 1995; Stamper 1996; Franckson 1994; Falkenberg *et al.* 1998; Korpela *et al.* 2000, 198). Verrijn-Stuart (1989) and Verrijn-Stuart *et al.* (1992) distinguish between the notions of information system narrow and information system broad. The information system narrow (ISN) corresponds to the traditional conception covering "all the aspects of a computerized system […] allowing storage, updating, manipulation and retrieval of data" (ibid p.481). The system standing for the latter conception is known as the information system broad (ISB). It covers "all informational aspects of the organisational system, irrespectively of the availability of computerized support as such" (Verrijn-Stuart *et al.* 1992, 481).

Stamper (1996) extends the aforementioned dichotomy by presenting the "organizational onion" composed of three layers of information systems one within each other. At the outer layer there is the informal information system, in which most of the organized behavior is informal. People generate and interpret messages without conscious effort. They know what other people mean and what they intend without having to apply any explicit method of analysis. The next layer stands for the formal information system, which means organisational behavior that takes place according to formalised or structured rules. This kind of information processing is appropriate when tasks are performed repetitively and the workload is heavy. The inmost layer stands for the technical information system, in which all actions are automated and performed by a computer system. A prerequisite for automation is that the rules for behaviour can be completely formalized.

For our study it is enough to distinguish between two kinds of information systems that are the computerized information system and the human information system. The *computerized information system* (CIS) is a system in which all information processing is automated, that is to say, performed by one computer system or by several of them. The *human information system* (HIS) is a system, in which human actors play the only role in the accomplishment of actions to process information in a structured way. For the HIS, a CIS is just one tool among others used to enable human information processing, or make it more efficient and effective. When there is no need to make a clear distinction between the CIS and the HIS, we use the generic term 'information system' (IS), which stands for the HIS and/or the CIS.

The notion of the information system is commonly defined in terms of functions or attributes (Ein-Dor *et al.* 1978; Ein-Dor *et al.* 1993; Hirschheim *et al.* 1995, 11). In the former case a definition expresses what the system does. In the latter case the view of a definition is focused on the components the information system comprises. These ways of defining correspond to the functional view and the structural viewpoint, respectively. There is still another viewpoint, the teleological viewpoint (von Wright 1971), which addresses the purpose for which the system exists. In the following, we first give definitions of the IS from the structural, functional and teleological viewpoints[104], with references to those advocating the viewpoints concerned. After that we present the general definition integrating the viewpoints.

- *Structural viewpoint*. The IS consists of actors, actions, information/data, facilities (incl. software and hardware), and locations, constituting a cohesive information processing system, which serves organizational purposes or functions (cf. Davis *et al.* 1985; Kroenke *et al.* 1987; Hirschheim *et al.* 1995).
- *Functional viewpoint*. The IS is a functional unity, which collects, stores, processes, and disseminates information/data on the state of affairs in reality (cf. Buckingham *et al.* 1987, 18; Hirschheim *et al.* 1995, 11; Alter 1996, 2).
- *Teleological viewpoint*. The IS exists for providing high-quality information that is correct, relevant, timely, etc., in order to satisfy the needs of the users at a variety of organizational levels and the requirements of business actions they are engaged in. (cf. Aktas 1987; Olle *et al.* 1988a, 229; Parker 1989, 10; Iivari 1991, 250).

Hence, whereas the structural viewpoint reveals the elements the system is composed of, the functional viewpoint considers the actions of information processing and their outcomes. The teleological viewpoint emphasizes that the information system is not an end itself but that the reason for its existence is a set of services it provides for the utilizing system (cf. Nilsson 2000, 280).

Integrating the three viewpoints, we arrive at the following holistic definition of the information system: The *information system* is a system, composed of actors, information/data, facilities and locations, which collects, stores, processes and distributes information about the relevant parts of reality, called the object system, in order to enable and/or improve actions in the other context, called the utilizing system.

Information systems appear in practice with different functions, capabilities, performance and social consequences. They also differ in their components, inputs, outputs, and the support they can provide for the users. Ein-Dor and Segev (1993) identify seventeen major types of information

[104] These three viewpoints correspond, on a coarse level, to the main points of view in Kerola and Järvinen (1975, 15-18), known as the pragmatic point of view, the semantic point of view, and the constructive point of view.

systems and define them by vectors of their attributes and functions. Mentzas (1994) presents a functional taxonomy for classifying computer-based IS (CBIS) along three dimensions indicating the extent to which the systems support information processes, decision processes, and communication processes. Barron *et al.* (1999) present an analytical framework based on semiotics to help understand, classify and compare information systems of various generations. The framework consists of ten features that can be used to characterize the relationships between the IS and its users, and to represent and organize the system's contents. The features are: application domain, action complexity, social consequence, acquisition complexity, acquisition scope, input usability, output usability, justification, real world relationship, and representation.

Classifications reviewed above distinguish a large variety of information systems: e.g. management information systems, decision support systems, office information systems, executive information systems, expert systems, electronic meeting systems, group support systems, strategic information systems, computer aided manufacturing, etc. Classifications with their notions and terms are, however, "children of their time". Information system types evolve especially by accretion of technologies (Ein-dor *et al.* 1993, 185-6). The point at which a set of technologies is considered distinct and requires a new name is somewhat arbitrary, and largely a matter a convenience. Therefore, we argue that it is more beneficial to base a classification of IS's on the contextual domains. This can be done in two ways. On a general level, it is possible to attach an information system type to the domain that is the strongest determinant for the type (e.g. for a voice processing system it is the object domain). For more wide-ranging types of information systems it is possible to make an analysis to reveal their features with contextual concepts of more domains.

As we are not able here to have a comprehensive discussion on the issue, we content ourselves with outlining contextual IS classifications (the first way). In Table 15 information system types are attached to contextual domains, the special features of which are the major determinants for the types. For instance, integral to the sub-division into personal, groupware and organizational IS's is a number of users and a kind of their relationships in interactions. The action domain gives the basis to classify the IS's according to the actions that are the most essential in the IS's: processing (transaction processing system, process control system), querying (information retrieval system), analyzing (data mining system), etc. On the basis of objects processed by IS's, we can distinguish between document management systems (documents), multimedia IS (multimedia objects), relational database system (relations), object database system (objects), voice processing system (voices), geographical IS (spatial objects), etc.

5.1.3 Utilizing System

We define the *utilizing system* (US) to mean a system, which exploits information services, provided by the information system, in its decision

TABLE 15 Contextual IS classifications

Domain	**Types of information systems**
Purpose	Strategic, tactic vs. operational IS
Actor	Personal, groupware vs. organizational IS
Action	Transaction processing system, data mining system, information retrieval system, communication system, etc.
Object	Document management system, multimedia IS, relational database system, object database system, voice processing system, geographical IS, etc.
Facility	Manual, computer-aided vs. computerized IS One-tier, two-tier, three-tier vs. n-tier IS
Location	Centralized vs. distributed IS Mobile vs. locational IS
Time	Real-time vs. batch processing system Historical, recovery vs. temporal database system

making or operational actions, in order to make plans and execute changes (i.e. state transitions) in the controlled system. The *controlled system* is a system, which the utilizing system has control over. Actors in the US are users of the IS. *Information service* is a service that is composed of informational objects. A *user* of the IS is an actor who potentially increases his/her knowledge about some phenomena in the object system with the help of the IS (cf. Krogstie *et al.* 1996, 286). This also amends his/her abilities to fulfil the goals concerning the controlled system. We can distinguish between two kinds of users (cf. Krogstie *et al.* 1996)[105]. *End-users* increase their knowledge by interacting directly with the CIS. *Indirect users* increase their knowledge by getting results from the CIS through other users of the information system (cf. Krogstie *et al.* 1996, 286).

In the IS literature, various terms with different meanings are given to the US. Welke *et al.* (1982, 42), for instance, define a user system to mean "one or more individuals cooperating on the accomplishment of one or more functions in an organization". Olle *et al.* (1988a, 229) state that the "IS supports a business activity (or group of them) by providing the information it needs or by automating some or all of it". In Iivari (1989a, 327) the host organization "means the organizational context of an IS". Kaasboll *et al.* (1996, 113) define the notion of the application domain (of a computer system) to be composed of the users, the organisational context, and the work in which the computer system is used. Elements of the application domain include employees, the coordination of work, communication, power structures, ad-hoc organized work, interruptions in work, etc. One of the four worlds distinguished by the NATURE Team (1996) is the usage world, which "describes how systems are used to achieve work, including stakeholders who are system owners, indirect or direct users, and

[105] Cotterman *et al.* (1989, 1315) distinguish between the consumers and the producers/consumers.

their organizational context" (ibid p. 517). This world is regarded as the major source of user-defined goals and requirements.

The utilizing systems also can be classified according to various criteria. The US can be situated on the strategic, tactical or operational level. Correspondingly, the actors in it can be in the roles of executives, middle management or grass-root level workers. The primary products of the US actions may be material (e.g. automobiles) or informational (e.g. insurances). The US may function in local or global markets.

5.1.4 Object System

The *object system* (OS) means a system about which the IS, due to the interests of the US, collects, stores, processes and disseminates information (services) for the US. As implied from the definition, the boundary of the object system is totally determined by the interests of the US.

In the literature different terms are used to signify the object system. Iivari (1989a) defines the universe of discourse to mean something, which the information types of an IS refer to or imply to (ibid p. 327, 335). The NATURE project (NATURE Team 1996) uses the term 'subject world' to mean something that "contains knowledge of the real-world domain that the information system is intended to maintain information about" (ibid p. 517). Brinkkemper (1990, 23) see a universe of discourse to be "a system of concrete entities, which were, are or will be relevant with respect to a given objective".

About the object system there are also divergent conceptions. The object system is seen, for example, as the target of the ISD (i.e. the target system). This implies that the object system is part of the reality that is aimed to be changed through the ISD. Hirschheim *et al.* (1995) state that the ISD is "a change process taken with respect to object systems in a set of environments" (ibid p. 15)[106]. According to van Slooten *et al.* (1993) "the object system, or universe of discourse, is the part of reality considered as problem area for the development of an information system" (ibid p. 169). In these cases the object system is not considered from the viewpoint of the IS, as we do here, but from the perspective of the ISD. We shall return to these conceptions in Section 5.3 when discussing the notions of US and OS at different processing layers.

5.1.5 Relationships between the OS, IS, US and CS

The information system, the object system, the utilizing system, and the controlled system are inter-related in the way that is shown in Figure 45. Let us first consider the relationship between the information system and utilizing system. As stated above, the information system provides information services to its utilizing system. IS services can be supplied in two modes, which we call

[106] ISD as "a change process occurring over time with respect to an object system ..." was first defined by Welke (1982) and later by Lyytinen (1986, 74). Here we refer to Hirschheim *et al.* (1995) because it is more available than the two others.

the descriptive mode and the prescriptive mode (cf. representational and normative roles in Wieringa (1989, 33)). The information services in the descriptive mode means descriptions that are, to a large extent, "input data" that the information system collects, processes ('enriches') and finally supplies to the utilizing system. The information describes the states of affairs or events in the object system, and the users in the US can do with it as they like (e.g. an inventory system supplies information about quantity-on-hands, suppliers, prices, etc. of the products in the inventories). Another kind of information service supplied by the information system is in the mode of prescriptions. In recent years more and more business rules are included in the CIS to support the utilizing system. In the process industry, such as in a paper mill, the process is, to a high degree, under the control of a computerized system. Information about exceptions and malfunctions as well as "pre-programmed" rules of handling them are considered prescriptions for personnel in charge. Another example concerns an inventory system, which, triggered by pre-specified alarm limits, requires end-users to send supply orders to the concerned suppliers. In a small-scale, linear menu structures in user interfaces, often accompanied by wizards, guide and prescribe end-users to accomplish their work with the CIS in a step-by-step fashion.

Where does the line go between the IS and the US? An answer to the question depends on the adopted viewpoint, but it is also affected by the nature of the systems. We can distinguish between the following cases. If the US mainly works with material objects, making the difference between the US and the IS is easy. Second, if the actions of the US are mainly managerial and the IS is operative by its nature, the systems can be well separated. This kind of arrangement is described in Carvalho (1999) with three autonomous sub-systems: managerial sub-system (cf. US), informational sub-system (cf. IS), and operational sub-system (cf. OS & CS). An example of this kind is strategic decision making based on the information that is provided by an operational information system. The third case concerns a situation where the US is mainly an information-intensive context, such as an insurance company, a software house, or an architecture design office. In this case, it may be difficult to draw the line between the IS and the US. There may be the same persons conducting actions of the US and the IS. Some (informational) objects are dealt with both in the US and the IS, and actors conduct their actions in the same locations in both of the systems. The crucial issue to separating the systems is their purposes. The US aims at fulfilling its "business" goals and that is the only reason for inquiring and utilizing information from the IS. In case the IS is a CIS and there is no HIS actors neither HIS actions mediating information to the US actors, the IS and the US can be clearly separated on the basis of what a computer does and a human being does.

Next, we consider how the OS is related to the other systems. The most fundamental relationship is the signifies relationship between the IS and the OS, meaning that objects of the information system signify things in the OS. Also informational US objects signify things in the OS. The relationships between the OS and the other systems depend on whether the systems are overlapping or disjoint. We can distinguish between four different cases with regard to an extent to which the OS share parts of the other systems. In the first case, the OS is totally disjoint with the other systems. That means, for instance, that information is gathered from completely different things compared to those affected by the US. This is, of course, a very rare situation[107]. In the second case, the OS overlaps with the controlled system. For example, an inventory is a source of information that is used to control it. In the third case, the OS overlaps with the US. In this case the information system is used, for instance, to plan, control or develop work in the US (cf. project management system). Finally, the OS can overlap with the IS. For example, a web-based order system may contain a part, which records information about successful and failed issuing. In this case, the dividing line between the OS and the IS depends on how the IS is conceived: Is it regarded as a whole system, containing also the recording part, or a constellation of sub-systems, one of which is the (separate) recording sub-system?

5.1.6 Comparative Review

The notions of OS, IS, US, and CS are of vital importance to the understanding of information processing as an action and as a context. They are also key concepts in the IS field. Because there are quite divergent conceptions of the notions in the IS literature, we make a short comparative review of them in this section. We have collected a large set of references, ranging over three decades, and compared their basic concepts to our notions. A summary of the review is represented (in the temporal order) in Table 16. 'X' means that there is a considerable match with our notion. In the 'IS' column we show if the notion in the literature corresponds to CIS.

We start with Langefors and Sundgren (1975). They first outlined the notion of an object system as "a system we wish to inform about" (ibid p. 8). Later, they elaborated a characterization providing a narrow conception and a broad conception of an object system (ibid p. 209-210). An object system in a more restrictive sense, called the object system proper, consists of two parts: the observed object system and the controlled object system. The observed object system is a part of reality to which a database refers. The controlled object system means a slice of reality that is consciously affected by the decisions, taken by the users and based on the information objects of the database. An object system in a broader sense means the part of reality that has significance for the existence and functioning of the database (Langefors *et al.* 1975, 209). This notion embraces, besides the two parts already mentioned, also the database itself, data base administration, database designers and so one. We ignore this as being too large a conception.

---

107 For a weather forecast, information is collected from the object system (i.e. atmosphere), which is not affected, at least directly, by the utilizing system.

TABLE 16 Summary of the comparative review

Concepts	**OS**	**IS**	**US**	**CS**
Langefors *et al.* (1975)				
- observed object system	X			
- controlled object system				X
- information system		X		
Welke (1977)				
- data processing system		CIS		
- information system		IS		
- user sub-system			X	
van Griethuysen (1982)				
- UoD: abstraction system	X			
- UoD: object system	X			
- information system		CIS		
Olive (1983)				
- object system	X		X	X
- information system		X		
Essink (1986, 1988)				
- object system	X		X	
- information system		X		
Iivari (1989a)				
- universe of discourse	X			
- information system		CIS		
- host organization			X	
Hirschheim *et al.* (1995)				
- object system	X	X	X	X
- information system		X		
Kaasboll *et al.* (1996)				
- problem domain	X			X
- application domain			X	
- computer system		CIS		
NATURE team (1996)				
- subject world	X			
- system world		CIS		
- usage world			X	
Carvalho (1999)				
- operational sub-system	X			X
- informational sub-system		X		
- managerial sub-system			X	

Welke (1977, 149) argues that a system is a view of something (real or abstract) and that "something" is called an object system. There are two kinds of perceivers. The first class of perceivers is associated with the production of information. Their object system is called the data processing system (DPS). The second class of perceivers is associated with the use of the data (information). Their object system is some subset of organization and/or organizational environment, and is called the user-subsystem (USS). The information system, as an object system, is the intersection of the DPS with the USS.

According to van Griethuysen (1982), the universe of discourse (UoD) is that portion of the real world or postulated world that is being modeled. An abstraction system is that portion of the universe of discourse which includes the classes, rules, etc, of the UoD relevant from the viewpoint of the information base, and which changes relatively slowly. An object system is that part of the UoD not contained in the abstraction system (ibid p. 5). This means that an object system and an abstraction system correspond to our instance-level and type-level OS, respectively.

Olive (1983) argues that "an IS is always designed to give service to or exercise control over another: its object system" (ibid p. 66). When an IS is implemented it is embedded in the OS. That implies that the object system is actually regarded as the controlled system and the utilizing system whose functions should be defined to determine "what information the IS should provide".

Essink provides slightly inconsistent conceptions about the object system in his two articles (Essink 1986; Essink 1988). On one hand, the meaning of the object system is said to be threefold (Essink 1986, 58): (a) it depicts the desired contribution of the IS to the organizational processes, (b) it answers the question with regard to what phenomena in the real world information is needed, and (c) it defines the desired contribution of the goals of the organization and the demands the proposed IS lays upon the organization. On the other hand, the object system is said to be part of the organization that is considered to be the problem area for which a new system is desired. This part of reality is studied to acquire knowledge about the dynamic and static characteristics: about goal structures, environmental interaction, business processes, etc. (Essink 1988, 356).

Iivari (1989a, 237-238) presents three levels of abstraction: the host organization, the UoD, and the abstract technology of the IS. The host organization defines the organizational context of the IS. The UoD expresses the propositional/conceptual meaning of information processed by the IS. For the IS no explicit definition is given, but we here assume it to correspond to our notion of the CIS.

Based on Welke *et al.* (1982), Hirschheim *et al.* (1995, 15) define the notion of an object system from a broad viewpoint. An object system consists of any phenomena 'perceived' by members of an ISD development group. This implies that 'object' in this case is the target of the ISD, as can be seen from the definition given for an ISD: "a change process taken with respect to object systems in a set of environments…" (p. 15). Thus, the object system stands for the OS, the IS, the US and the CS in our terminology

Mathiassen *et al.* (2000, 6) argue that the system's context can be viewed from two complementary perspectives: the system models something (the problem domain) and it is operated by the users (the application domain). A problem domain means that part of a context that is administrated, monitored and controlled by a system. An application domain means the organization that administrates, monitors and controls a problem domain. Kaasboll *et al.* (1996,

113) extend the dichotomy in Mathiassen *et al.* (2000) by defining a computer system to include its application program, data/object base, user interface module, and communication modules.

Based on the suggestion originally presented in Mylopoulos *et al.* (1990, 340-342), the NATURE approach (NATURE Team 1996, 517) organizes the domain context (of what kinds of requirements exists) according to four worlds. Each of the worlds holds a family of related models with a different perspective on the ISD. The subject world "contains knowledge of the real-world domain that the information system is intended to maintain information about" (ibid p. 517). The usage world "describes how the systems are used to achieve work, including stakeholders who are system owners, indirect and direct users, and their organizational context" (ibid p. 517). The system world "contains descriptions of the technical entities, events, processes, etc. representing the system world in the required system" (ibid p. 517). The fourth world is the development world, which stands for the ISD.

Carvalho (1999) uses an arrangement of three sub-systems to classify information systems. The sub-systems are: operational, managerial and informational. The operational sub-system includes those activities that perform actions directly related to the system's purpose or mission. The managerial sub-system includes the activities that manage (organize, plan, control, coordinate etc.) the operational activities. The informational sub-system establishes communication among the other two sub-systems.

To conclude from the review, we can state that an object system is the most controversial concept (of those considered here) in the literature. On one hand, the term 'object system' means different things. In Olive (1983), for example, the OS stands for the US and the CS, and in Hirschheim *et al.* (1995) the OS means any slice of reality perceived by ISD stakeholders. On the other hand, the concept of the object system is signified with different terms. For instance, it is called the abstraction system & the object system (van Griethuysen 1982), the universe of discourse (Iivari 1989a) and the subject world (Mylopoulos *et al.* 1990, NATURE Team 1996). We argue that the notion of the object system is worthy of a special term and regardless of ambiguity related to the term 'object system' we prefer to use it here.

Most of the presentations deploy the general notion of the IS, but there are also those that use the more confined notion of the CIS. The utilizing system has been signified by various terms such as the user-subsystem (Welke 1977), the object system (Olive 1983; Essink 1986; Essink 1988), the host organization (Iivari 1989a), the application domain (Kaasbol *et al.* 1996), and the managerial sub-system (Carvalho 1999). We argue that the term 'utilizing system' expresses the most essential nature of the system, namely utilizing information services supplied by the information system. Only Langefors *et al.* (1975) provide a special term for the controlled system (controlled object system). Some others include the CS in larger contexts (e.g. Olive 1983, Hirschheim *et al.* 1995, Carvalho 1999). In our opinion, it is important to recognize the notion with a special term, although the CS shares parts of the IS and/or the US.

5.2 Information Processing Layers

Up till now we have considered information processing at a general level, on one hand, and in association with the notion of the IS, on the other hand. Besides being related to the IS, information processing is essential to many other kinds of efforts. In this section we distinguish between various processing layers. For each layer, the concepts defined in Section 5.1 apply, although adapted with layer-specific features. The overall structure of the layer ontology is presented in Figure 46. In what follows, we will discuss and define the concepts in the meta model. In Section 5.2.1 we distinguish between the primary actions and the development actions in information processing. In Section 5.2.2 we define, deriving from this dichotomy, the system of four processing layers. The layers are: information system, information system development, method engineering, and research work. Each layer is characterized from the teleological, functional and structural viewpoints. We also discuss the contents of and relationships between the contexts positioned at these layers.

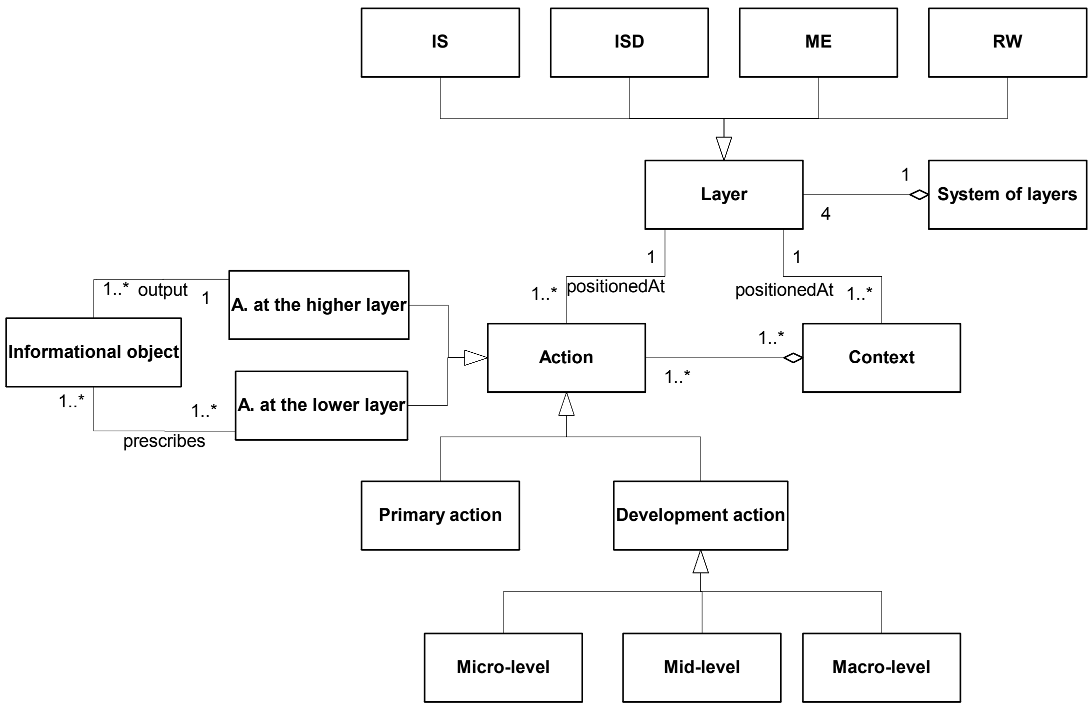


FIGURE 46 Meta model of layer-related concepts and relationships

5.2.1 Primary and Development Actions

Above we have considered information processing to be part of an IS context, which aims to provide the users in the US with high-quality information. Some part of information processing is carried out with random and opportunistic processes, and in unique and idiographic forms. But there are situations in which this way of working is not efficient, or even acceptable. For example, if the same kinds of situations recur, or processing requires considerable amount of resources, or there are strict demands on delivery times and quality of information, it is better to structure, at least some part of information processing, into a manageable and controllable process guided by predefined prescriptions. Prescriptions, as operational instructions or general guidelines, dictate who will do what, why, for which, when and/or where.

For practical work the prescriptions are never all-inclusive and complete, for several reasons. First, it is impossible to produce prescriptions that would apply to all the situations. Reality is just too multifarious to be "pre-programmed". The prescriptions have to be adapted in a contextual fashion. Second, reality changes and evolves all the time. The states of affairs having existed at time when the prescriptions were produced no longer exist when applying them. Organizations, persons, technologies, markets, etc. may have changed. The more there is task uncertainty, the more probable it is that exceptions to and deviations from the prescriptions are needed (Galbraith 1973). Hence, customization and re-specification of prescriptions occur frequently and in parallel to routine information processing. We call the routine-like information processing carried out according to the prescriptions the *primary actions* (cf. Gasser 1986). Respectively, making changes in routines of the primary actions is called the *development actions.*

Figure 47 illustrates the division of information processing into the primary actions and the development actions. The primary actions proceed in the form of daily routines. Every now and then, there is a need to deviate from the customary ways, or it is found out that there exist no guidelines for the situation at hand. It is also possible that guidelines and rules are on such a general level that they cannot be followed as such but are presumed to be specialized and/or instantiated. This situation corresponds to what Carroll (2004) calls "completing design in use". Due to this, before carrying out primary actions, it is necessary to decide on how to carry them out, that is to say, to develop a plan of action for them. In the figure the vertical dimension stands for the comprehensiveness of these development actions. The comprehensiveness can be expressed in terms of (a) duration of carrying out a development action, (b) resources (money, manpower, energy, etc.) needed for the work, (c) a number and quality of personnel involved in the work, and (d) the scope with which a development action affects the primary action(s).

For simplicity, we categorize the occurrences of development actions into three classes according to their comprehensiveness. First, there are small-scale tasks that are normally carried out in conjunction with daily routines by individuals themselves. These are called the *micro-level development.* Second, discussion groups or working groups of two or three persons are established, informally or formally, to consider how to deal with deviating or problematic situations. Solutions serve as new or renewed prescriptions for the primary action(s) from that point forward. These kinds of actions are called the *mid-level*

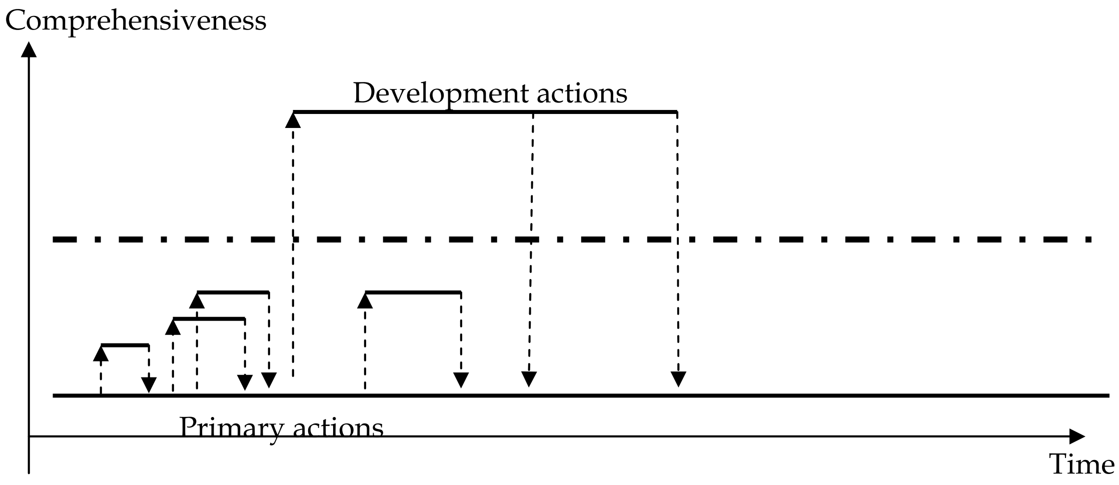


FIGURE 47 Information processing as primary and development actions

development. In some cases, it is seen necessary to deliberate more carefully, not only over ways of working, but also how to organize working, how to obtain benefits from applying new technology, how to better compete in the markets, etc. This calls for a pre-planned, controlled and coordinated development endeavor that involves several individuals with various skills, takes weeks or months, sometimes years, and may cost a lot of money. This kind of work is called the *macro-level development.* Due to its wide scope and complexity, development actions at the macro-level are commonly organized as a project work with pre-specified goals, organization, resources, and schedule.

The borderlines between the classes of the development actions are not clear-cut. For instance, informal projects can be established to carry out some development work in a couple of weeks with resources that may be decided on during the work. The agile approaches, for example, blur a dividing line between an ad'hoc –like action and project-like development (e.g. Agile Alliance 2002; Cockburn 2001; Astels *et al.* 2002). The approaches of evolving information systems (e.g. Falkenberg *et al.* 1992a; Jarke *et al.* 1992; Oei *et al.* 1994; Nguyen *et al.* 1996), in turn, enable to make "on-fly" changes in a current information processing. There are, however, certain factors, which add needs for a project-like working: (a) The scope of problems encountered in existing information processing is large; (b) The variety and profoundness of changes that will be caused by new technology adoption in an organization are estimated to be substantial; (c) Due to the specificity of problems, application area, applied technology etc., a large number of people with special skills are needed; (d) Acquirement (of hardware, software and "peopleware") required by planned changes are significant; (e) There is a definite need to reach the goals in time and with given resources. In the following we mainly consider the development action(s) that are accomplished in an organized project. That means those development actions that are placed above the broken line in Figure 47.

The division of information processing into two or three types of work is common in the IS literature. Checkland (1981) identifies two domains of inquiry in ISD: the problem system and the problem solving system. Iivari (1989a) distinguishes between conversion functions, corresponding to our notion of the primary action, and development and rearrangement functions. The latter involve "changes in the prescribed organizations, for instance reorganization concerning the authority relationships, reallocation of organization functions to organizational positions, etc" (ibid p. 333). Gasser (1986) distinguishes between the primary work addressing agendas of the work situation and the articulation work, which "serves to establish, maintain, or break the coordinated intersection of task chains in the primary work (ibid p. 211)[108]. Conradi *et al.* (1993) distinguish between software production processes and software meta-processes. The former carry out software production activities, and the latter improve and evolve the whole software process. The software meta-processes are in charge of several activities (e.g. process requirements analysis, process design, and process assessment). Nonaka (1994, 1995) presents an organizational model, the hypertext organization, for the organizational knowledge creation process. The hypertext organization supports organizational knowledge creation in all its stages and contexts. The model consists of three layers: business-system layer, project-system layer, and knowledge–based layer. The business-layer is the bureaucratic structure, which is responsible for performing the routine work. The project-system layer is the layer where project groups concentrate on knowledge creation and sharing through dialogues. The results of this layer are internalized and used in routine work. At the knowledge-base layer the knowledge created in the two layers is stored.

5.2.2 Processing Layers

In the previous section we considered the division of information processing into the primary actions and the development actions in conjunction with the IS. In that context the macro-level development is called information systems development (ISD). The dichotomy of the primary action and the development action can be recognized within the ISD, too (cf. Iivari 1989a, 333). Consequently, some part of ISD proceeds as routine-like actions[109]. That part forms the primary action of ISD. Every now and then there is a need to deviate from routines, resulting in that prescriptions given for ISD need to be customized and new ways of modeling, working, organizing, etc. have to be created. This 'development of IS development' appears as work at three

[108] The notion of articulation work in Gasser (1986) is based on the work of Strauss (1978).

[109] With this we do not want to argue that the ISD is routine work by its very nature. Usually it is far from it. However, from the perspective of the last four decades, we can say that one of the strongest trends has been the aim to collect and engineer a set of conventions, as some kind of the "best practices", to be disseminated to and shared in forthcoming ISD efforts. These conventions, in the form of a method, has been accepted as "norms" according to which the accomplishment of ISD has been tried to make more structured, efficient and effective.

different levels exactly like in the IS. Thus, at the micro-level, IS analysts and IS designers have daily to consider what is a practical and beneficial way of doing things. This corresponds to what Ciborra (1999) calls improvisation in which planned action is overlooked. Second, each phase in an ISD project begins with planning what tasks, models, and techniques of the selected method will be applied and in which way. This planning corresponds to the development action at the mid-level. Finally, before launching an ISD project it is necessary to select and customize, and if not available, construct prescriptions for the ISD work. These prescriptions are composed into a method, and this work to construct a method belongs to method engineering (ME). In some cases, method engineering is organized as a separate project with defined goals, given resources and schedule.

Method engineering, in turn, can be considered as the primary action prescribed by instructions, guidelines, etc. And as above, in parallel to this "routine work", special development actions are accomplished to revise, customize and construct prescriptions for ways of modeling, working, organizing, etc. in the ME. These development actions appear, also here, at three levels (i.e. the micro-level, the mid-level, and the macro-level) with the same kinds of meanings as in the ISD. At the macro-level, the development action means engineering of an ME method. This work is, to a large extent, based on conventions and "rules" of IS research, which, when taken broadly, is considered to be an investigation into the development, operation, use, evolution and impacts of information systems in organizations and society (Iivari 1991, 250).

We could still proceed upwards by considering the research work (RW) as the primary action and distinguishing special actions to revise, customize and construct prescriptions for the research work. This goes, however, beyond our scope in this study.

To summarize, by applying the dichotomy of primary action and development action repeatedly to information processing, we can distinguish systems at four layers: information system (IS), information system development (ISD), method engineering (ME) and research work (RW). Next, we define the concepts of a layer and a system of layers.

A *processing layer* is composed of those information processing actions, which share similar goals and the same target of action. A *system of layers* is a system that is composed of processing layers, which constitute a hierarchical structure, in which actions at a higher layer produce informational objects to be used as prescriptions in the actions at the next lower layer. Prescriptions can be expressed in various forms: as goals, guidelines, rules, commands, etc. Via the actions, also the contexts containing those actions can be positioned onto the processing layers. Positioning the contexts onto the layers is, however, much more complicated, as we show below.

Information processing actions of different layers engage in several kinds of interplay with one another. We can distinguish between the following main types of interplay (Figure 48):

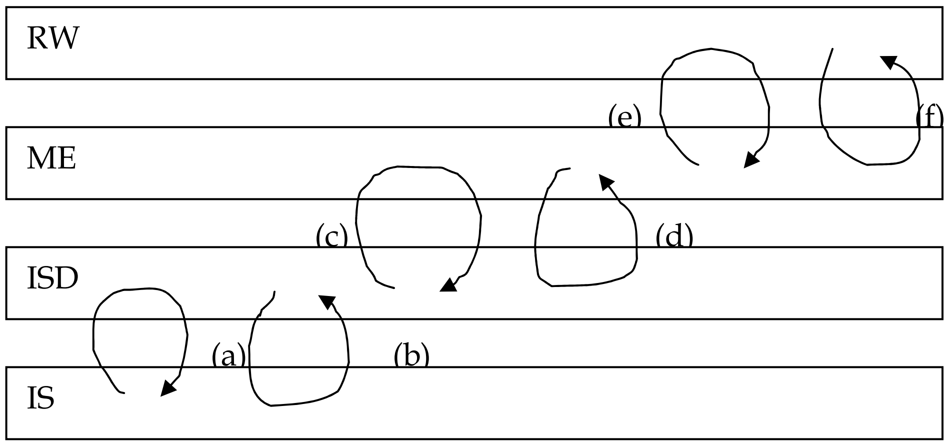


FIGURE 48 Interplay between IS actions, ISD actions, ME actions, and RW actions

(a) *Development during information processing in an IS*
During daily routines in an IS there appear frequently needs to deviate from the normal course of action to handle exceptional cases or other unexpected cases. This results in customization of prescriptions or negotiation about new ways of working in the areas that are not covered by the current prescriptions.

(b) ISD by experimentation
During ISD several kinds of experimentations are done to further the understanding of what it is all about and to test the functionality and acceptability of human and technical aspects of the designed system. That requires that some implementations are done and deployed during the development work. Implementations can involve only a small part of the information system (cf. prototypes), or cover major portion of the system (cf. pilot testing). For example, in the evolutionary approach an initial version of the system is delivered to intended users and it continues to be improved until it becomes the final system.

(c) Method adaptation during ISD
Information system development is most commonly carried out according to some ISD method. ISD contexts are, however, too unique for any method to provide a complete match with the needs. That is why, during the development of an information system some actions of method adaptation are constantly carried out. Changed practices may become a part of a renewed method if externalized and made available to other actors (cf. the evolutionary or incremental ME approach (Tolvanen 1998, 196).

(d) Method engineering by experimentation
A method is quite an abstract thing. In order to "prove" its applicability before the delivery, it is necessary to test it with some real cases, or in some pilot projects. Experience got from the usages can be deployed to better the method. Compared to the previous case, in which an ISD

method is “evolutionarily” changed in an ISD context, here ME actions pursue radical changes in ISD.

(e) *Method adaptation during method engineering*
Method engineering should also follow some ME method. Because ME contexts, much more than ISD contexts, are unique, no method can be employed as such without adaptations. Changed practices may become a part of a renewed ME method if externalized and made available to other actors.

(f) *Research work by experimentation*
Also during the research work, aiming at engineering an ME method, it is necessary to test the ME method under construction before its delivery.

In Figure 48 six types of interplay are labeled (from a to f) and depicted as circles starting from and ending to those layers, which correspond to the major aims and actions of the concerned efforts. Concluded from the above, we can say that in working at a certain layer there is always a need of accomplishing actions at the next lower layer as well as at the next higher layer. Hence, situations are far from what Orlikowski (1996) calls time-space disjuncture. Consequently, it is beneficial, and even necessary, to obtain experience from using the outcomes under construction in circumstances that to a sufficient degree correspond to real usage situations. Through these experimentations evidences of the applicability are collected for a basis for further work. On the other hand, there is always a need to improve ways of working at each layer, and this is accomplished by carrying out actions of the next higher layer. In conclusion, actions on a certain layer are contained in up to three kinds of contexts, and vice versa, contexts at a certain layer can contain three kinds of actions. Whether a context is regarded as an ISD context, or an ME context, for instance, depends on its main purpose. For instance, among the six types defined above there are two ISD contexts (b and c) aiming to develop an IS, and two ME contexts (d and e) aiming to engineer an ISD method. Figure 49 illustrates how the contexts and the actions at the information processing layers

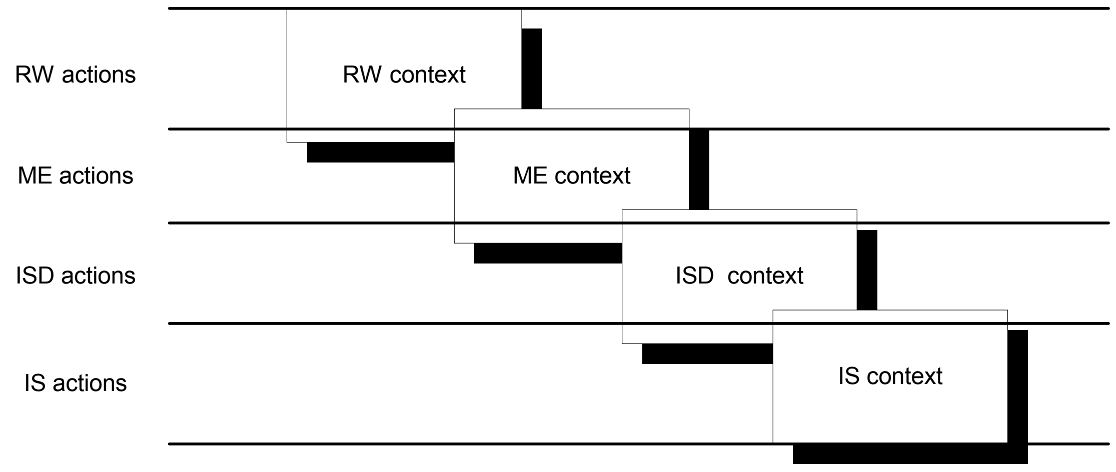


FIGURE 49 Actions and contexts at the processing layers

are related. The rectangles and the areas between the lines correspond to the contexts and the processing layers, respectively.

At the end of this section we characterize the contexts at the four processing layers from the teleological, functional and structural viewpoints (Table 17). As the characterizations show, the contexts are quite similar as to their purposes, functions, and contextual structures. We use this finding as a justification for treating the contexts as analogous to one another when defining the concepts and constructs for the corresponding ontologies (see Chapters 8 and 10).

5.3 US and OS at the Processing Layers

We defined the utilizing system (US) to mean a system that exploits information services, provided by the IS, in decision making or operational actions. The object system was defined to be part of reality about which the IS, due to the interests of the US, collects, stores, processes, and disseminates information to the US. The information can be in the form of descriptions or prescriptions. Although the definitions are aimed to be suitable as such for the systems at the bottom layer, they also suit a more general use. That is to say, we can assume that the ISD layer, instead of the IS layer, is the root layer and consider it to be a kind of IS context. Or alternatively, the root layer may be considered to be the ME layer. In this way, we can derive the definitions for the US and OS at each processing layer, with special features of course. To denote more clearly the layer from the viewpoint of which the US and the OS are considered, we use subscripts: e.g. US_{ISD} means the utilizing context of the ISD context. We start with discussing the US and then proceed to consider the OS at each layer.

At the bottom layer, the US is a business system for which the IS provides information about the OS. At the ISD layer, the ISD produces prescriptions for the next lower layer to facilitate the IS to satisfy the needs of the US_{IS}. At the ME layer, the ME produces prescriptions for the next lower layer to facilitate an ISD context to efficiently and effectively produce prescriptions for an IS so that it could satisfy the needs of the US_{IS}. Finally at the RW layer, the RW produces prescriptions for the next lower layer to facilitate an ME context to efficiently and effectively produce prescriptions for an ISD. The primary actions at each layer are guided by needs and constraints determined by the "utilizers", and the higher the layer is at which the primary actions are accomplished, the more layers the needs and constraints of the "utilizers" come from. Likewise, the higher the layer is, the broader the area is on which the effects of the primary actions focus. This multi-layer structure of the US's is illustrated in Figure 50.

TABLE 17 Contexts at four processing layers characterized from three viewpoints

Layers	Teleological	Functional	Structural
RW	Context that aims, through organizational and technical changes, at improvements in a ME, satisfying the needs of and possibilities in the ME and its US, in order to facilitate and/or improve ME actions.	Context that produces, through the processes of requirements engineering, analysis, design, implementation, and evaluation, a new or renewed ME method (and related CAME tool).	Context where, to satisfy the RW goals, RW actors carry out RW actions for RW deliverables by means of RW facilities in a certain spatio-temporal space.
ME	Context that aims, through organizational and technical changes, at improvements in an ISD, satisfying the needs of and possibilities in the ISD and its US, in order to facilitate and / or improve ISD actions.	Context that produces, through the processes of requirements engineering, analysis, design, implementation, and evaluation, a new or renewed ISD method (and related CASE tool).	Context where, to satisfy the ME goals, ME actors carry out ME actions for ME deliverables by means of ME facilities in a certain spatio-temporal space.
ISD	Context that aims, through organizational and technical changes, at improvements in an IS that satisfy the needs of and possibilities in the IS and its US, in order to facility and / or improve IS actions.	Context that produces, through the processes of requirements engineering, analysis, design, implementation, and evaluation, a new or renewed IS.	Context where, to satisfy the ISD goals, ISD actors carry out ISD actions for ISD deliverables by means of ISD facilities in a certain spatio-temporal space.
IS	Context that provides high-quality information that is correct, relevant, timely, etc. in order to satisfy the needs of the users at a variety of organizational levels, and the requirements of the business actions they are engaged in.	Context that collects, stores, processes, and disseminates information about the state of affairs in reality.	Context that consists of actors, actions, data, facilities (incl. hardware and software), and locations, forming a cohesive structure, which serves organizational purposes.

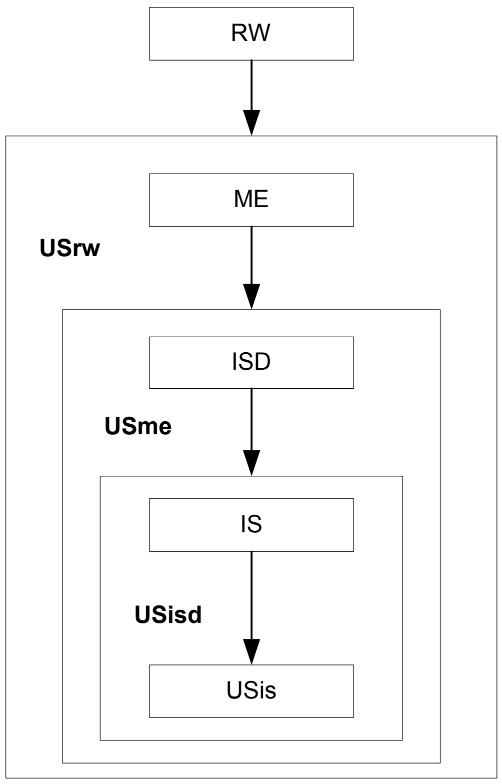


FIGURE 50 US's at four processing layers

In Figure 50 we can see that the utilizing system at the ISD layer (US_{ISD}) consists of the IS and the US_{IS}, and the utilizing system at the ME layer (US_{ME}) consists of an ISD, an IS and a US_{IS}. At the highest layer, the utilizing system (US_{RW}) comprises an ME, an ISD, an IS and its utilizing system (US_{IS}). For all the processing layers, the generic relationship 'provides information services to' holds. There are, however, differences in how direct the affects of exploiting services are in each case. For instance, an ME method resulted from the RW context affects indirectly on a US_{IS} through the following "chain": using an ME method, better ISD methods can be (hopefully) constructed, and by a better ISD method information systems can be developed that can support (hopefully) better the users in the US_{IS}. Due to this multi-layered nature of the US, for an ME context, for instance, requirements and needs should be collected from the concerned stakeholders at every layer.

Let us next consider the notion of OS at each layer. We defined the signifies relationship to stand for the relationship between the informational objects in the IS and the UoD constructs in the OS (cf. Section 4.4.4). At the IS layer, the OS is composed of all those things that are seen relevant to be informed about for the users in the business system (US_{IS}). At the ISD layer, informational objects signify the existing IS and a new IS, as well as their US's and OS's (i.e. US_{IS} and OS_{IS}). At the ME layer, informational objects signify the prior ISD contexts and the current ISD, as well as their US's and OS's (i.e. US_{ISD} and OS_{ISD}). Prior ISD contexts means that those ISD contexts in which the ISD method under consideration in the ME context have been deployed. Finally, at the RW layer, informational objects are created, processed, and disseminated which have signifies relationships with UoD constructs of the prior ME contexts and the current ME, as well as of their US's and OS's (i.e. US_{ME} and OS_{ME}). Prior ME contexts mean those ME contexts that have contributed to the creation and engineering of the method under engineering. This complex structure of the OS's is illustrated in Figure 51.

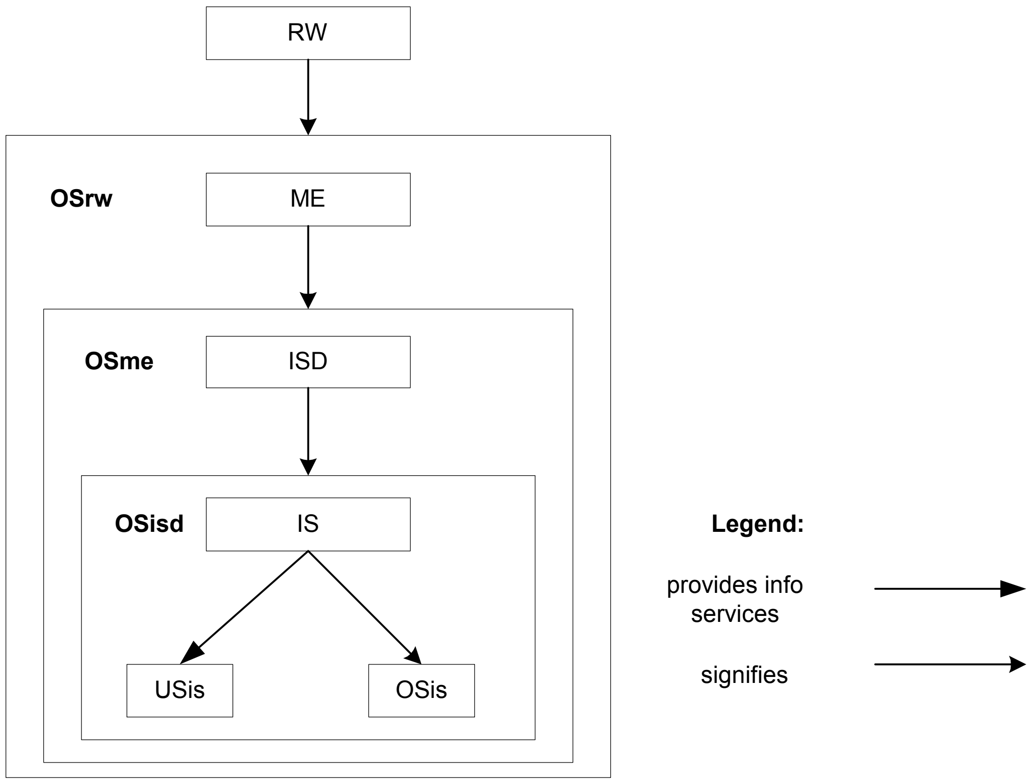


FIGURE 51 OS's at four processing layers

Due to the multi-layer structure of the OS, there is a large variety of the signifies relationships between the informational objects and the UoD constructs. At the bottom layer, the informational objects are concrete signifying e.g. individual actors, actions, and objects in the IS. At the higher layers, the informational objects also signify actors, actions, and objects in the IS but with a greater number of abstract concepts, that is to say, through meta concepts and/or meta meta concepts. Actually, during an ME effort and all the less during an RW effort it is not, perhaps, even known in which real ISD contexts the ISD method under construction will be deployed, not to speak of which instances of the IS will be developed on the basis of the ISD method. It is not until the ME method (and the ISD method) is instantiated, when more concrete concepts are parts of the informational objects.

The settlement in Figure 51 enables us to consider divergent conceptions of the object system presented in the IS literature. Most commonly the object system (or the corresponding term) means the OS_{IS} (e.g. Langefors *et al.* 1975;

van Griethuysen 1982; Iivari 1989a; NATURE Team 1996). In some presentations (e.g. Hirschheim *et al.* 1995; van Slooten *et al.* 1993) the object system is seen from the viewpoint of the ISD, meaning the OS_{ISD}. For instance, Hirschheim *et al.* (1995) state that the ISD is "a change process taken with respect to object systems in a set of environments" (ibid p. 15). According to van Slooten *et al.* (1993) "the object system, or universe of discourse, is the part of reality considered as problem area for the development of an information system" (ibid p. 169).

5.4 Summary

In this chapter we presented the layer ontology that has been specialized from the concepts and constructs of the context ontology defined in Chapter 4. The purpose of the layer ontology is to provide concepts and constructs to conceive, understand, structure, and represent static and dynamic features of information processing at four layers. The ontology is composed of two parts. The first part addresses information processing in general. It comprises concepts such as knowledge, data, information, information processing, information system, object system, utilizing system, and controlled system. The second part of the layer ontology provides concepts and constructs to establish a hierarchical system of processing layers. Four layers, called information system, information system development, method engineering, and research work, are distinguished and related. We defined the layers, discussed the contents of and relationships between the contexts on the layers, and considered how the notions of utilizing system and object system are understood on each of the layers.

The layer ontology is an important component in OntoFrame for two reasons. First, through the ontology, it is possible, for the first time in this study, to address issues of information processing that is a focal area in our research domain. Second, the ontology forms the foundation for vertical "structuration" of things in the UoD, distinguishing between actions (and contexts) in relation to IS, ISD, ME, and RW. The system of processing layers is one of the key dimensions in OntoFrame.

6 PERSPECTIVE ONTOLOGY AND IS PERSPECTIVES

Typical for a human being is his/her ability to “extract” just those features from reality that are most essential to the situation or problem at hand. With this capability, based on the use of viewpoints, he/she is able to conceive, handle and manage extremely complex and comprehensive situations. In everyday life viewpoints are established and applied in an intuitive and ad hoc fashion. In professional work, like in information system development and method engineering, there is a need for a more strict approach to establishing and applying viewpoints.

The purpose of this chapter is to first define the perspective ontology, which provides concepts and constructs for conceiving, understanding, structuring and representing things in reality from a set of pre-defined perspectives. The ontology is particularly aimed for organizational contexts, in which information processing plays a major role. The perspective ontology has been derived from the layer ontology and the context ontology (see Figure 52). The layer ontology provides the essential concepts and constructs for understanding and structuring information processing, in particular through the notions of information system, object system and utilizing system. It also serves as the conceptual foundation for structuring information processing at four layers (information system, information system development, method engineering, and research work). The context ontology contains detailed concepts and constructs of seven contextual domains and inter-domain relationships.

The second aim of this chapter is to present the IS perspectives. We define concepts and constructs with which the IS can be conceived from five different perspectives specialized from the perspective ontology. This part is included in this chapter for two reasons. First, we do not have a separate chapter for presenting the IS ontology, of which the IS perspectives constitute the major part. Second, defining the IS perspectives here gives a concrete example of how to utilize the perspective ontology by specialization.

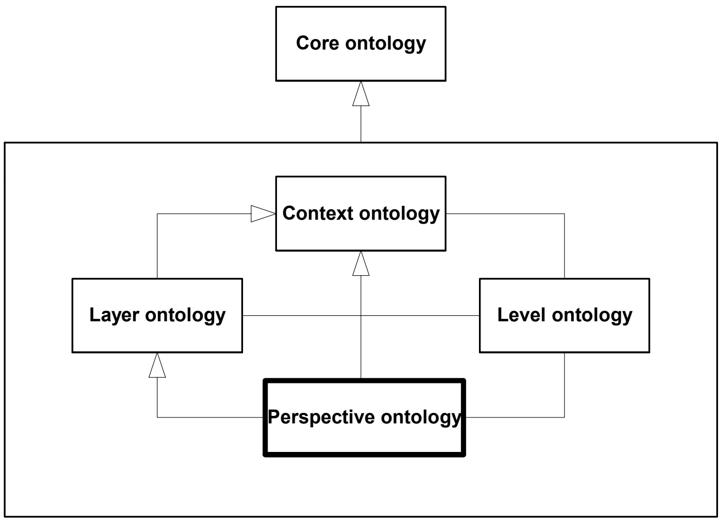


FIGURE 52 Focus of Chapter 6

The chapter is organized as follows. In Section 6.1 we define the perspective ontology, which establishes the system of perspectives along particular dimensions and specifies the contents of five perspectives. The perspectives are: systelogical, infological, conceptual, datalogical, and physical. In Section 6.2 we consider how these perspectives can be applied at the four processing layers. In Section 6.3 we define the IS perspectives, meaning that concepts and constructs of the IS from five perspectives are provided. We also specify the relationships between the IS perspectives. In Section 6.4 we present a comparative analysis of IS perspectives suggested in the IS/ISD literature. For the analysis we have selected eleven frameworks containing clearly defined perspectives. The analysis is composed of three parts, covering an overview, conceptual contents, and detailed concepts of the perspectives. Section 6.5 contains a summary and discussions.

6.1 Perspective Ontology

In this section we first define the general notions of a perspective and a system of perspectives. Second, we define five perspectives based on three particular dimensions.

6.1.1 System of Perspectives

Reality contains a myriad of details so that it goes fully beyond the capacity of any human being to recognize and conceive them all simultaneously. For this reason, it is typical for a human being to focus one's attention upon some specific things. The focus depends on the adopted point of view. In everyday life, a point of view can be situational and intuitive, established in an ad hoc fashion. But for recurrent situations it is necessary to have structured and predefined viewpoints. This holds especially for situations, like ISD and ME, where abstract thinking is commonplace and which involve a large number of people in cooperation. To differentiate the intuitive points of view from the predefined points of view we define the notion of a perspective as follows: a *perspective* is a strictly defined point of view[110].

Conceiving reality in a systematic way necessitates that there are more than one perspective available and the relationships between the perspectives are specified. A *system of perspectives* means a (static) system, which is composed of related perspectives. A system of perspective is the focal notion in the perspective ontology (see Figure 53). The *perspective ontology* provides concepts and constructs for conceiving, understanding, structuring and representing things in information processing contexts through a system of pre-defined perspectives.

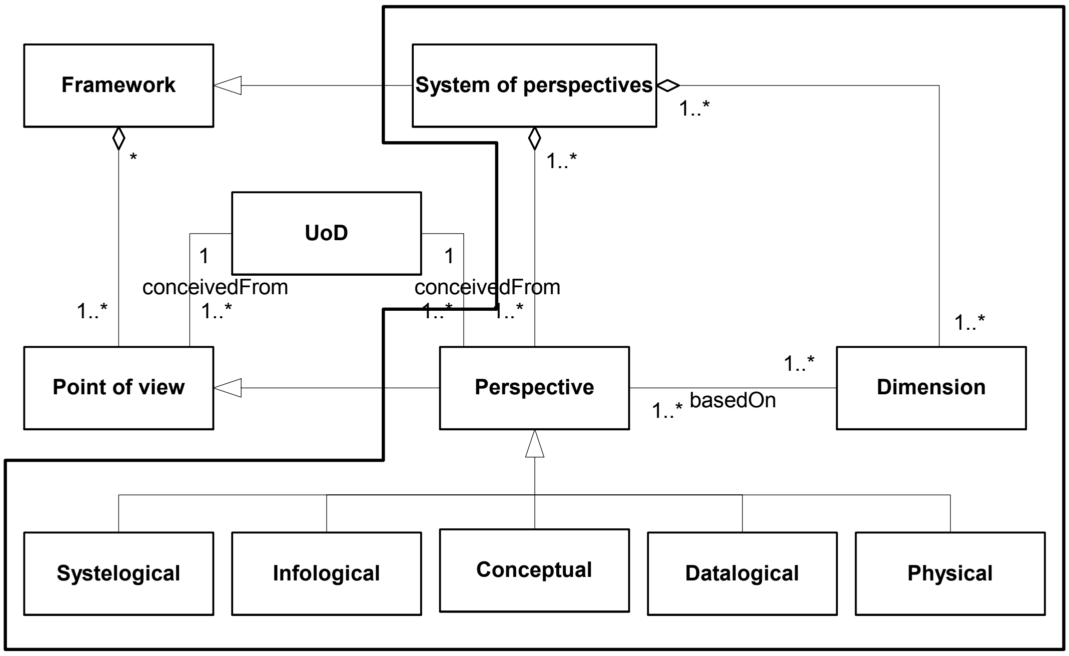


FIGURE 53 Perspective ontology

A system of perspectives is a strictly defined framework (cf. the notion of a framework in the generic ontology in Section 3.3). The relationships between the perspectives in the system can be based on one or more dimensions or criteria. An example of the systems of perspectives, which is based on one criterion, is levels of abstraction. Based on the systems theory, Mustonen (1978,

110 There are different meanings for perspectives in the literature. Mathiassen (1982), for instance, defines a perspective to be a conceptual abstraction of a view or specific phenomena. In Webster (1989) a perspective is defined to be "the faculty of seeing all the relevant data in a meaningful relationship", "the state of one's ideas, the facts known to one, etc., in having a meaningful interrelationship", and " a mental view or prospect".

53) defines the levels of abstraction, or the levels of stratification, on the basis of the semantic characteristics of the concept structures and imposes four characteristics for it: (a) The hierarchical relationship is linear. (b) The levels describe different predicates of the same system. (c) The relationship between the levels fulfills the condition: the upper level includes in some sense more abstract or holistic description of the system than the lower level. (d) The concept structure includes the definition of the relationships between levels (cf. Iivari 1989a, 325).

We set up the following goals for a system of perspectives needed in this work: (a) The perspectives have to support the structured consideration of multifaceted features of IS, ISD and ME. (b) Each perspective should be defined in a way that enables decisions on what aspects are relevant from that perspective and what should be ignored. (c) There should be well-defined relationships between the perspectives. Having these goals in mind, we can easily find out that a system of perspective based only on one criterion or one dimension is not suitable for our purpose. The reason for this is that it is not possible to find a single theory or principle that would cover all the desired features and provide the necessary concepts and constructs. For instance, systems theory or semiotics alone is too limited in its descriptive power.

We define a system of perspectives that is composed of three dimensions. The dimensions are: (a) decomposition dimension, (b) linguistic – conceptual dimension, and (c) realization independence – dependence dimension. The decomposition dimension is based, as suggested by its name, on the decomposition principle (see Section 3.9.2.3). Applied to the IS, this means that the IS can be viewed as part of the environment, in particular in relation to the US, or decomposed into information sub-systems and further into informational objects, IS actions, etc. This dimension is commonly applied in systems theories to make an imperceivable system more perceivable (Langefors 1971, 67)[111].

The second dimension in the system of perspectives is based on the semiotics (Peirce 1955). The dimension is 'dichotomic', having two ends, linguistic and conceptual. In the context of information systems this means that the IS can be viewed as a complicated whole of linguistic expressions and their transformations, or as conceptual constructs which the expressions signify. The third dimension is based on the predicate abstraction with the criterion of realization independence (see Section 3.9.3). It enables the partitioning of the features of the IS into predefined sets. At one end of this dimension the IS is viewed as being completely independent from any aspects of realization, while at the other end of the dimension one particularly concentrates on physical things in the realization, e.g. on physical actors, detailed procedures, concrete data files and documents in certain spatiotemporal space.

Figure 54 presents the perspectives in relation to US, IS, and OS, along the three dimensions. The dimensions are orthogonal to one another. In the

[111] An imperceivable system is "a system such that the number of its parts and their interrelations is so high that all its structure cannot be safely perceived or observed at one and the same time" (Langefors 1971, 67).

following section we define the perspectives and then return to comment on this figure.

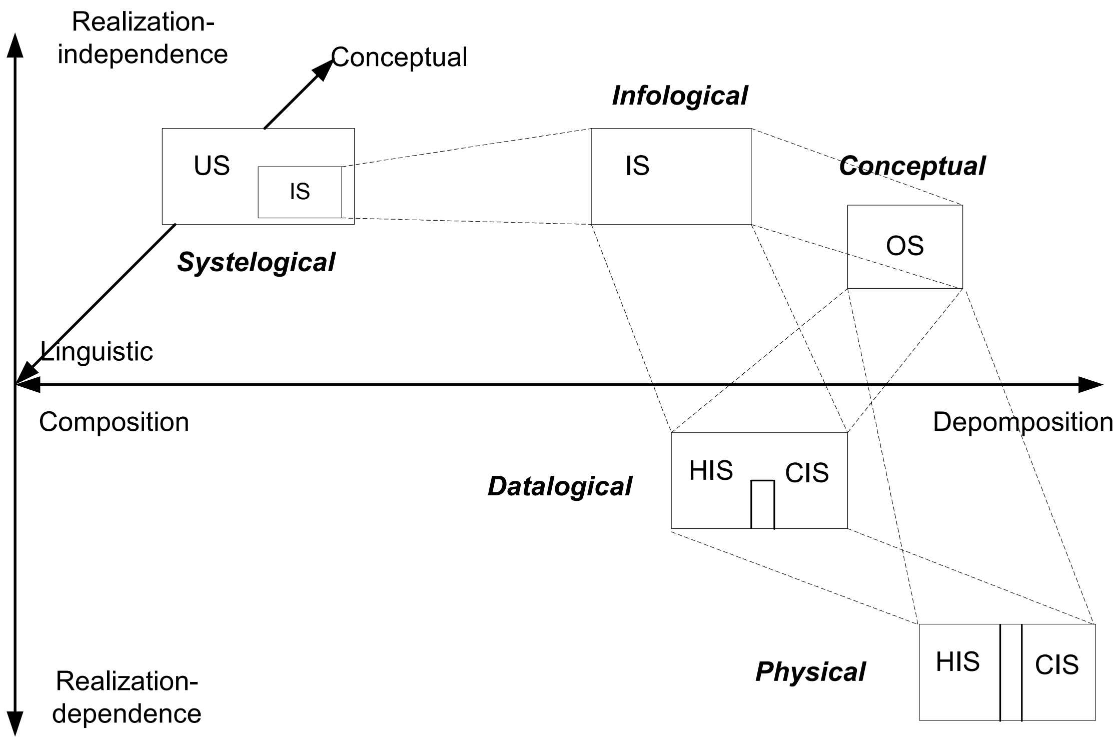


FIGURE 54 Dimensions and perspectives

6.1.2 Definitions of the Perspectives

The system of perspectives is composed of five perspectives. These are the systelogical perspective, the infological perspective, the conceptual perspective, the datalogical perspective, and the physical perspective. In the following we define them. Because defining the perspectives without connections to any target system would yield characterizations that are too generic, we have formulated the definitions that apply to the IS. The definitions are, however, also applicable to the other processing layers. We demonstrate that in Section 6.2 in discussing the perspectives of the other processing layers.

According to the *systelogical perspective* the IS is considered in relation to its utilizing system (US). The IS has no value or purpose by itself. It becomes desired and necessary through the support it provides to its utilizing system. Hence, the IS's organizational, social, economic and informational impacts on the utilizing system form the essence which the systelogical perspective is interested in. The generic question to be answered from this perspective is "Why". To put it more precisely, applying the systelogical perspective means considering the following issues:

- Why does the IS exist? For which utilizing system?
- What kind of utilizing system does it have? What are its objectives, actors, actions, events, rules and objects on a general level?

- What information services should the IS provide, for whom and for which actions in the US?

According to the *infological perspective* the IS is seen as a functional structure of information processing actions and informational objects, independent from any representational and implementational features. The IS is regarded as a context, in which the given mission is pursued by actions and information flows between them. The generic question to be answered is "What". This means in more detail:

- What information is processed in the IS and why?
- What are the actions and rules of processing?

According to the *conceptual perspective* the IS is considered through the semantic contents of information it processes. This means that whereas the infological perspective is based on linguistic terms, the conceptual perspective concentrates on the understanding of the meaning of those things in the object system which linguistic terms signify. The question to be answered is "What does it mean?" To put it more precisely, the conceptual perspective is interested in the following issues:

- What is the meaning of the information processed in the IS?
- What does the information signify?
- What kinds of structural and dynamic constraints are valid in the object system?

From the *datalogical perspective* the IS is viewed, through representation-specific concepts, as a context, in which IS actors work with IS facilities to process data. This implies that the perspective makes a separation between two parts: a human information system (HIS) and a computerized information system (CIS). Considerations cover all those contextual non-physical phenomena that are relevant to executing actions of data processing within and between those parts (cf. user interface). The datalogical perspective is interested in "How" questions such as:

- How is information represented in data in the IS?
- How are the rules of information processing derived from US rules and formulated into concrete work procedures and algorithms?
- How do the IS users and the CIS communicate with each other?

The *physical perspective* ties the datalogical concepts and constructs to a particular organizational and technical environment, showing how the IS looks like and behaves when it is implemented. It answers the following questions:

- Who are those actors carrying out actions of the HIS, how and when they act, and where are they located?
- Where and how are the data stored?
- How are the facilities used and by whom?
- What hardware and software are used, and how are they related?

After having defined the perspectives we will next discuss the selected terms and their counterparts in the literature, as well as the relationships between the perspectives and the dimensions.

The term 'systelogical' was introduced in Welke (1977), although in a slightly different meaning, to stand for the perspective of how "the changes to the existing information system alter/facilitate changes in the affected object systems (i.e. user-subsystem and data processing system)" (ibid p. 150). The terms 'infological' and 'datalogical' were first coined in Sungren (1975) and Langefors *et al.* (1975)[112] in the same meanings as we use them here. On the basis of the so-called infological approach by Langefors (1971) the system from the datalogical perspective should be called a data system. Despite the importance of making the difference between 'data' and 'information', we here follow a more common approach to deploy only one term 'information system'[113].

We can make the following remarks on the relationships between the perspectives and the dimensions (see Figure 54). The systelogical perspective provides the point of departure for considerations about the IS. The main focus of the perspective is on the US, and the IS is viewed as something which provides support for its US. Changing the perspective from systelogical to infological means a shift along the decomposition dimension: the IS seen as a "black box" is now conceived as a system that is composed of IS purposes, IS actions and IS objects. Compared to the infological perspective, the datalogical perspective and the physical perspective mean shifts along two dimensions, along the decomposition dimension on one hand and along the realization independence – dependence dimension on the other hand. The IS purposes, the IS actions, and the IS objects are, in the first stage, decomposed into smaller "pieces". In addition, IS actors and IS facilities are, on a general level, recognized. In the second stage, the process of decomposing continues and more and more realization-related aspects of the IS and its components are observed. The three perspectives (i.e. the infological, datalogical, and physical perspectives) constitute a "hierarchical system of stratified levels" as defined by Mustonen (1978). The conceptual perspective is based on the use of the semiotic dimension. While all other perspectives consider linguistic objects, the conceptual perspective focuses on their conceptual contents. This actually means that the focus shifts from the IS to its OS.

112 Langefors and Sundgren (1975, 3) mention a synonym for 'infological' that is 'informatological'.

113 In some literature, the use of the terms 'information system' and 'data system' does not depend on the perspective. Krogstie (1995, 479), for example, defines an information system as "a system for the dissemination of data between persons", and a data system as "a system to preserve, transform, and transport data". From this viewpoint, a data system is seen as a sub-subsystem of an information system.

6.2 Perspectives at the Processing Layers

Although the perspectives were defined above, for reasons of concreteness, in relation to the IS, they are aimed to apply to any processing layer. To show what the position of the perspective ontology in the overall arrangement of other ontologies is we present Figure 55. The figure portrays six ontologies related to one another. The IS ontology, the ISD ontology and the ME ontology consist of two parts: a domain part and a perspective part. The baseline for deriving (by specialization) the domain parts is provided by the context ontology. The perspective ontology serves as the foundation for deriving the perspective parts (by specialization) on each of the three processing layers, structured by the layer ontology. It should be noticed that we have not provided explicit definitions for the IS domains (in the IS ontology), because the concepts and relationships in the IS domains can be derived, in quite a straightforward manner, from the corresponding domains of the context ontology. For instance, an IS actor in the IS ontology is an actor in the context ontology with some specific features. In contrast, we do specify the IS

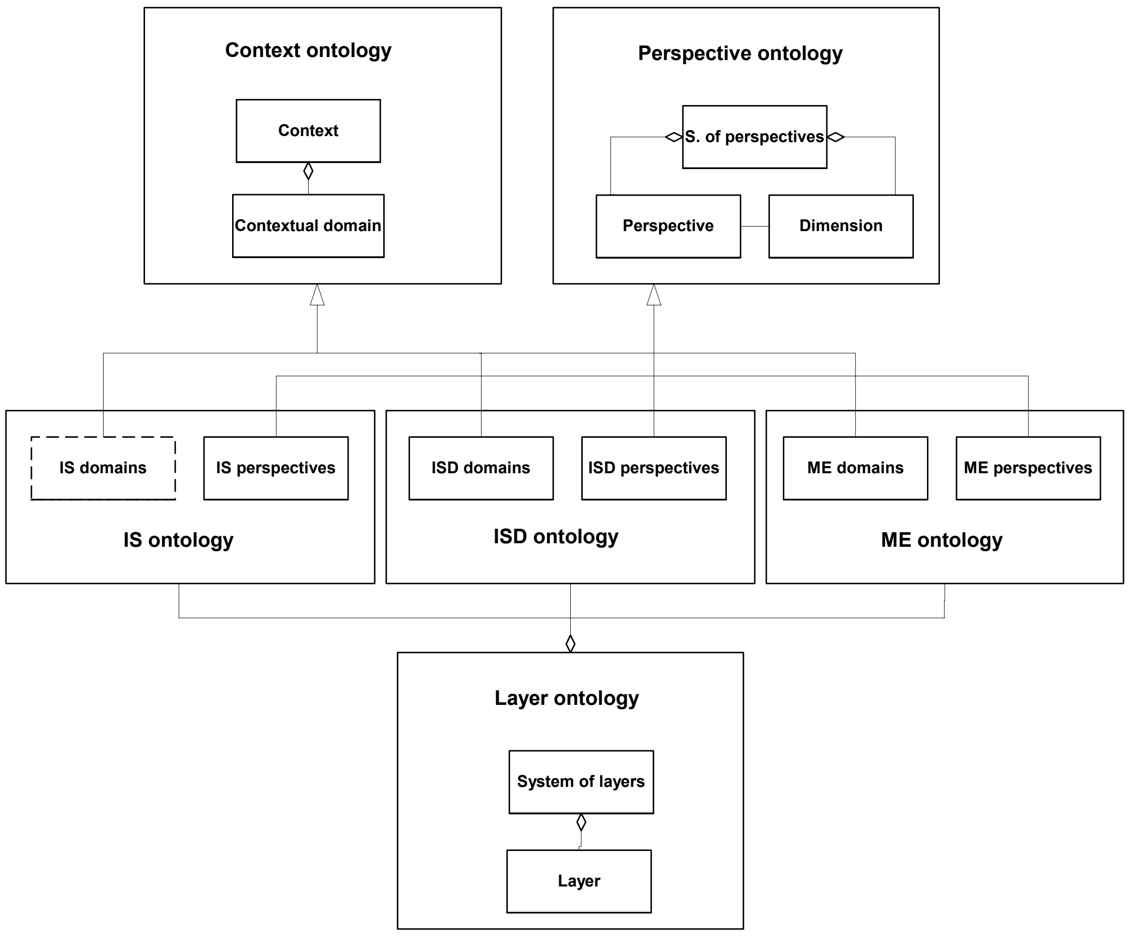


FIGURE 55 Perspective ontology in the overall settlement

perspectives (see Section 6.3). The perspectives for ISD and ME will be elaborated in Chapters 8 and 10, respectively.

Next, we give short characterizations of the perspectives on four processing layers. The RW layer is included in the considerations although we are not going to define the RW ontology. The RW perspectives are needed in specifying the methodical skeleton (MEMES) in Chapter 11. To denote which layers the perspectives concern we use the abbreviations of the names of the layers as prefix (e.g. the ISD infological perspective). The summary of the characterizations is presented in Table 18.

From the systelogical perspective the perceived system is considered in relation to its utilization system. On the ISD layer, this means that the perspective addresses the support ISD provides to the IS and US_{IS}. Relevant questions to be answered are, e.g.: what kind of IS is it for which the ISD project is or was launched, what kind of US_{IS} is it that should be supported with information services, what kinds of services should the ISD provide to US_{ISD}, and what are the goals and constraints for the approaches and principles of the ISD context? Implied from the above, we can state that the stakeholders applying the ISD systelogical perspective are the US_{IS} actors and the IS actors. Different ISD approaches can be distinguished based on whose role in ISD is emphasized - the US actors (cf. the client-led approach (Stowell 1991)) or the IS actors (cf. the participatory approach (Mumford 1981; Mumford 1983)).

The ME systelogical perspective reveals the support ME provides for its utilizing system (US_{ME}) comprising the ISD and the US_{ISD}. The perspective focuses one's attention to, e.g. what are the ISD contexts for which an ISD method should be engineered like, what are the IS contexts for which ISD projects are to be launched like, what are the US_{IS} which the IS's should support with services like, what are the "services" the ME should provide for US_{ME}, and what are the goals and constraints for the approaches and principles of the ME context? Compared to the ISD layer, here the set of real and potential stakeholders is much larger, including the US actors and the IS actors of several IS's, as well as the ISD actors of perhaps several ISD contexts. It depends on a situation and on how the views of each stakeholder group are taken into account.

From the RW systelogical perspective one considers the support RW provides for its utilizing system (US_{RW}) comprising contexts from the ME layer down to the IS layer. Due to the multi-layer structure of the utilization system (see Figure 51), the perspective concerns a large number of issues, e.g. what kinds of ME contexts are there for which an ME method should be engineered, what kinds of ISD contexts exist for which method engineering should be accomplished, what kinds of IS's can be found for which ISD projects are to be launched, what kinds of US_{IS} are there which the IS's should support with services, what kinds of "services" the ME should provide to US_{ME}, and what are the goals and constraints for the approaches and principles of the RW context? In addition to the actors mentioned above, the RW systelogical perspective involves ME actors as well. It should, however, be noted that the sayings of the

TABLE 18 Perspectives at four processing layers

Layers	Systelogical	Infological	Conceptual	Datalogical	Physical
RW	Considers what required / materialized benefits and outputs the RW provides to the US_{RW} (i.e. the ME's, the ISD's, the IS's and US_{IS}).	Considers functional structures and informational objects in the RW.	Considers the semantic contents of informational objects in the RW, in other words the OS_{RW} (i.e. the ME, the ISD, the IS and OS_{IS}).	Considers actors, actions, objects, and facilities as well as their interplay on a general level in the RW.	Considers the RW as a physical and technical construct within organizational and technical infrastructure.
ME	Considers what required / materialized benefits and outputs the ME provides to the US_{ME} (i.e. the ISD's, the IS's and US_{IS}).	Considers functional structures and informational objects in the ME.	Considers the semantic contents of informational objects in the ME, in other words the OS_{ME} (i.e. the ISD, the IS and OS_{IS}).	Considers actors, actions, objects, and facilities as well as their interplay on a general level in the ME.	Considers the ME as a physical and technical construct within organizational and technical infrastructure.
ISD	Considers what required / materialized benefits and outputs the ISD provides to the US_{ISD} (i.e. the IS and US_{IS}).	Considers functional structures and informational objects in the ISD.	Considers the semantic contents of informational objects in the ISD, in other words the OS_{ISD} (i.e. the IS and OS_{IS}).	Considers actors, actions, objects, and facilities as well as their interplay on a general level in the ISD.	Considers the ISD as a physical and technical construct within organizational and technical infrastructure.
IS	Considers what required / materialized benefits and outputs the IS provides to the US_{IS}.	Considers functional structures and informational objects in the IS.	Considers the semantic contents of informational objects in the IS, in other words the OS_{IS}.	Considers actors, actions, objects, and facilities as well as their interplay on a general level in the IS.	Considers the IS as a physical and technical construct within organizational and technical infrastructure.

US_{IS} actors and the IS actors have a marginal effect on the views and decisions made from the RW systelogical perspective.

From the infological perspective the perceived system is considered to be a functional structure of information processing purposes, actions and objects, independent from any representational and implementation features. Because only the perceived system (i.e. IS, ISD, ME, or RW) is relevant in considerations, the interpretation of the perspective is quite straightforward. The following issues are relevant in our considerations: what information is processed and why in the ISD context / in the ME context / in the RW context, and what actions and rules, on a general level, are needed for processing in the perceived context?

The conceptual perspective considers the perceived system through the semantic contents of informational objects, meaning that the perspective addresses the OS of the context at a layer. As shown in the previous section, the OS is very large and multifaceted at the higher processing layers. In the following we provide an overview of the OS's at each layer. A more detailed picture is built up in Chapter 7, where the model levels are integrated into the discussion. At the ISD layer the informational objects[114] refer to the (possibly) existing IS, the new IS as well as their US_{IS} and OS_{IS}. At the ME layer, the informational objects signify, besides those signified at the lower layer, also the prior ISD contexts and the current or planned ISD context(s). At the RW layer, the informational objects signify, besides those referred to at the lower layers, also the prior ME contexts and the current or planned ME context(s). In all those cases it is considered what the meaning of the information processed is, what information signifies, and what kinds of structural and dynamic constraints are valid in the OS.

From the datalogical perspective the perceived system is considered through representation-specific concepts, involving, besides the purposes, the actions and the objects, also the IS actors and the IS facilities, on a general level. Because also here the central focus is on the IS only, applying the perspective at each layer is straightforward. The following issues are relevant: How information is represented in data in the ISD context / in the ME context / in the RW context? What are the rules of information processing derived from the US rules, and how are they formulated into work procedures and algorithms in the ISD context / in the ME context / in the RW context? How do the actors and the computer-aided tools (CASE / CAME / CARW) communicate with each other in the ISD context / in the ME context / in the RW context?

From the physical perspective the perceived system is tied together with a concrete organizational and technical context. Examples of the issues covered by the perspective are: Who are the actors carrying out the actions, how do they act, and where are they located in the ISD context / in the ME context / in the RW context? Where and how are the data stored in the ISD context / in the ME

114 Here and also at the higher layers we only consider those informational objects that result from the execution actions, not from the management actions.

context / in the RW context? What hardware and software are used and how are they related in the ISD context / in the ME context / in the RW context?

6.3 IS perspectives

In this section we define concepts and constructs with which the IS can be perceived from the IS systelogical, the IS infological, the IS conceptual, the IS datalogical, and the IS physical perspectives, the emphasis being on first three perspectives. The IS perspectives are defined in this chapter for two reasons. First, we have no separate chapter for presenting the IS ontology, in which the IS perspectives could also be discussed, as it is done for ISD (Chapter 8) and for ME (Chapter 10). Second, defining the IS perspectives here gives a concrete example of how to specialize the perspective ontology (see Figure 55).

6.3.1 IS Systelogical Perspective

From the *IS systelogical perspective* the IS is seen in relation to its utilizing system (US_{IS}). The utilizing system is a business system, such as a manufacturing department producing machines ordered by customers, or a library accumulating and lending copies of publications to registered customers.

There are several approaches to viewing the utilizing system. It can be seen as an enterprise (e.g. Loucopoulos *et al.* 1998; Kavakli *et al.* 1999), a business process (e.g. Phalp 1998; Melao *et al.* 2000; Mentzas *et al.* 2001), or as a communicating organization (e.g. Dietz 2003). Each approach applies different concepts and constructs to conceive and structure things in the utilizing system. Dietz (2003), for instance, states that "an organization consists of people who, while communicating, enter into and comply with commitments (social interaction) about the things they bring about in reality" (ibid p. 148). This so-called PSI approach (Performance in Social Interaction) differs, to a substantial degree, from another approach, called the IPO (Input-Process-Output) approach, which is commonly applied in enterprise modeling, business process modeling and workflow modeling. Besides these approaches, there are different views specified on more detailed levels. Melao *et al.* (2000), for instance, identify four perspectives on business processes: business processes as deterministic machines, as complex dynamic systems, as interacting feedback loops, and as social constructs. In addition, there are presentations in which some specific issues are emphasized. Koubarakis *et al.* (2002), for instance, present a business goal oriented framework and Herbst (1995) suggests a business rule oriented framework. In this work it is not possible to cover all the approaches and views. Our approach of viewing the utilizing system integrates the IPO approach with the main concepts of the purpose domain and the actor domain.

Depending on the nature of the IS, we have two somewhat different viewpoints on the US_{IS}. If the IS is a CIS, the IS is seen as a tool used in the US_{IS}.

If the IS contains a HIS as well, the IS is seen as a related context providing information services to the US_{IS}. In the following we first present a meta model of the IS systelogical perspective from the tool viewpoint and then from the service viewpoint. In both cases, we exploit the concepts and constructs defined in the context ontology (Chapter 4).

Figure 56 presents the meta model of the IS systelogical perspective from the tool viewpoint. A *US organization* is an organization (i.e. enterprise, a department or some other administrative arrangement), which utilizes, or is going to utilize, an IS. It consists of *US organizational units*, which in turn are composed of US positions. A *US position* is a post of employment occupied by one or more *US human actors.* US positions are composed of *US roles* with responsibilities and authorities to conduct certain US actions. One of the US roles is a user. A *US action* is an action, which strives for one or more utilization purposes. Some US roles are related to the use of a CIS (cf. users in Section 5.1.3). US actions are governed by *US rules.* The rules are composed of three or four parts (cf. ECAA structure): *US event, US condition, thenUSAction* and *elseUSAction.* Conducting US actions may raise new US events that may trigger other US actions.

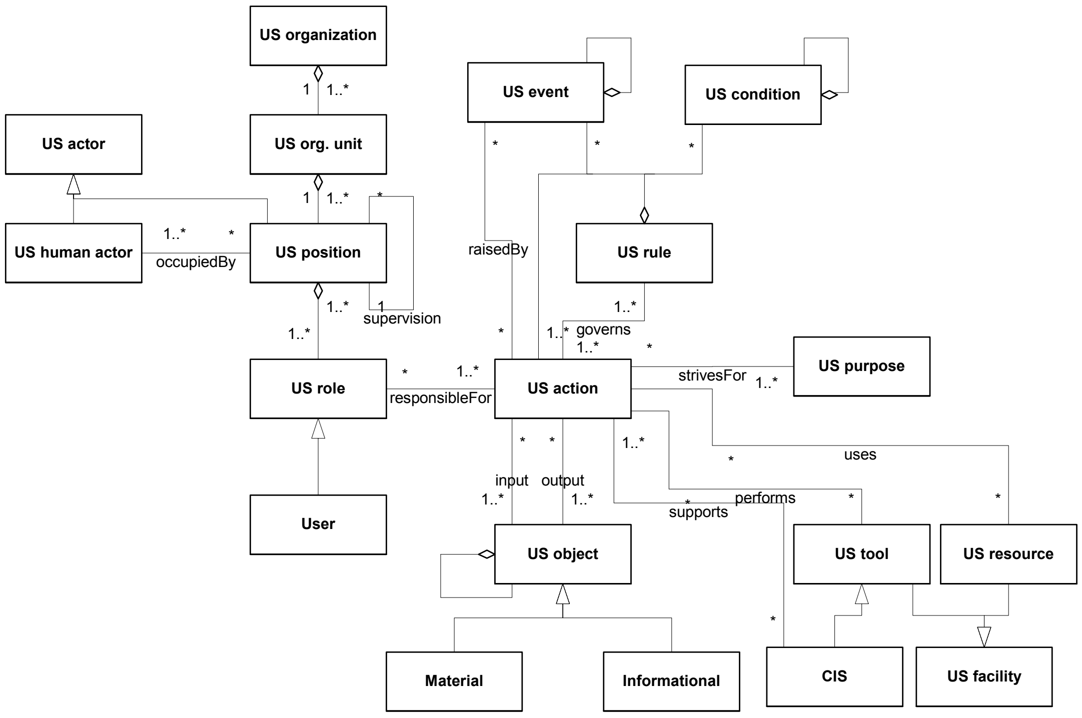


FIGURE 56 Meta model of the IS systelogical perspective (the tool viewpoint)

The *US purposes* mean goals for business processes and/or reasons for setting up those goals. The US actions use *US objects* as their inputs and may produce US objects as their outputs. The US objects can be *material* (e.g. machines, components, bridges, china, etc.) or *informational* (e.g. insurance contract, payment, reorder, etc.). The US actions are partly performed by *US tools* (e.g.

lathe, circular saw, nailer). Some of the US tools can be computerized information systems (CIS) supporting US actions. A US actor conducting US actions with the support of a CIS is called a user. The US actions consume *US resources,* like money, energy, goods, manpower, etc.

Figure 57 presents, on a rough level, the meta model of the IS systelogical perspective from the service viewpoint (cf. Figure 45 in Chapter 5). According to it the IS provides IS services to be exploited by the US_{IS}. The relationship between the US_{IS} and the IS can be elaborated in many ways: (a) by decomposing the IS services into informational objects we can state more clearly which kinds of services are provided, (b) by recognizing US purposes, US actions and US actors in the US_{IS} we can state more explicitly, who needs/uses, in which actions and for what purposes services from the IS, (c) by recognizing IS purposes, IS actions and IS objects in the IS we can reveal in which way various parts of IS services are produced in the IS. This process of decomposition and recognition leads to the arrangements of two interacting contexts, between which there are a multitude of relationships, not only between the actions, but also between the purposes, the actors, the objects, the facilities and the locations. This goes, however, far beyond the scope of the IS systelogical perspective, which should, by definition, treat the IS as a kind of "black box" in an organizational context.

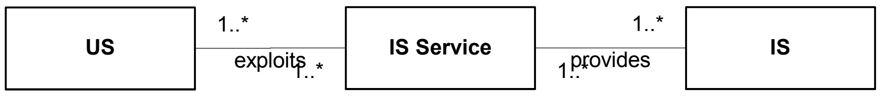


FIGURE 57 A rough meta model of the IS systelogical perspective (the service viewpoint)

6.3.2 IS Infological Perspective

In the *IS infological perspective* the focus is on the IS, which is seen as a functional structure of information processing and informational objects[115]. No attention is paid to how the objects are represented or implemented. This means that the "black box" conceived from the IS systelogical perspective is "opened" to reveal the aspects of the IS within three contextual domains: purpose, action, and object. The concepts in the purpose domain are used to specify why information is processed. The concepts in the action domain are used to conceive functional structures needed to produce informational objects. Correspondingly, the informational objects are decomposed, classified, and structured with the concepts and relationships in the object domain. The meta model of the IS infological perspective is presented in Figure 58. Next, we define the concepts and the relationships in the meta model.

[115] There are two main approaches to IS modelling, the structured approach (e.g. Yourdon 1989) and the object-oriented approach (e.g. Booch *et al.* 1999). Our approach mainly follows the structured approach that views information processing as information flows between the processes.

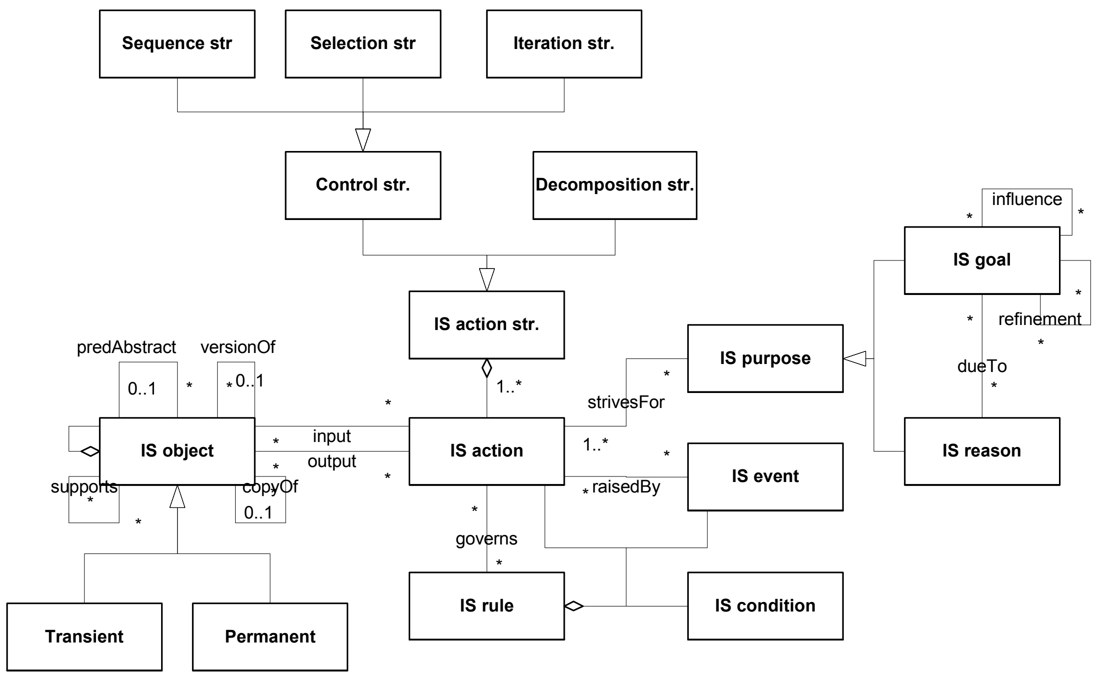


FIGURE 58 Meta model of the IS infological perspective

IS purposes mean IS goals for information processing and/or reasons for setting up those goals. An *IS goal* is a desired state of affairs in the IS. *IS reasons* can be functional or non-functional requirements for information processing, problems in prevailing information processing, strengths and weaknesses in, opportunities for and threats against existing or planned information processing. Between the IS goals there are complex influence relationships and refinement relationships.

In striving for the IS purposes, the *IS actions* use informational objects, called *IS objects,* as inputs and produce IS objects as outputs. The range of various types of IS actions is huge. An IS action can mean e.g. collecting, storing, processing, transmitting, coding, encoding, arranging, locating, discovering, interpreting, integrating, reviewing, testing, approving, editing, etc.

From the action structures defined in Section 4.4.3 relevant structures from the IS infological perspective are the decomposition structure and the control structure. The decomposition structure splits IS actions into IS functions, IS activities, IS tasks, and IS operations. Because the terms with which the IS actions in the decomposition structure are referred to are varying, we do not include them in the meta model in Figure 58. The control structure allows to present sequence, selection and iteration relationships between the IS actions.

The IS actions are governed by IS rules. An *IS rule* is composed of *IS events, IS conditions, thenISActions* and *elseISActions.* The IS rules can be classified in many ways. First, there are dynamic and static rules. The dynamic IS rules restrict or guide IS actions and IS events. The static IS rules restrict IS objects. Examples of the IS rules are: back-ups of the files should be run once a week; a social security number of a person cannot be changed; a salary of an hourly paid employee is derived by the rule 'Salary := number of hours x hourly fee'. The first example is a business rule. The second rule is called an integrity constraint (Elmasri *et al.* 2000). The last rule is called a derivation rule (cf. Iivari 1989a).

An IS object is an informational object in the form that is free from any representational and implementational aspects. An IS object can be transient or permanent. A *transient IS object* lasts only a short time (e.g. a reply to a routine request). A *permanent IS object* is valuable enough to "live" longer (e.g. personnel information, vehicle information).

The IS objects are interrelated in many ways. They are composed of other IS objects. Producing them is supported by other IS objects (cf. derivation of the monthly salary from hourly fee and number of hours). An IS object can also be a version of, a copy of, or an (predicate) abstraction from, another IS object.

6.3.3 IS Conceptual Perspective

The IS conceptual perspective aims to reveal the semantic contents of the IS objects. This means that of those things in the OS_{IS} that are signified by the IS objects, the structure and behavior are brought out. The IS conceptual perspective addresses the so-called deep structure of the IS (Wand *et al.* 1995b).

The OS is here seen as being composed of related things having states and affected by state transitions (cf. Section 3.7). The structural view concerns the states, and the dynamic view addresses the state transitions. In OS modelling there are several approaches. Some of them are structural, such as the ER approach (Chen 1976) and ORM approach (Halpin 1988; 2001), some other cover both views, such as the object-oriented approach (Booch *et al.* 1999). We prefer the ER approach to the ORM approach and other attribute-free approaches, because we consider it important to separate between entities and attributes. We want also to make a clear distinction between the static features and the dynamic features of the OS, unlike in the object-oriented approach. The meta model of the IS conceptual perspective based on the ER approach (the structural view) and the state machine (the dynamic view) is presented in Figure 59.

In the core ontology, thing was defined as a generic notion to mean any phenomenon in reality. Likewise, the notion of a relationship means anything that relates two or more things together. To emphasise the specificity of the IS conceptual perspective and the OS, we introduce here another elementary concept, called entity. An *entity* means any perceivable thing in the object system with an independent existence (cf. Elmasri *et al.* 2000, 45). Only those things that are relevant and "independent" enough to be signified by the IS objects are regarded as entities. Examples of entities are John and Mary.

It would be better to introduce, instead of the generic notion of a relationship included in the core ontology, a separate concept for relating the entities. However, because in the ER approach (Chen 1976) it is customary to use the term 'relationship' in this meaning, we do not want to make any

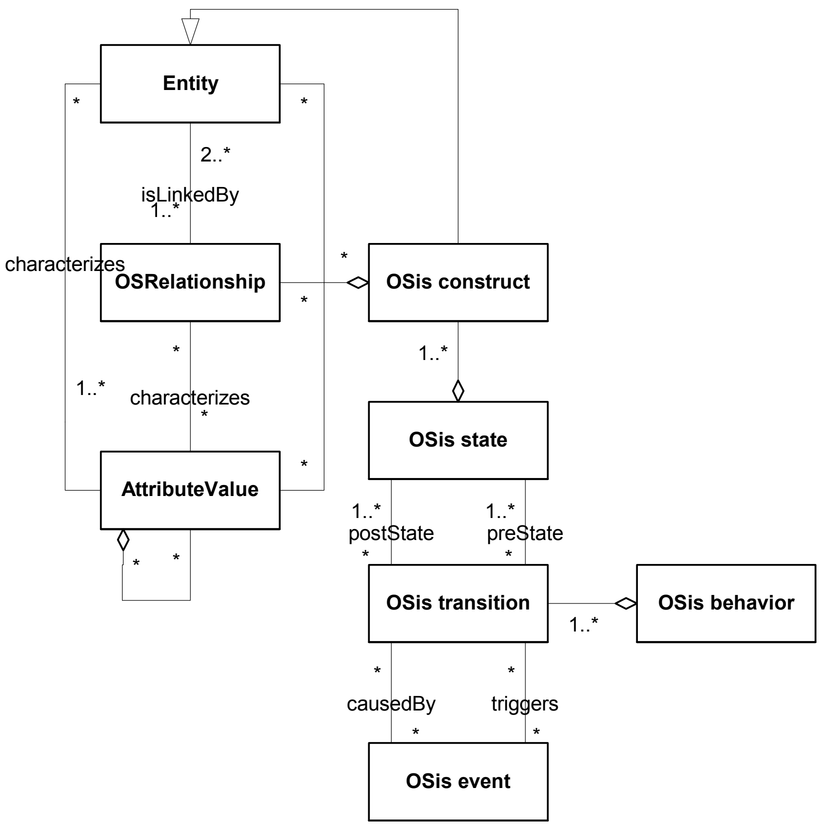


FIGURE 59 Meta model of the IS conceptual perspective

deviation in this case. However, to avoid possible confusions, we use the special term 'OS relationship' to differentiate it from the core notion of a relationship. Hence, an *OS relationship* between two or more entities means any relevant connection, association or like (i.e. a relationship) between the entities. A marriage between John and Mary is a relationship. All the abstraction relationships (classification, generalization, composition, grouping) defined in the abstraction ontology apply, of course, to the entities as well.

An attribute is a relevant predicate used to characterize an entity or an OS relationship. A particular entity or OS relationship has one or more *attribute values* for each of its attributes. For instance, 25 and 26 are ages of John and Mary, respectively. In some cases, a particular entity or OS relationship may not have an applicable value for an attribute. In such situations, a special value, called null, is used.

An *OS_{IS} construct* means a conceptual construct composed of specific entities related to one another through OS relationships and characterized by specific attribute values. An OS_{IS} construct is a UoD construct defined in Section 4.4.4, here conceived from a more specific viewpoint. The notion allows us to refer to complex structures in the OS with one term. The OS_{IS} constructs are here considered at the instance level.

An *OS_{IS} state* means a state of the object system or its parts, composed of OS_{IS} constructs. An *OS_{IS} transition* means a transition from one OS_{IS} state, called the pre-state, to another OS$_{IS}$ state, called the post-state. An OS$_{IS}$ transition can involve entities (e.g. the "birth" of an entity), OS relationships (e.g. the divorce) and/or attribute values (e.g. the quantity available). The transitions constitute the potential *OS$_{IS}$ behavior* (cf. Section 3.8). OS$_{IS}$ transitions can be composed to establish OS$_{IS}$ transition structures like those defined in the state transition ontology (Section 3.7). An *OS$_{IS}$ event* means an event which may trigger an OS$_{IS}$ transition from the pre-state to the post-state and which may be caused by another OS$_{IS}$ state transition[116].

6.3.4 IS Datalogical Perspective

From the *IS datalogical perspective* the IS is viewed, through representation-specific concepts, as a context, in which IS actors work with IS facilities to process data. Thus, the IS objects, seen as informational objects from the IS infological perspective, are here considered to be data objects presented in some non-formal, semi-formal or formal language(s). There are also special IS actions which transform data objects from one form to another. No reference is made to data carriers or other physical features of the IS context. The IS datalogical perspective enables, however, to distinguish between human data processing and computerized data processing. Due to these two related parts (i.e. HIS and CIS), there is also a need to consider how the parts communicate and cooperate, i.e. what is the user interface of a CIS.

The meta model of the IS datalogical perspective is presented in Figure 60. The concepts and relationships of the HIS have been specialized from the context ontology (Chapter 4). The CIS is conceptually very large. In this study it is possible to introduce only a small number of its concepts. For the UI part, both structural and behavioral aspects on a logical level are covered. Next we define the concepts and relationships of the IS datalogical perspective, first for the HIS, then for the UI and finally for the CIS.

A. Human Information System

The *human information system* (HIS) means a system in which human beings have the only role in the accomplishment of the IS actions. From the IS datalogical perspective, the HIS is seen as a context, in which HIS actions process data objects, governed by HIS rules, to attain HIS purposes. Because the HIS is a context, all the generic contextual concepts and constructs within the aforementioned domains defined in Chapter 4 apply to it. For that reason, we are not going to define all the concepts and relationships here but concentrate on the most essential ones.

[116] Note that some researchers (e.g. Iivari 1989a; Freeman *et al.* 1994) include also the notion of an action in the OS. Because we want to make a clear separation between the OS and the IS, all behavioral aspects of the OS are modeled through state transitions.

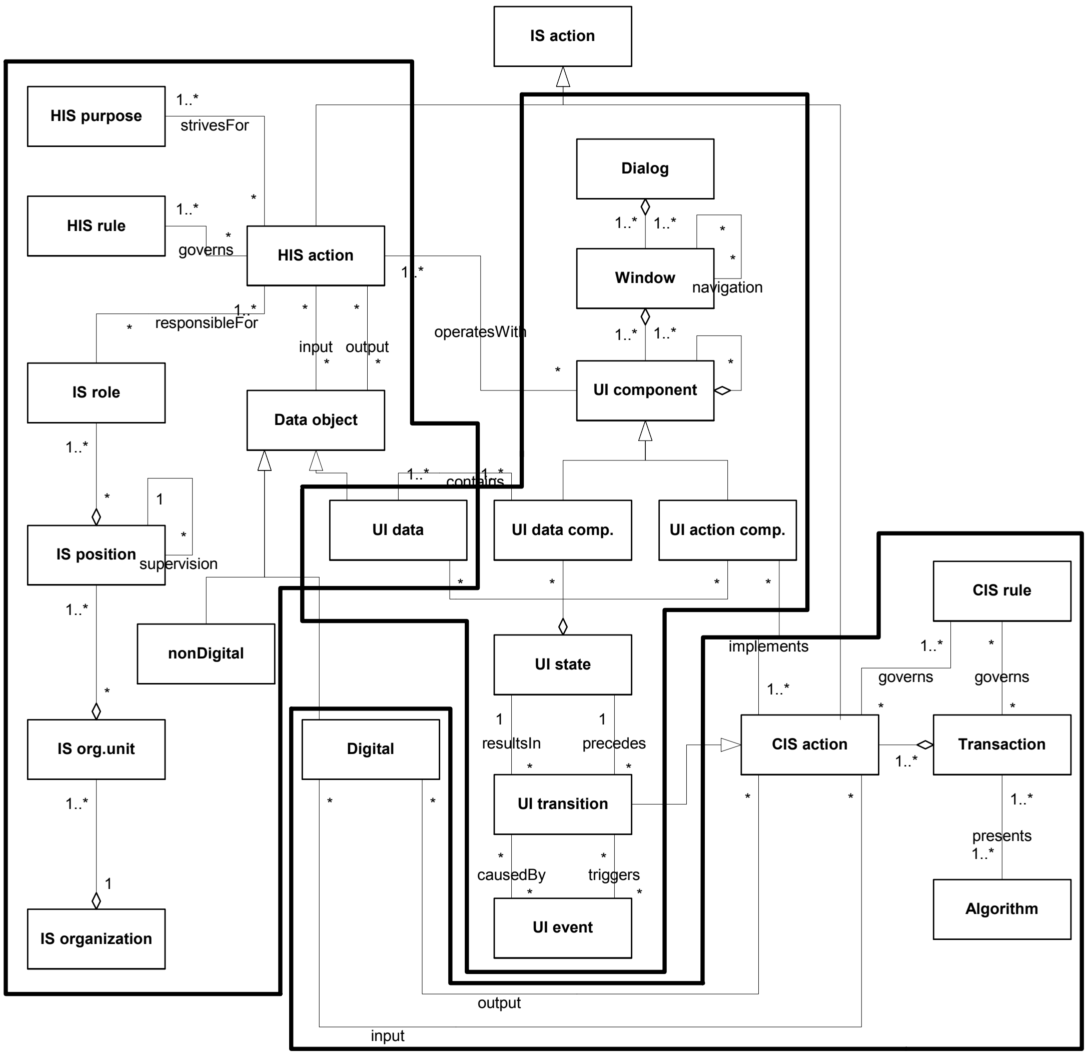


FIGURE 60 Meta model of the IS datalogical perspective

A *HIS action* is an IS action carried out by a human IS actor. A *HIS purpose* is an IS purpose, which concerns the HIS as a whole, or parts thereof. An *IS role* is a collection of responsibilities, stipulated in terms of HIS actions. One IS role is a user of the CIS. An *IS position* is a post of employment specified in terms of IS roles. Between two IS positions, a supervisor and a subordinate, there is the supervision relationship. An *IS organization* is an organization whose main responsibility is to develop, manage and/or execute information processing in a business organization. It is composed of one or more *IS organizational units.*

The HIS actions are governed by *HIS rules* with the ECAA structure. The HIS actions are related to one another with generic action structures (i.e. decomposition, control structures, and temporal structures), the problem solving structure, and management-execution structure (see Section 4.4.3).

A *data object* is an IS object represented in some language. It can be in a digital or non-digital form. *Non-digital data* means an IS object that is presented in a language that can be interpreted by a human being. The HIS actions mainly handle non-digital data. *Digital data* is in a digital form and can be read by a computer.

B. User Interface

The support of the CIS to the users appears as services that a user requests and receives from the system. Receiving services requires communication or interaction between a user and the system. A *user interface* (UI) is a part of the CIS which facilitates the interaction between the users and the CIS. From the IS datalogical perspective, interaction means "what a user is able to do and what the system does in response to the user's stimuli" (cf. the logical level in de Rosis *et al.* (1998, 101)). In the following we define the main concepts that are used in the UI design at the datalogical level.

A *dialog* means an interaction between a user and the CIS, occurring through windows. A *window*[117] is a logical whole composed of UI components. Windows are related with the navigation relationships. A *navigation relationship* means a possibility for a user to move control from one window to another window. A *UI component* can be a UI data component or a UI action component. A *UI data component* is intended to display data to a user or to accept data from a user (cf. a feedback tool in Jaaksi (1995, 1212). This data is called *UI data.* Depending on whether UI data is handled by a CIS or a human being, it is digital or non-digital data. A UI data component can be a title, a text, a data field, a table, a picture, a graph, etc. A *UI action component* is intended to the manipulation of the window and the control of the dialog (cf. a manipulation tool in Jaaksi (1995, 1212). It can be a button, a menu, a slider, etc. The UI action components are used to realize navigation among the windows. An UI action component is implemented by performing one or more *CIS actions* governed by *CIS rules.* A UI component can contain both UI data components and UI action components. The HIS operates with UI components through HIS actions.

Interaction between the HIS and the CIS proceeds from one UI state to another. A *UI state* is composed of those UI data, UI data components and UI action components that are present at the certain time. A *UI transition* from one UI state to another can be triggered by the HIS (i.e. an HIS action) or by the CIS (i.e. a CIS action). *UI events* correspond to all those happenings that can trigger UI transitions. For example, a UI pre-state can concern a window, which contains the search condition 'John Doe' in a text field and Search, Cancel and Clear buttons. After the Search button is pressed, the CIS searches for the data concerning John Doe and displays it in the next window (the post-UI state).

C. Computerized Information System

A *computerized information system* (CIS) a system in which all data processing is automated, that is to say, performed by one or more computer systems. Also

[117] In a web-based application a web page corresponds to a window.

here the IS datalogical perspective reveals only those features that are not related to the physical technology.

A *CIS action* means an IS action that is performed in the CIS. A CIS action records, searches for, orders, merges, updates and/or deletes, i.e., processes digital data. Some of this data is displayed as UI data through a UI data component. Logically related CIS actions are assembled to form *transactions.* A CIS action and a transaction are governed by *CIS rules.* Transactions, represented in a formal language, are called *algorithms.*

6.3.5 IS Physical Perspective

The *IS physical perspective* considers the IS with all its physical aspects. It ties the IS datalogical concepts and constructs to a particular organizational and technical environment, showing how the IS looks like and behaves when it is implemented. The IS contains one or three major parts, namely the HIS, and possibly the CIS and the UI. For all the parts, highly detailed and realization-dependent view is enabled. It is quite impossible for us here to address all those details. Instead, we content ourselves with presenting an outline of the IS from the IS physical perspective. After that we present the meta model of the CIS (Figure 61) and define the concepts contained in it.

The IS roles are combined to form IS positions with organization-specific responsibilities and authorities. For each position one or more IS human actors are assigned. IS positions are structured to establish IS organizational units. The HIS actions are organized according to the organization-specific management – execution structures. They are also decomposed into detailed tasks and operations, governed by specific HIS rules that are realized to suit the organizational culture and practice. To non-digital data objects (e.g. reports, forms and tables) suitable data carriers are attached and layouts fixed. Resources are assigned to IS organizational units and allocated into parts thereof.

The UI is composed of physical windows built up from physical UI components; e.g. a command button, a radio button, a spin button, a check box, a list box, a drop-down list box, a pop-up menu, and a tool bar. UI components as well as transactions are realized through software components programmed in one or more programming languages. For software a proper architecture is specified and implemented. Digital data is structured and stored in data storages. For data communication through data messages, communication lines between nodes with compatible interfaces are established. Finally, all the human actors, data and facilities are situated into specific locations.

The skeleton of the implemented CIS is composed of a hardware (HW) architecture and a software (SW) architecture. A *hardware architecture* consists of interoperable hardware. Hardware means physical equipment used in data processing (e.g. workstations, servers printers, mass data storages) (cf. IEEE 1990). A *software architecture* is composed of compatible software (e.g. operating systems, database management systems, application software). There are

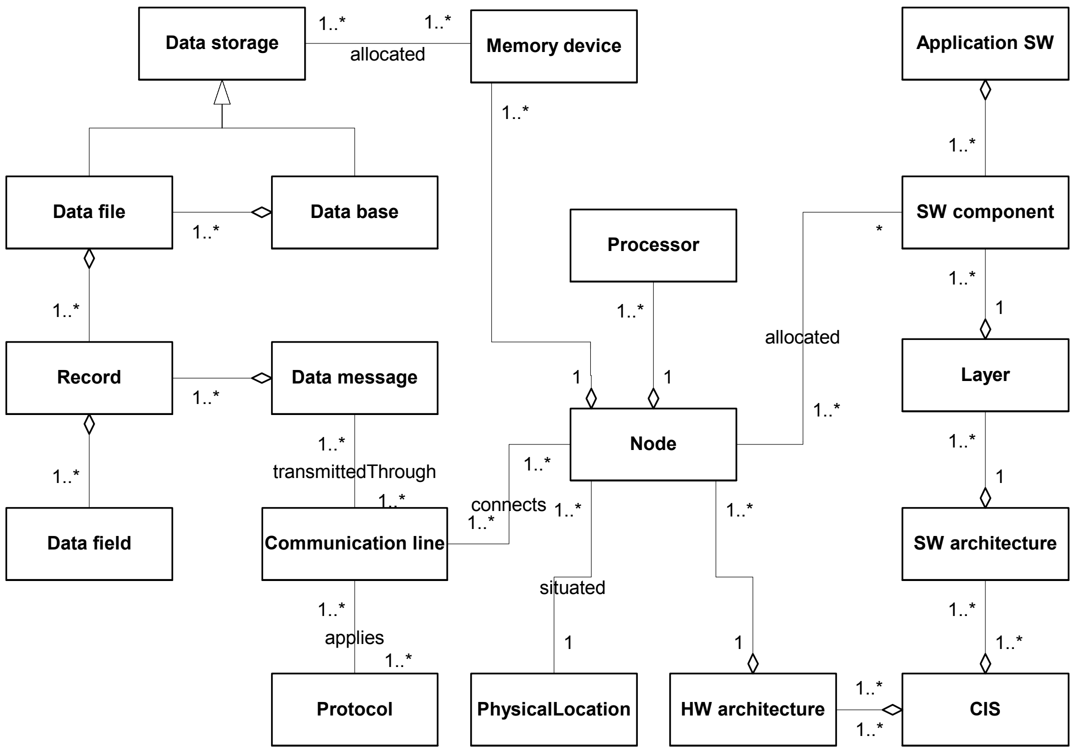


FIGURE 61 Meta model of the IS physical perspective covering a part of the CIS

several SW architecture types and styles (Buschmann *et al.* 1996, Bass *et al.* 1998) that can be applied. On the basis of a layered architecture, an *application software* is composed of SW components that are layered according to some basic architecture model. *Layers* are related to one another with the black box strategy or the while box strategy. In the former case, a component on a higher layer only knows the interface of the called component on the next lower layer. In the latter case, the component on a higher layer also sees the inner structure of the called component. A *SW component* means an executable unit of code that provides a physical black-box encapsulation of related services. Its services can only be accessed through a consistent, published interface (Allen *et al.* 1998, 4). Examples of SW components (of a database system) are stored procedures, functions, data base triggers, UI components, etc

Hardware is organized into nodes according to the selected HW architecture. A *node* is composed of e.g. memory devices, processors, printers and displays. A software component is allocated into one or more nodes. Each node is situated in some physical location. Communication from one node to another takes place through data messages sent along *communication lines.* During the communication, encoding and decoding of messages is performed based on specified protocols. A *protocol* means a set of conventions or rules that govern the interactions of processes or software components through communication lines in a CIS or between CIS's (e.g. TCP/IP, HTTP) (cf. IEEE 1990).

A *data storage* stands for all kinds of structured digital data (e.g. data file and database). A *database* is structured according to some database model (e.g. a hierarchical model, a relational model, an object-relational model, an object model, a document model, XML-native model). *Data files* are decomposed into *records* and further *data fields.* A data storage is allocated into some *memory device*(s).

6.3.6 Relationships between the IS Perspectives

In the previous sections we defined the contextual concepts and relationships within each IS perspective. Here, our purpose is to relate the IS perspectives, first on a general level and then in more detail, through the contextual domains involved by the IS perspectives.

The perspectives have been established along three dimensions. Based on the dimensions and the related discussions in Section 6.1, we can sketch the relationships between the IS perspectives on a general level as shown in Figure 62[118]. The common denominator between the IS systelogical perspective and the IS infological perspective is the IS: moving from the former perspective to the latter means that the IS, first seen as a black box, is opened in order to expose IS purposes, IS actions, IS objects and relationships between them. In this process, the principles of decomposition and specialization (by contextualization) are mainly applied. Boxes inside the IS systelogical, IS infological, IS datalogical and IS physical perspectives stand for informational objects which signify conceptual constructs in the object system.

The IS infological, IS datalogical and IS physical perspectives are parts of a hierarchical system of perspectives within which the relationships are based on the same criterion of realization independence. Applying the criterion of realization independence means carrying out the process of predicate abstraction (cf. Section 3.9.3). And vice versa, moving downwards in the hierarchy, conceptions about the IS first become representation-specific (cf. the IS datalogical perspective) and then implementation-specific (cf. the IS physical perspective). In parallel to realizing, concretizing by decomposition and specialization is applied. In the last "stage" also instantiation is carried out.

Each of the aforementioned IS perspectives recognizes informational objects. In the IS systelogical perspective they are called (informational) US objects. According to the IS infological perspective, the information system contains IS objects. Based on the IS datalogical perspective they are digital or non-digital data objects. Data files, data records and data fields represent the conceptions of IS objects from the IS physical perspective. In all those cases, there are signifies relationships between informational objects and things conceived as OS_{IS} construct from the IS conceptual perspective. Through these relationships it is possible to make sense of the semantic meanings of the informational objects.

118 Note that because Figure 62 does not present a meta model we use arrows to show, in a more illustrative way, the directions of the relationships.

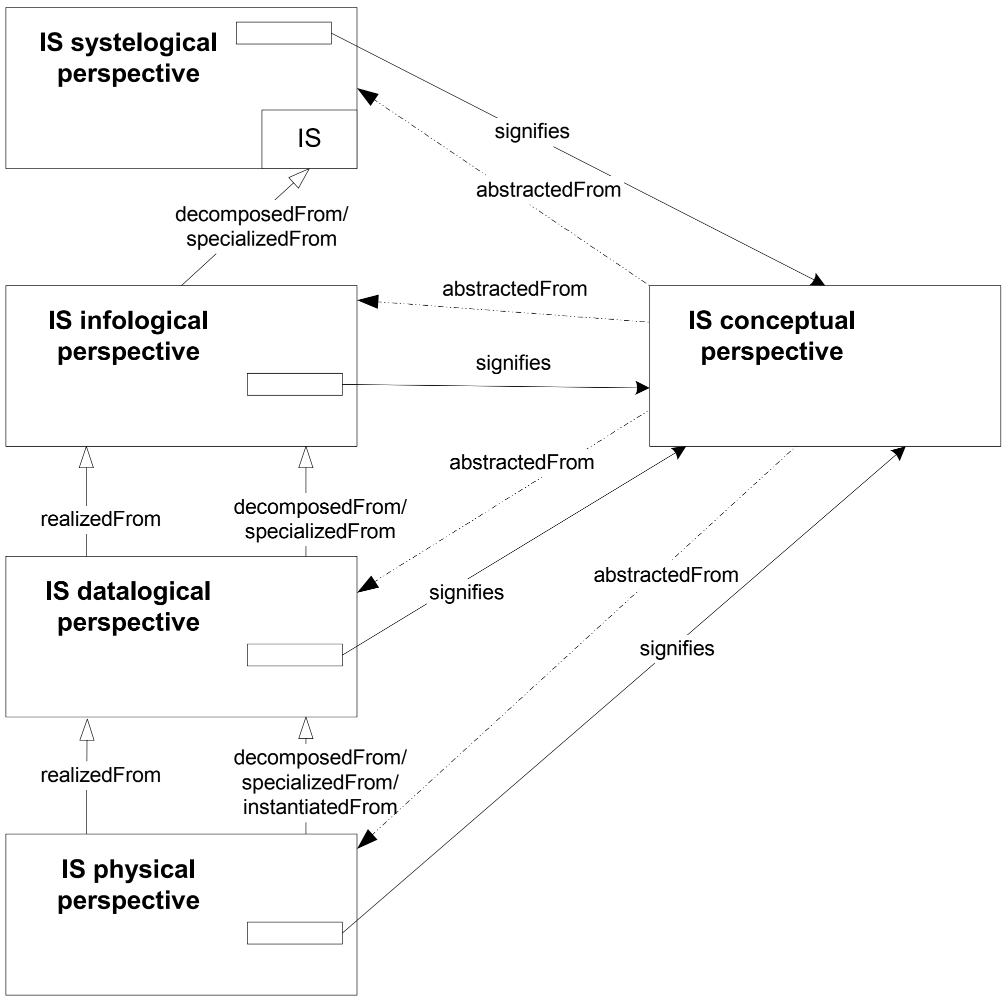

IS systelogical perspective
IS
signifies
abstractedFrom
decomposedFrom/ specializedFrom
IS infological perspective
abstractedFrom
signifies
IS conceptual perspective
abstractedFrom
realizedFrom
decomposedFrom/ specializedFrom
IS datalogical perspective
signifies
abstractedFrom
signifies
realizedFrom
decomposedFrom/ specializedFrom/ instantiatedFrom
IS physical perspective

FIGURE 62 A general view of the relationships between the IS perspectives

Still one type of a generic relationship can be found between IS perspectives. If the OS overlaps with the US or the IS, there is abstractedFrom relationships between the OS_{IS} and the US_{IS}, on one hand, and between the OS_{IS} and the IS_{IS}, on the other hand. By this abstraction, most of the contextual aspects of the US_{IS} / IS_{IS} are ignored in order to establish OS_{IS} constructs composed of entities, OS relationships and attribute values. Let us consider two examples of the abstractedFrom relationship between the OS_{IS} and the US_{IS}. If considered relevant, US actors, US objects, US facilities and/or US locations can be conceived as entities that are related via OS relationships corresponding to the relationships in the US_{IS} (viz. responsibleFor, occupiedBy, supports, performs etc.). In the same way, OS transitions can be abstracted from the US actions. For example, hiring and firing an employee affect on the OS state of a particular employee.

Next, we consider the relationships between the IS perspectives more closely through the contextual domains involved with the IS perspectives. An elaborated view embracing the concerned contextual domains is presented in Figure 63. Applying the IS systelogical perspective (the tool view) all the contextual domains of the US context are recognized. The IS infological

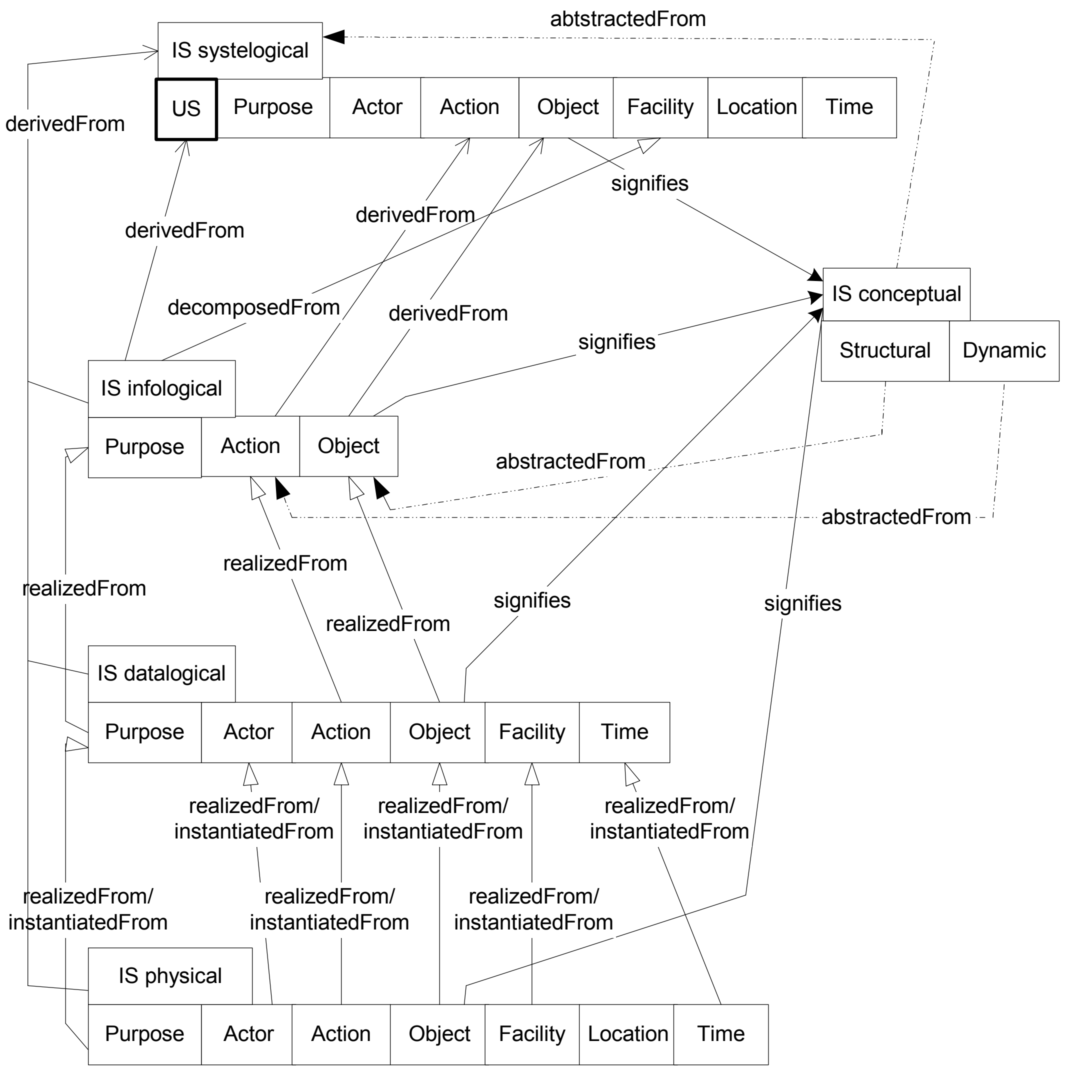


FIGURE 63 A detailed view of the relationships between the IS perspectives

perspective involves only the purpose, action and object domains of the IS context. The IS conceptual perspective is on the most general level. It does not divide the aspects of the OS_{IS} into contextual domains at all. Instead, it considers the OS_{IS} constructs as states (i.e. structural features) and transitions between the states (i.e. dynamic features). Compared to the IS infological perspective, the IS datalogical perspective brings forward two new contextual domains, the actor domain and the facility domain. Both of them are considered on a general level. The most detailed view of the IS in all the contextual domains is naturally got when looked at from the IS physical perspective.

In Figure 63 the relationships between the IS perspectives are depicted with arrows (a) between the domains, (b) between a domain and a perspective, and (c) between the perspectives. In the first case, the relationships are defined on the basis of individual domains. In the second case, a thing in some domain is argued to have a relationship with a system, either the US_{IS} or the IS, seen as a whole from a certain perspective. The arrows between the systems indicate a more general connection between the concerned perspectives. To avoid excessive complexity in the figure, some of the decomposition and specialization relationships are omitted in Figure 63. Next, we make some remarks on the relationships.

To start with, the concepts and the relationships within the domains of the IS infological perspectives are derived from the ones within the corresponding domains of the IS systelogical perspectives. A particularly important role in this derivation is played by the informational US objects used in decision making or operational actions in the US, because the US objects are exactly what the IS should provide as information services. Following the infological approach (Langefors *et al.* 1975; Lundeberg 1982) the informational US objects are regarded as final outcomes of the IS from which IS actions and their inputs are derived in a step-by-step manner. Contextual information for this derivation is obtained from the knowledge about those US actions which utilize the US objects and on the purposes for which the US objects are used. In parallel to the derivation, IS purposes, IS actions and IS objects are decomposed into more elementary parts. In this process, also more elementary concepts in the hierarchical structures in the domains (cf. goal/means hierarchy, action decomposition structures, object decomposition structure) are applied. Furthermore, the business rules stated within the US_{IS} can be applied to specify IS rules. More grounds for the considerations from the IS infological perspectives can be got from the aspects (e.g. requirements) related to the CIS as a US tool. It should also be noticed that implications of the IS systelogical perspective to the IS infological perspective also appear at a more general level. The whole purpose of the IS is affected by the purpose of the US_{IS}. For instance, if the goal of an enterprise is to deliver the goods within two days to customers living not further off than 100 miles, it implies that the IS should support nearly on-line responses to the issued orders.

Second, the informational objects in the IS systelogical, IS infological, datalogical and IS physical perspectives give a scope and basis for viewing the OS_{IS} from the IS conceptual perspective. If the OS_{IS} overlaps with the US_{IS} and/or the IS, the overviews from the related perspectives can help unveil static and dynamic features of the corresponding parts of the OS_{IS}. Thus, the relationship between the IS conceptual perspective and the other IS perspectives is twofold: (a) informational objects signify OS_{IS} constructs, and (b) the IS conceptual perspective abstracts from the relevant parts of the US_{IS} and/or the IS.

Third, because the IS infological, IS datalogical and IS physical perspectives are parts of the hierarchical system of perspectives based on the realization-dependence criterion, for each domain type more concrete concepts and relationships are deployed at lower levels of predicate abstraction. Consequently, data objects are derived from information objects, IS actions are divided into HIS actions and CIS actions, and temporal specifications of the IS datalogical perspective are realized into more concrete specifications of the IS physical perspective. Derivation is, of course, a very complicated process, which follows strategies, approaches, principles, and techniques selected in an

ISD endeavor. As an example of these normative guidelines, we refer to Bailey (1989, 189), who distinguishes between five strategies of allocation of IS actions into an HIS or a CIS (Kueng *et al.* 1996, 104-105):

- *Comparison allocation.* Each IS action is analyzed and then compared with established human and machine performance criteria.
- *Leftover allocation.* As many IS actions as possible are allocated to a machine and the activities left over are done by humans.
- *Economic strategy.* The decision, man versus machine, is based completely on financial assessment.
- *Humanized task approach.* The main goal of this strategy is to design meaningful human jobs / roles (cf. socio-technical approach in Mumford *et al.* (1979) and Mumford (1981)).
- *Flexible allocation.* Humans allocate activities in the HIS or in the CIS based on their values, needs, and interests.

Proceeding into lower levels of predicate abstraction also brings forward new domains. In the IS datalogical perspective some of the IS actions sketched in the IS infological perspective are elaborated and combined to establish IS roles, IS positions and IS organizational units in the actor domain. For the HIS the CIS is seen as a tool, which belongs to the facility domain. In the IS physical perspective, the concepts of the location domain are used to situate all the structural things of the IS in their proper locations.

Progress from the IS infological perspective through the IS datalogical perspective to the IS physical perspective is not just the application of the principle of realization. The view of the US_{IS}, formed by the IS systelogical perspective, has a significant effect on design decisions on all the levels of predicate abstraction. For example, decisions on the division of IS actions into HIS actions and CIS actions, on assembling HIS actions into IS roles and IS positions, and on forming HIS action structures, on a detailed level, are based on contextual knowledge about the US. The effect of the IS systelogical perspective becomes particularly important when moving to the IS physical perspective, which ties the IS designs to concrete organizational and technological settings.

6.4 Comparative Analysis of IS Perspectives

The IS/ISD literature provides a large variety of IS architectures (e.g. Zachman 1987; Sowa *et al.* 1992), IS frameworks (e.g. Olive 1983; Olle *et al.* 1988a; Iivari 1989a; van Swede *et al.* 1993), reference models (e.g. ISO 1996) and IS meta models (e.g. Freeman *et al.* 1994) that contain, in one form or another, views, perspectives and viewpoints to structure descriptions of the IS. Starting from the early days, the ANSI/SPARC Study Group (ANSI/X3 SPARC 1975), for instance, introduced the so-called ANSI/SPARC architecture for data bases, composed of three schemas: an internal schema describing the physical storage structure of the data base, a conceptual schema presenting the structure of the whole database at a conceptual level, and an external schema describing a part of the database that a particular user group is interested in. Langefors and Sundgren (1975) distinguished between the infological viewpoint and the datalogical viewpoint. Kerola and Järvinen (1975) specified the so-called PSC model which is based on three main points of view: the pragmatic point of view, the semantic point of view, and the constructive point of view. Welke (1977) distinguished between the systelogical perspective, the infological perspective, and the datalogical perspective. Iivari (1978) and Mustonen (1978) elaborated the points of view in the PSC model further and defined the pragmatic (P), input output (I/O), constructive (C), and operation (O) viewpoints, which Iivari (1983) later combined into the so-called PIOCO model. Gane *et al.* (1979) recognized the physical view, representing the current work practice, and the logical view, representing technology-independent solutions. Since those days, a substantial number of presentations have been published in which the usefulness of perspectives has frequently been emphasized.

The purpose of this section is to make a short review and a comparative analysis of the IS perspectives defined in frameworks, reference models, architectures and the like in the IS/ISD literature. For simplicity we call those presentations that specify perspectives, one way or another, the frameworks. First, we specify the criteria for the selection of frameworks. Second, we set up the objectives for our analysis. Third, we describe and analyze the IS perspectives included in the frameworks from multiple viewpoints.

Overviews of the most prominent frameworks have already been given in Chapter 1. For the analysis, we select those frameworks (a) which have been developed for a comprehensive analysis and/or comparison of the concepts in the fields of IS and/or ISD and (b) in which systems of perspectives have been clearly specified. Some of the frameworks are detailed providing also concepts within the perspectives, while the others only generally characterize the conceptual domains addressed by the perspectives. The criteria set up above are fulfilled in eleven frameworks. These are: Welke (1977), Olive (1983), Essink (1986, 1988), Olle *et al.* (1988a), Iivari (1989a), Sol (1992), Sowa *et al.* (1992), van Swede *et al.* (1993), Freeman *et al.* (1994), Avison *et al.* (1996)[119] and ISO (1996).

Many of the frameworks presented in the literature were left out. We ignored, for instance, frameworks, which are predecessors of some frameworks already included in the analysis (e.g. Kerola *et al.* 1982), or alternatively, which are significantly based on some framework included in the analysis (e.g. Punter *et al.* (1996) is based on Essink (1986, 1988), and Evernden (1996) is based on Sowa *et al.* (1992)). Some of the frameworks are too technical (e.g. ANSI/X3

[119] Multiview by Avison *et al.* (1996) is actually a methodology, but because it does not aim to provide a particular way of working neither a single set of techniques but rather a contingency framework according to which the selections and customizations of techniques are to be carried out for a project (cf. Watson *et al.* 1995; Vidgen 2002), we regard it here as a framework.

SPARC 1975; Booch *et al.* 1999, 31) or domain-specific (e.g. Frank's framework (Frank 2002) is based on enterprise modeling and Tardieu's framework (1992) is aimed at process modeling) to be included in the analysis.

Our aim is to make (a) an overview of the perspectives in the frameworks and of the dimension(s) with the criteria used to establish them, (b) a rough comparative analysis of the conceptual contents of the perspectives, and (c) a more detailed analysis of the concepts within the perspectives. For the third part of the analysis we select only the most detailed frameworks. The first two parts address the systems of perspectives generally (cf. the perspective ontology in Section 6.1). In the third part we want to find out how the IS perspectives defined in Section 6.3 compare to the perspectives in the frameworks in the literature. This part of the analysis has importance also more generally, because we did not make any comparative analysis for the context ontology. Hence, this analysis also reveals how our context ontology, from which the IS perspectives have been derived, compete with the frameworks in the literature.

6.4.1 Overview of the Perspectives

In Table 19 we present an overview of the perspectives of the selected frameworks. The overview covers the dimension(s), the criteria by which the perspectives have been derived and are interrelated, and the perspectives. In the following we comment on the table with some remarks. The frameworks will be outlined and their perspectives will be defined in the next part of the analysis. Here, our discussion is based on the information in the table only.

The dimensions, along which the perspectives in the frameworks have been established, are called "levels of abstraction" (Olive 1983; Essink 1988; Iivari 1989a; Freeman *et al.* 1994), "design levels" (Sowa *et al.* 1992), "aspects" (Olle *et al.* 1988a)," views" (Avison *et al.* 1996), "viewpoints" (ISO 1996), or "sub-problems" (Sol 1992). van Swede *et al.* (1993) and Welke (1977) call them "perspectives", as we do.

The frameworks use different criteria to distinguish between and relating the perspectives. Sowa *et al.* (1992) and van Swede *et al.* (1993) have based their design levels / perspectives on views of stakeholders. With an analogy to the construction of a building, Sowa *et al.* (1992) recognize five stakeholders. van Swede *et al.* (1993) have reduced the number of stakeholders to three: those concerned with the way the system supports the business, those concerned with the use of the system, and those concerned with design matters irrelevant to users. The view of the first role is further divided into two abstraction levels. As a refinement of the well-known dichotomy of logical IS model and internal IS specification, Essink (1986, 1988) has devised his levels of abstraction by "grouping the design decisions according to their functional cohesion" (Essink 1986, 57).

Iivari (1989a, 327) has derived his levels of abstraction from abstractions of the host organization, universe of discourse, and technology. The host system constitutes the organisational context of the IS. The UoD gives the propositional/ conceptual meaning of information in the infological model. The

TABLE 19 Overview of the frameworks and their perspectives

Framework	**Dimension(s)**	**Criteria**	**Perspectives**
Welke (1977)	Perspectives	Changes in IS and/or its user-subsystem should be addressed from several perspectives (ibid p. 150)	Systelogical perspective Infological perspective Datalogical perspective
Olive (1983)	Levels of abstraction	"The model at the highest level is the most general and those at the lower level are more detailed" (ibid p. 63)	External level Conceptual level Logical level Architectural level Physical level
Essink (1986, 1988)	Levels of abstraction	"..are classes of problems that are relevant from a specific view on IS's" (Essink 1988, 356)"	Object system model Conceptual IS model Data system model Implementation model
Olle *et al.* (1988a)	Aspects	Not clearly specified	Business analysis stage System design stage Construction design stage
Iivari (1989a	Levels of abstraction	Derived from abstractions of the host system, the UoD and technology	Organizational level Conceptual/infological level Datalogical/technical level
Avison *et al.* (1996)	Views	"..necessary to form a system which is complete in both technical and human terms"	Human-activity view Information view Socio-technical view Human-computer interface view Technical view
Sol (1992)	Sub-problems	Not clearly specified	Systelogical problems Infological problems Datalogical problems Technological problems
Sowa *et al.* (1992)	Design levels	Levels correspond to views of specific stakeholders	Scope Enterprise or business model System model Technology model Components
van Swede *et al.* (1993)	Perspectives	Perspectives correspond to views of specific groups of people	Business perspective Information perspective Functionality perspective Implementation perspective

(continues)

TABLE 19 (continues)

Framework	Dimension(s)	Criteria	Perspectives
Freeman *et al.* (1994)	Levels of abstraction	"..loosely corresponds to the phases of software development" (ibid 287)	World level Conceptual level Design level Implementation level
ISO (1996)	Viewpoints	"..to focus on particular concerns within a system" (ibid 3.2.7)	Enterprise viewpoint Information viewpoint Computational viewpoint Engineering viewpoint Technology viewpoint

abstract technology describes the allocation of the functional components of the datalogical model to the abstract technical resources. Welke (1977) has based his perspectives on consequences that changes to the existing IS result in the object system, the use of information, and the data processing sequences. Avison *et al.* (1996) argue that their five views are needed to answer the vital questions of users. The corresponding stages move from general to specific, from conceptual to hard facts, and from issues to tasks. Freeman *et al.* (1994) compare their levels of abstraction with the phases of software development. Sol (1992) has made his division on the basis of the kinds of problems that must be solved during ISD.

Some frameworks give no explanation for the perspectives, nor apply any explicit criteria. Olive (1983), for example, says that the analysis of ISD methods has resulted in five levels of abstraction. The model at the highest level is the most general and those at the lower level are more detailed (ibid p. 63). ISO (1996) has aimed at "covering all the domains of architectural design" and derived the viewpoints "from current distributed processing development".

6.4.2 Detailed Analysis of the Perspectives

In this section we describe and analyse the perspectives of the frameworks in more detail. Our aim is to compare the conceptual contents of the perspectives in the frameworks. For this purpose, we use our system of perspectives as the basis for the analysis. In Table 20 "strong" correspondences between our perspectives and those in the other frameworks are marked with 'X' and "weak" correspondences are indicated by 'x'.

Welke (1977) states that changes in the IS and/or its user-subsystem (i.e. the US in our terminology) should be addressed from the following perspectives: (a) systelogical perspective (How will changes to the existing IS alter / facilitate changes in the affected object system?), (b) infological perspective (How will changes to the existing IS alter/facilitate changes in the use of information by individuals?), (c) datalogical perspective (How will changes to the IS alter/facilitate changes in the data processing sequences

TABLE 20 Comparative analysis of the perspectives

Frameworks	**Syste-logical**	**Info-logical**	**Concept-ual**	**Data-logical**	**Physi-cal**
Welke (1977)					
- Systelogical perspective	X	x	x		
- Infological perspective		X		x	x
- Datalogical perspective				X	x
Olive (1983)					
- External level	X	x			
- Conceptual level			X		
- Logical level		X			
- Architectural level				X	
- Physical level					X
Essink (1986, 1988)					
- Object system modelling	x		X		
- Conceptual information system modelling		X		x	
- Data system modelling				X	
- Implementation modelling					X
Olle *et al.* (1988a)					
- Business analysis stage	X		X		
- System design stage				X	
- Construction design stage					X
Iivari (1989a)					
- Organizational level	X				
- Conceptual/infological level		X	X	x	
- Datalogical/technical level				X	X
Avison *et al.* (1996)					
-Human-activity view	X				
-Information view		X	X		
-Socio-technical view				X	X
-Human-computer interface view				X	X
-Technical view					X
Sol (1992)					
- Systelogical problems	X				
- Infological problems		X	x		
- Datalogical problems				X	
- Technological problems					X
Sowa *et al.* (1992)					
- Scope level	X				
- Enterprise model level	x		x		
- System model level		X	x	X	
- Technology model level					X
- Components level					X
van Swede *et al.* (1993)					
- Business perspective	X				
- Information perspective	X	x			
- Functionality perspective		x		x	
- Implementation perspective				X	X

(continues)

TABLE 20 (continues)

Frameworks	**Syste-logical**	**Info-logical**	**Concept-ual**	**Data-logical**	**Physi-cal**
Freeman *et al.* (1994)					
- World level	X				
- Conceptual level		X	X		
- Design level				X	
- Implementation level					X
ISO (1996)					
- Enterprise viewpoint	X				
- Information viewpoint		X	X		
- Computational viewpoint				X	X
- Engineering viewpoint					X
- Technology viewpoint					X

associated?). The perspectives are not clearly defined. As far as we interpret them right, they differ considerably from our perspectives[120]. The systelogical perspective in Welke (1977) addresses much larger part of reality than in our case (cf. 'object system' in Welke (1977) means "a view of something (real or abstract)" (ibid p. 149). The infological perspective in Welke (1977) covers also "personalization of the new solutions to the individual (e.g. type of channel, method of interaction, display more and format)" (ibid p. 156), that is to say, representational and individual-specific aspects, which are included in the IS datalogical and IS physical perspectives in our system of perspectives. The datalogical perspective in Welke (1977) corresponds to our IS datalogical and IS physical perspectives, though physical aspects are only vaguely visible in the definition of the physical perspective in Welke (1977). The conceptual issues are not explicitly discussed in the perspectives of Welke (1977).

Olive (1983**)** suggests a framework for the analysis of ISD methods, which is based on five levels of abstraction. The levels are: (a) external level (What information should the IS provide to the object system and what are the functions and input/output flows between them?), (b) conceptual level (What are the states of the object system, i.e. what to record into the data base?), (c) logical level (What are the operational requirements of the IS?), (d) architectural level (What is the overall architecture of the IS?), (e) physical level (What is the physical structure of the data base and the detailed structure of each process?). The external level covers the IS systelogical perspective and, to some extent, considers issues of the IS infological perspective. The conceptual level corresponds to our IS conceptual perspective. The logical level, architectural level and physical level correspond to our three "lower" perspectives, respectively.

120 Welke (1977, 162) mentions that the terms 'infological' and 'datalogical' have been borrowed from Langefors *et al.* (1975). He argues that their meanings are only slightly different from those of Langefors *et al.* (1975). On the bases of the definitions, we would like to disagree with him.

Essink (1986, 1988) presents the MADIS (A Modelling Approach for Designing Information Systems) framework, which is based on levels of abstraction and a set of aspects. The characterizations of the framework differ, to some extent, depending on a reference, so what follows is our synthesis of the presentations in Essink (1986) and Essink (1988). The framework consists of five levels of abstraction: (a) object system modeling (the IS is seen from the point of view of the organization, which is called the object system, but also "it answers the questions with regard to what phenomena in the real world information is needed", Essink 1986, 58), (b) conceptual information system modeling (the IS is seen as a set of functional components with which the information requirements should be fulfilled), (c) data system modeling (the IS is considered as a data processing system), (d) implementation modeling (the IS is viewed as a concrete system). Essink, who does not see any difference between the US_{IS} and the OS_{IS}, uses the conceptual IS modelling level to mean the IS infological perspective, including in it also aspects of user interface and user groups. Object system modelling stands for some aspects within the IS systelogical perspective and the IS conceptual perspective. The data system modelling level and the implementation modelling level are counterparts of our IS datalogical perspective and IS physical perspective, respectively.

Olle *et al.* (1988a) present a comprehensive framework for understanding information systems planning, business analysis and design[121]. The framework classifies design products and steps of the design process. It does not suggest any explicit levels of abstraction. However, it provides meta models on three levels corresponding to the stages in an IS life cycle: IS planning, business analysis and IS design. For each stage, data oriented, process oriented and behavior oriented perspectives are applied. Here, we examine how the two lower stages as well as construction design correspond to our perspectives. IS planning is omitted because it precedes the launching of an ISD project. Business analysis means the study of the existing state of affairs in a given business with the concepts such as organizational units, business activities, business events, flows of information / material set, and semantics of the information sets. System design means preparing a prescriptive statement about an IS. This includes e.g. tasks carrying out business activities triggered by menu selections and performed according to algorithms, constructs of the relational model, and reports and screen forms used as an interface. Construction design associates the designs with hardware and software environment. The business analysis stage corresponds to our IS systelogical and IS conceptual perspectives. The system design stage and the construction design stage stand for the IS datalogical perspective and the IS physical perspective, respectively. Interesting in the framework of Olle *et al.* (1988a) is that it addresses the infological aspects of the IS with only a few concepts (information/material set, flow) and also these concepts are used to exhibit

121 Construction design is also mentioned but not elaborated in the same way as the other parts of the framework.

what material/information are used in business activities, not in information processing.

Iivari (1989a) presents a conceptual framework for a "systematic recognition, comparison and synthesis of different perspectives on the concept of an information system". The framework is based on the following levels of abstractions: (a) organizational level defines the organisational role and context of the IS. It contains two parts: the designed organisational context and the application concept. (b) conceptual/infological level defines the "implementation independent" specifications of the IS. It embraces three parts: the universe of discourse ("part of reality of which the IS is concerned"), IS specifications, and user interface. (c) datalogical/technical level defines technical implementation for the IS. The organizational level is clearly the counterpart of our IS systelogical perspective. The conceptual / infological level corresponds to the IS conceptual and IS infological pespectives, except that Iivari (1989a) addresses issues of user interface on this level. To our view, considering user interface requires some decisions on allocations of IS actions into either a HIS part or a CIS part. These decisions should not be done until the IS datalogical perspective is applied. It should, however, be noted that for Iivari (1989a) the IS is the CIS (see Table 16). The datalogical /technical level corresponds to our IS datalogical perspective and IS physical perspective, but only for the CIS.

In their Multiview approach Avison *et al.* (1996) distinguish between five views on the IS. The views are: (a) human activity view (How is the IS supposed to further the aims of the organization using it?), (b) information view (What information processing function is the system to perform?), (c) socio-technical view (How can the IS be fitted into the working lives of people in the organization using it?), (d) human-computer interface (How can the individuals concerned best relate to the computer in terms of operating it and using the output from it?), and (e) technical view (What is the technical specification of the system that will come close enough to meeting the identified requirements?). The human activity view corresponds to the IS systelogical perspective. The information view addresses issues relevant to the IS infological and IS conceptual perspectives. The socio-technical view and the human-computer interface view focus on human roles, positions, actions and interaction with the CIS, meaning that they extend the IS datalogical and IS physical perspectives.

Sol (1992) considers the ISD from the viewpoint of problem solving. He distinguishes between four sub-problems related to way of thinking: (a) systelogical problems that are concerned with the modeling of an object system from an organizational approach, (b) infological problems that are concerned with the data structures and processing required to produce the data, (c) datalogical problems that concern the way and the form in which the data processing system is realized, and (d) technological problems that concern the technical resources used. The categorization of Sol (1992) comes closest to our system of perspectives. However, Sol sees no difference between the IS conceptual perspective and the IS infological perspective. Second, he considers datalogical and technological problems in relation to the CIS only.

Sowa *et al.* (1992) refine the information systems architecture, originally proposed by Zachman (1987), into the ISA framework with six columns and five rows. The rows correspond to views of different stakeholders (i.e. planner, owner, designer, builder, and subcontrantor). With an analogy to the construction of a building, Sowa *et al.* (1992) distinguish the following design levels: (a) scope (executives' view of the IS), (b) enterprise or business model (i.e. the IS from an operational view, as it would appear to the people who work with it in daily business routines), (c) system model (i.e. implementation-independent view of the IS that reveals data elements and functions that represent business entities and processes), (d) technology model (i.e. the view of a builder which shows how to implement the IS in some programming language or the like), and (e) components (i.e. detailed specifications of the IS for the programmer who codes individual modules). In the ISA framework, for each cell a set of concepts and relationships are also provided. Based on the characterizations of the levels and the concepts attached to them we can make the following observations. On the level of scope, aspects of the IS are considered from the IS systelogical perspective. The enterprise model level addresses aspects from the IS systelogical and IS conceptual perspectives. On the system model level the IS is viewed from the IS infological and IS conceptual perspectives. The two lowest levels correspond to the IS physical perspective.

van Swede *et al.* (1993, 535) present a framework for contingent information systems modelling. The framework consists of four perspectives and nine aspects. The perspectives correspond roughly to views of specific interest groups. The perspectives are: (a) business perspective (How is the business done?), (b) information perspective (What information supply is necessary to support the business?), (c) functionality perspective (What is the external behavior of the IS?), (d) implementation perspective (What is the internal functioning of the IS?). In the framework of van Swede *et al.* (1993) the organizational issues are especially emphasized. The business perspective and a major part of the functional perspective address issues that belong to the scope of our IS systelogical perspective. The functionality perspective covers issues of our IS infological and IS datalogical perspectives. Actually, van Swede *et al.* (1993) include also some implementational aspects (cf. "If users decide that implementation must be on a specific type of computer, such will be part of the functionality specified" (ibid p. 539)). The framework does not provide any perspective for conceptual aspects. The implementation perspective stands for our IS datalogical and IS physical perspectives.

Freeman *et al.* (1994) present a meta-model, called the Global System Model, of information systems to support reverse engineering. It is based on four levels of abstraction: (a) world level (represents a real-world view of the IS and functional areas of an organization), (b) conceptual level (represents an implementation-independent, abstract view of the IS), (c) design level (represents a functional decomposition of the computer-based information system, including user interface aspects), (d) implementation level (represents a physical view of the implemented software system). The world level with the concepts, such as user goal, resource, activity and agent, correspond to the IS systelogical perspective. The conceptual level addresses issues of the IS infological perspective (cf. process and data structure) and the IS conceptual perspective (entity, relationship, attribute). The design and implementation levels stand for the IS datalogical and IS physical perspectives, covering the CIS and user interface.

A joint standardization effort of the ISO and ITU-T (International Telecommunication Union) has resulted in the Reference Model for Object Distributed Processing (RM-ODB) (ISO 1996) that comprises a framework for specifying architectures for distribution, interoperability and portability of applications based on the object-oriented technology. The reference model divides the application specification into five parts, corresponding to five different, but related and consistent viewpoints. The viewpoints are: (a) enterprise (e.g. business, roles and policies), (b) information (What does the system deal with?), (c) computational (What does the system do?), (d) engineering (How is the system distributed?), and (e) technology (How is the system implemented?). In the reference model, the technical aspects of the CIS are emphasised. The enterprise viewpoint is focused on the issues relevant in the IS systelogical perspective. The information viewpoint contains elements of the IS infological and IS conceptual perspectives. The three lower viewpoints consider computational objects, sequences of their distributed interaction and the choice of technology to implement that interaction. Their counterpart in our system of perspectives is primarily the IS physical perspective.

We can draw the following conclusions from the above analysis. The frameworks provide 3 – 5 levels, views, viewpoints, sub-problems, or perspectives for the categorization of the aspects of the IS. For simplicity, we call them the levels in the following. In the frameworks the upper levels are more US-related and the lower levels are more technology related. The levels between the extreme ends are defined using expressions with "independence" from something (e.g. from "the object system" in Olive (1983), and from the technology in Iivari (1989a)). There are some differences in the emphases and focuses of the frameworks. In the frameworks of Welke (1977), van Swede *et al.* (1993) and Olle *et al.* (1988a) the emphasis is clearly on the upper levels. They use fine-grained levels for perceiving business issues and only one or no level to address technological issues. The levels are supposed to be mainly applied in the top-down order, although no explicit statements are given for that, except in van Swede *et al.* (1993)[122]. IS requirements are commonly (e.g. Iivari 1989a; Essink 1986; Essink 1988; Olive 1983) included in the topmost level, except in van Swede *et al.* (1993) which provides a specific perspective, called Information

122 Welke (1977, 150) considers the top-down "approach" to be a natural ordering but does not exclude any other order, provided that all the perspectives are applied.

perspective, for that purpose. Olle *et al.* (1988a) provides hardly any concepts for describing infological aspects of the IS.

Conceptual issues are included in the topmost levels (as in Essink 1988, and partly in Olle *et al.* 1988a) or as is the case more commonly, in the next lower level. The framework of Olive (1983) is the only one, which provides a special level for perceiving the conceptual aspects of the IS. van Swede *et al.* (1993) seems not to consider IS conceptual issues at all. Iivari (1989a), Avison *et al.* (1996), Freeman *et al.* (1994), Sowa *et al.* (1992) and Olle *et al.* (1988a) pay special attention to user interface. Essink (1988) mentions it only incidentally, and the frameworks of Welke (1977) and Sol (1992) are too general to recognize it.

The frameworks, which contain a small number of levels, such as Iivari (1989a), combine different aspects into single levels, with the result that the criteria for the levels are not clear-cut. In some frameworks, the linguistic – conceptual dimension is not fully recognized. In van Swede *et al.* (1993) it is totally ignored. There are also differences in the numbers of perspectives covering the realization dependence - independence dimension.

In our view, it is important to have separate perspectives for each set of different aspects of the information system. Therefore, the IS systelogical perspective is needed to consider the IS in relation to the US. The IS conceptual perspective is necessary to address the conceptual contents of the IS objects. Unlike Avison *et al.* (1996), Sowa *et al.* (1992), Sol (1992), Freeman *et al.* (1994) and ISO (1996), we see it vital to clearly differentiate between the infological perspective that represents the "linguistic world", and the IS conceptual perspective which stands for the "conceptual world". On the realization-independence dimension, at least three related perspectives can be clearly distinguished. The first one (infological) is independent from representational and realization-dependent aspects. The second one (datalogical) is independent from realization-dependent aspects. The third one (physical) recognizes all the concrete issues related to a specific realization. This last perspective, namely the IS physical perspective, can be further divided, if necessary, into more specific perspectives (cf. the OSI reference model by ISO (1984) and the RM-ODP framework (ISO 1996)). That we have not thought necessary in this work.

6.4.3 In-Depth Analysis of the Concepts in the Perspectives

In this section we examine how the frameworks define and present the concepts and their relationships contained by the perspectives. First, we divide the selected frameworks into three classes:

- *Class 1.* Frameworks that provide comprehensive sets of defined concepts and relationships, organized into levels and represented in meta models.
- *Class 2.* Frameworks that provide large sets of concepts and possibly relationships that are defined, or at least characterized, and organized into levels.

- *Class 3*. Frameworks that provide the definitions of the levels but for the levels they provide only some concepts, possibly with no definitions.

From the eleven analyzed frameworks, four belong to the first class (Iivari 1989a; Olle *et al.* 1988a; Sowa *et al.* 1992; Freeman *et al.* 1994). Iivari (1989a) and Olle *et al.* (1988a) provide the most comprehensive sets of concepts and relationships, organized according to the levels / the stages and represented in meta models. The concepts are clearly defined, and in Olle *et al.* (1988a) they are also illustrated by examples. Two other frameworks in this class, Sowa *et al.* (1992) and Freeman *et al.* (1994), also present meta models of the concepts and relationships within and between the levels but the sets of the concepts are not as large and the definitions of the concepts not as clear and explicit as in the other frameworks in this class.

The frameworks of Essink (1986, 1988), van Swede *et al.* (1993) and ISO (1996) belong to the second class. Essink (1986, 1988) provides a large set of concepts but in a rather unstructured manner. Only essential concepts are defined. No explicit treatment of relationships and no meta model are provided. The set of the concepts in van Swede *et al.* (1993) is somewhat smaller, but as to the other aspects the framework is comparable to Essink (1986, 1988). ISO (1996) gives strict definitions for most of its concepts but makes only a superficial examination of the relationships.

Three of the analyzed frameworks, namely Welke (1977), Olive (1983) and Sol (1992), belong to the third class, meaning that they provide the definitions of the perspectives but do not do more than mention some of the essential concepts. We have also included Avison *et al.* (1996) in this third class, because though the number of the concepts is large they are not treated as a framework that would make them comparable to the others.

To get a yet more detailed view on the frameworks, we make an in-depth analysis of the four frameworks included in the Class 1. In Table 21 the concepts of the frameworks are presented for comparison both with one another and with our concepts. The concepts are classified into five categories according to our perspectives. The numbers 1-4 attached to the concepts in the four

Legend in Table 21.

Olle *et al.* (1988a):	Iivari (1989a):
1. Business analysis stage	1. Organizational level
2. System design stage	2. Infological / conceptual level
	3. Datalogical / technical level
Sowa *et al.* (1992):	Freeman *et al.* (1994):
1. Business model	1. World level
2. Information systems model	2. Conceptual level
3. Technology model	3. Design level
4. Implementation level	

TABLE 21 Comparative classification of the concepts of the frameworks according to the perspectives

IS Perspective	**OntoFrame**	**Olle *et al.* (1988a)**	**Iivari (1989a)**	**Sowa *et al.* (1992)**	**Freeman *et al.* 1994)**
IS Systelogical	US purpose US environment US organization US org. unit US position US human actor US role User US action US rule US object US tool US resource	Information/ material set (1) Org. unit (1) Business activity (1) Activity precondition (1) Flow (1) Flow precondition (1) Business event (1) Business event condition (1)	Org. actor (1) Org. position (1) User (1) Creator/recipient (1) Org. channel (1) Org. function (1) State/stores (1) Material/information (1) Work procedure (1) Instrument (1) IS service (1) Event (1) Situation (1) Trig. condition (1) Org. act (1) IS use act (1)	Business function (1) Business process (1) Business event (1) Business plan (1) Business objective (1) Business resource (1) Business product/service (1) Business policy (1) Business rule (1) Business work unit (1) Job position (1) Business location (1) Business channel (1) Business exchange (1) Org. unit (1)	Goal (1) User goal (1) Org. unit (1) Activity (1) Resource (1) Agent (1) External institute (1) External event (1)
IS Infological	IS purpose IS action IS rule IS event IS condition IS object	---	Information base (2) Information type (2) IS function (2) Input inf. type (2) Output inf. type (2) State inf. type (2) Derivation rule (2) IS process (2) State (2) State transition (2) Event (2)	System process (2) System objective (2) System plan (2) System event (2) Information req. (2) User view (2)	Action (2) Data structure (2) Process (2) Internal event (2)

(continues)

TABLE 21 (continues)

Perspective	OntoFrame	Olle *et al.* (1988a)	Iivari (1989a)	Sowa *et al.* (1992)	Freeman *et al.* (1994)
IS Conceptual	OS_{IS} construct Entity OS_{IS} Relationship Entity role Attribute value OS_{IS} state OS_{IS} transition OS_{IS} event OS_{IS} behavior	Entity type (1) Relationship type (1) Attribute (1) Attribute group (1) Value constraint type (1)	Entity type (2) Event type (2) Time interval type (2) Association type (2) Attribute (2) Consistence rule (2) Action/operation type (2) Event (2) State (2) Trig. condition (2) Action/operation (2)	Business subject area (1) Business fact (1) Business entity (1) Business relationship (1) Business property (1) Business identifier (1) Data entity, Data entity rel. (2) Data attribute, Data domain (2) Data identifier (2) Data subject area (2) Object rule (2) Integrity constraint (2)	Entity (2) Relationship (2) Attribute (2) State (2)
IS Datalogical	HIS purpose HIS action HIS rule IS role Data object Dialog Window UI component UI state UI transition UI event CIS action CIS rule Algorithm Transaction	Table (2) Column (2) Row (2) Constraint (2) External form (2) Display option (2) Data type (2) Task (2) Access control (2) Menu (2) Menu hierarchy (2) Algorithm (2) System event (2) Condition (2)	Interface object type(2) Representation (2) Interface operation type (2) Interface act (2) Abstract machine (3) Abstract data com-munication link (3) Input data type (3) Output data type (3) Algorithm (3) Derivation rule (3) Event (3) Control state (3) Control rule (3) Data process (3) Control state transition (3)	Work group (2) User role (2) System interaction (2) Processing site (2) Communication link (2) Human interface architecture (2) System process (2) System plan (2) System event (2) Information req. (2) User view (2)	Function (3) Dialog structure function (3) Dialogue function (3) Object (3) User interface object (3) User (3)

(continues)

TABLE 21 (continues)

Perspective	**OntoFrame**	**Olle** ***et al.*** **(1988a)**	**Iivari** ***et al.*** **(1989a)**	**Sowa** ***et al.*** **(1992)**	**Freeman** ***et al.*** **(1994)**
IS Physical	Data storage Data file Data base Data message Node Memory device Processor HW architecture Application SW SW component SW architecture	---	Database / file (3) Software / program component (3)	Database/file (3) Data record (3) Data element (3) Key (3) Record relation (3) Physical domain (3) System rule (3) Transaction (3) Constraint mechanism (3) Stored procedure (3) Dialog/batch job (3) Application system (3) Service plan (3) Service objective (3) System product (3) Device format (3) System user (3) User interface (3) User profile (3) System resource (3) Communication line (3) Security architecture (3) Component architecture (3) Interruption (3) Timing definition (3)	Module (4) Data (4)

frameworks indicate the stages (Olle *et al.* 1988a) or the levels (Iivari 1989a; Sowa *et al.* 1992; Freeman *et al.* 1994) to which the concepts belong in the original frameworks (see the legend below). In the following, we first make some general remarks on the criteria used and decisions made in selecting and classifying the concepts. After that we bring out our findings and conclusions from this in-depth analysis.

We have selected all those concepts in the frameworks which are essential, are treated as "entities with independence", and are metamodeled in the frameworks. This implies that we have ignored concepts that appear only in the texts or are relationships in the meta models. The reason for this policy is our aim to keep the lists of concepts short enough to enable their comparison in a reasonable space. From the framework of Olle *et al.* (1988a) we have included the concepts of only two stages (business analysis, systems design), because the concepts of the construction design stage are not metamodeled. In the framework of Sowa *et al.* (1992) the detailed meta model covers only three columns and three rows (levels), originally introduced in the information system architecture by Zachman (1987). Although we could have tried to divide the concepts of three original rows into five levels of the extended framework (Sowa *et al.* 1992), we decided, in order to avoid difficulties and risks in subjective interpretations, to use the concepts of the three rows of the original framework, extending the structure with some concepts suggested informally by Sowa *et al.* (1992) for the three new columns (Motivation, People and Time). From our perspective ontology, we have only included concepts of the CIS in the IS physical perspective, to follow the view adopted also in the other frameworks. We want, however, remind that the IS physical perspective equally covers the concrete concepts of the HIS.

Olle *et al.* (1988a) and Iivari (1989a) divide the concepts on each stage / level into three categories: in Olle *et al.* (1988a) according to data, process and behavior perspectives, and in Iivari (1989a) according to structure, function and behavior abstractions. In Sowa *et al.* (1992) the concepts are categorized into six context-based aspects. These kinds of sub-categorizations can be expected to have positive impact on their coverage in terms of contextual domains, as compared to the framework of Freeman *et al.* (1994), which does not contain any categorization. In contrast to the others, Iivari (1989a) also provides specific instance-level concepts (e.g. Org. act, Action/Operation, IS process), in addition to the type level concepts, within the behavior abstraction. We have included these concepts in the table, although they are not comparable to the others.

After the general remarks on the frameworks we present findings and conclusions from the analysis (in the order of the IS perspectives). The scope of the IS systelogical perspective is addressed in various degrees in the frameworks. In Olle *et al.* (1988a) only three contextual US domains are addressed, namely the US actor, US action and US object domains. The framework clearly applies the information/material flow approach to modeling functional features of the utilizing system. Iivari's (1989a) framework contains the concepts of the US actor, US action, US object, and US facility domains. US objects are considered on a general level and from the IS point of view (cf. IS services). In addition to those presented in the table, Iivari (1989a) defines the notions of organizational channel, situation and input/output flow. Sowa *et al.* (1992) provide the largest set of concepts from the IS systelogical perspective, covering all the US domains (the time domain is implicitly covered). In the framework the US objects are addressed in two ways, first through the notions of business resource and business product/service, and second through the linguistic and conceptual notions (e.g. business fact, business subject area, business entity). The latter notions belong to the IS conceptual perspective. Freeman *et al.* (1994) suggest a framework which contains only eight concepts but cover four US domains. The notion of resource is not defined, actually not even mentioned in the text although included in the meta model, but concluding from its relationships with other concepts in the Global System Model the notion corresponds to US objects and US resources in our terminology. The framework also introduces the notion of external institute that has no counterpart in OntoFrame. The US tool-specific concepts are not included in any framework, except in Iivari (1989a) and in OntoFrame.

The IS infological perspective is not covered at all in Olle *et al.* (1988a)[123]. Iivari (1989a) applies the information flow approach to exhibit the functional nature of the IS and the state-transition approach to describe the IS behavior. In addition he introduces the instance-level notion of IS process, but no concepts for the IS purpose domain. Sowa *et al.* (1992) provides notions for the IS purpose domain (system plan, system objective, information requirement), the IS action domain (system process) and the IS object domain (user view). The framework of Freeman *et al.* (1994) contains only four notions addressing the IS action domain and the IS object domain.

For the IS conceptual perspective, all four frameworks in the literature provide the structural notions of entity (type), relationship (type) and attribute/property. In Sowa *et al.* (1992), these concepts are defined and applied at two levels: at the general level (business model) the framework operates with business entities, relationships and properties, and at a more detailed level (information system model) with the notions of e.g. data entity, data entity relationship and data attribute. Dynamic features of the object system are only addressed in Iivari (1989a), Sowa *et al.* (1992) and OntoFrame. It should be noted that in OntoFrame the concepts are at the instance level whereas in the other frameworks they are at the type level.

For the IS datalogical perspective, Iivari (1989a) provides the most comprehensive set of concepts. Due to the fact that the IS is seen as a CIS in the framework, the concepts are technology oriented. Human part is assumed to be included in the host system (cf. the IS systelogical perspective). Also in Olle *et al.* (1988a) the HIS is not addressed. Sowa *et al.* (1992) do not distinguish

123 In Olle *et al.* (1988a) the only concepts referring to infological aspects of the IS are Information/Material set and Flow, but they are used to exhibit what information/material the business processes use, not to describe what information the IS processes.

between the IS infological and IS datalogical aspects of the IS. Therefore we here include the concepts contained by the IS infological perspective as well as the concepts related to the network and agent aspects (as they are known in the information system architecture). Freeman *et al.* (1994) present only some concepts that belong to the IS datalogical perspective, and none of them refer to the CIS. All the frameworks provide concepts related to the user interface.

The concepts of the IS physical perspective are totally missing in the framework of Olle *et al.* (1988a), resulting from the fact that the construction design is not included in this analysis. Iivari (1989a) and Freeman *et al.* (1994) provide only two specific concepts for this perspective. The technology model in Sowa *et al.* (1992) contains a large set of concepts related to most of the contextual domains within the IS physical perspective.

To summarize, Olle *et al.* (1988a) provide the least concepts, and the concepts from the IS infological and IS physical perspective are totally missing from this framework. Freeman *et al.* (1994) define a slightly greater number of concepts, covering several IS perspectives and IS domains. In the framework of Iivari (1989a) all other IS perspectives, except the IS physical perspective, are well addressed. Sowa *et al.* (1992) provide most concepts, covering all the IS perspectives and most of the IS domains.

How does OntoFrame compete with the others in the light of the comparative analysis? Due to its dimensional structure, OntoFrame applies, for each IS perspective, the predefined contextual domains, thus guaranteeing that the largest possible coverage of relevant issues is reached and the perspective-specific structures of the concepts can be easily interrelated. This becomes evident when comparing the nature and number of concepts within each perspective in Table 21. The clearly defined perspectives serve as a definitional skeleton allowing to decide on which IS perspective each concept belongs to. With the uniform contextual structure underlying the concepts it is possible to relate the concepts of different IS perspectives to one another in a consistent way. This helps moving from one IS perspective to another during the ISD. We have derived the conceptual contents of the IS perspectives from the contextual domains of the context ontology. To avoid the unnecessary redundancy with Chapter 4, we have included in the IS perspectives only the most essential concepts. From the five perspectives we have here concentrated on the IS systelogical perspective, the IS infological perspective and the IS conceptual perspective. For the other IS perspectives we have mainly provided only some examples of the essential concepts. Despite these limitations our IS perspectives were found to reflect more contextual features of the IS than most of the analyzed frameworks.

6.5 Summary and Discussions

In this chapter we first defined the perspective ontology. It is a domain-specific ontology, which provides concepts and constructs for conceiving, understanding, structuring and representing things in information processing contexts from pre-defined perspectives. According to the ontology, perspectives constitute a system with strictly defined relationships along one or more dimensions. Our ontology comprises five specific perspectives (i.e. the systelogical perspective, the infological perspective, the conceptual perspective, the datalogical perspective, and the physical perspective) that are established along three orthogonal dimensions (i.e. the decomposition dimension, the semiotic dimension, and the realization independence-dependence dimension). The systelogical perspective views the IS in relation to its utilizing system. Applying the infological perspective the IS is seen as a functional structure of information processing actions and informational objects. The conceptual perspective reveals the semantic contents of the IS objects. The datalogical and physical perspectives provides concepts and constructs for viewing representation-specific and implementation-specific features of the IS, respectively.

Second, we specialized the perspective ontology onto four processing layers and characterized the corresponding contexts on each layer from all five perspectives. In addition, we elaborated the characterizations of the IS perspectives by defining a large set of IS concepts from each of the perspectives. The same will be done for the ISD perspectives and the ME perspectives in Chapters 8 and 10, respectively. While the IS domains, i.e. the first part of the IS ontology, are assumed to be implicitly derived from the contextual domains (cf. Chapter 4), the IS perspectives defined here constitute the second part of the IS ontology.

Third, we conducted a comparative analysis of IS perspectives suggested in the IS/ISD literature. For the analysis we selected eleven frameworks containing clearly defined perspectives. The analysis was carried out in three parts. In the first part we made an overview of the frameworks, their perspectives and criteria used in establishing perspectives. The overview showed that the sets of perspectives are quite divergent and the criteria, if explicated at all, are more or less ambiguous. In the second part we compared the sets of perspectives in eleven frameworks with one another and with the perspectives in our ontology. In conclusion, we stated that there are several differences in the emphases and focuses of the frameworks. The frameworks of Welke (1977), van Swede *et al.* (1993) and Olle *et al.* (1988a), for instance, clearly emphasize the "upper" perspectives. They use fine-grained perspectives for conceiving business issues and only one or no perspective to address technological issues. The perspectives are commonly supposed to be mainly applied in the "top-down" order. IS requirements are mostly (e.g. Iivari 1989a; Essink 1986; Essink 1988; Olive 1983) included in the topmost perspective. Olle *et al.* (1988a) provide hardly any concepts for describing infological aspects of the IS. Conceptual issues are either included in the topmost perspective (as in Essink 1988), in the next lower perspective (as in Avison *et al.* (1990) and Freeman *et al.* (1994)), or ignored (as in van Swede *et al.* (1993)). The numbers of perspectives vary between 3 and 5. Those suggesting three perspectives (e.g. Iivari 1989a) combine different aspects of the IS into single perspectives, resulting in that dimensions and criteria are not clear-cut anymore.

For the third part of the analysis we selected those comprehensive frameworks in which the concepts are explicitly defined and modeled in meta models. The frameworks are: Iivari (1989a), Olle *et al.* (1988a), Sowa *et al.* (1992) and Freeman *et al.* (1994). The aim of this in-depth analysis was to investigate what contextual concepts each of the perspectives of the frameworks contains. We used the IS perspectives and the contextual domains of OntoFrame as the baseline for the analysis. The analysis clarified and detailed the views obtained from the other parts of the analysis.

In this work we have pursued an ontology, which is general enough to cover the most common principles to establish and apply viewpoints for conceiving information processing contexts on four processing layers. To demonstrate its applicability in viewing the IS, we specified the IS perspectives that comprise dozens of IS concepts and IS constructs. The number and scope of the concepts of the perspectives are much larger than in any framework analyzed, especially when taking into account that besides those presented in the meta models in this chapter a lot more concepts can be easily derived from the context ontology. More important than the number of the concepts is the degree to which our ontology is able to reflect contextual features of the IS. In this respect, the context ontology was found more coverable than the other frameworks. We have put an emphasis on the IS systelogical, IS ontological and IS conceptual perspectives. More work is needed to specify additional concepts for the other IS perspectives. Likewise, the relationships between the IS perspectives could be specified on a more detailed level with an intensive use of the intra-domain and inter-domain relationships defined in Chapter 4.

The perspective ontology is useful in many respects. Besides being specialized at the IS layer in this chapter, it can be specialized at the ISD layers and at ME layers, as we will show in Chapters 8 and 10. The resulting ISD perspectives will be used to categorize and relate the aspects of ISD to better manage the complexity related to ISD. The ISD perspectives also provide the conceptual basis for specifying the contents and structure of an ISD method. At the ME layer, the specialized perspectives organize an ME effort into an array of logically related stages, thus enabling the planning and execution of a process of method engineering in a well-structured manner.

7 MODEL LEVEL ONTOLOGY

In any context encompassing complex problem solving, a human being tends to activate and incorporate epistemological and analytical constructs that can help him/her conceive, analyze, design and implement a solution to the problem at hand. If one expects the same kind of context to appear repeatedly, a conceptual machinery is set up to refine and integrate those concepts and constructs that are experienced helpful in those contexts. These machineries are models.

The purpose of this chapter is to define the model level ontology that is the fourth one among the contextual ontologies. The model level ontology provides concepts and constructs for conceiving, understanding, structuring, and presenting things in reality in terms of models within a system of model levels. The concepts and constructs have been derived from the context ontology, the layer ontology and the perspective ontology (see Figure 64). The model level ontology is an essential ingredient of OntoFrame, which is aimed to support the analysis, design and implementation of the ISD methods.

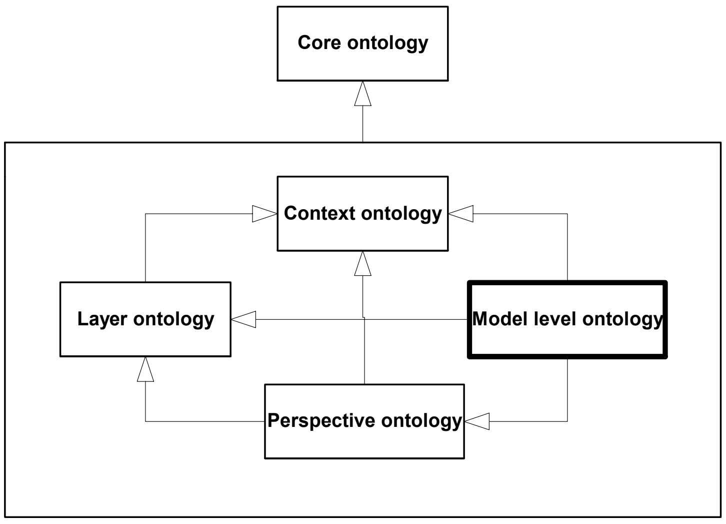


FIGURE 64 Focus of Chapter 7

The chapter is structured into four sections. First, we define the notions of model and modeling, and present the main classifications of models. Second, we extend our consideration to concern models at different levels. We also derive classifications of models and meta models based on the contextual domains and the perspectives. Third, we present a comparative analysis of conceptions about systems of levels suggested in the ISD literature. Fourth, we examine how the contextual ontologies are related to one another. We can do this here because all the contextual ontologies have now been specified. The chapter ends up with a summary and discussions.

7.1 Model and Modeling

Models serve as means to gain knowledge about relevant things. This is what researchers, not only in the ISD field but also in nearly every branch of science, agree on. Instead, conceptions about the nature, basis and form of a model greatly diverge from one another. In this section, we first give some examples of conceptions presented in the literature to bring out features that are seen important in the notion of a model. Second, we present our definitions of basic concepts related to a model and modeling. Third, we elaborate more specialized concepts based on the ontologies defined in the previous chapters.

7.1.1 Basic Concepts

The notion of a model applies to many kinds of phenomena, e.g. things, styles or even persons (cf. Webster 1989). A model can be a small copy of a ship or a building, or it can be a preliminary representation of something, serving as the plan from which the final object is to be constructed. It can be a piece of sculpture in wax from which a finished work in bronze is to be made. Further, it can be a style or design of a particular product (cf. a car model), or a person who poses for an artist or who is employed to display clothes by wearing them. Finally, it can be a generalized, hypothetical description used to analyze or to explain something. The terms like 'example', 'pattern', 'architype' and 'standard' are often used as synonyms for a model in a common language.

When considering various definitions given for a model in the literature, we can recognize three issues on the basis of which their meaning can be analyzed and compared. The first issue concerns the purpose of a model. Minsky (1965) observes that a model is a thing, which can answer certain questions about some other thing for a certain questioner. In the Frisco Report (Falkenberg *et al.* 1998, 55) a model is defined as "a purposely abstracted, clear, precise and unambiguous conception". According to Rosemann *et al.* (2002, 78) a model is …"created for the purpose(s) of a subject". The second issue concerns the relationship between a model and modeled phenomena. For instance, van Gigch (1991, 91) argues that a model "stands at one level of abstraction higher than the systems from which the properties and attributes are obtained". According to Wijers (1991, 6), a model "is a simplified, stylised representation of a system, abstracting the essence of the system's problem studied". Yourdon (1989), Avison *et al.* (1990, 452) and Firesmith *et al.* (1999, 31) also refer to abstraction in stating that a model is used to 'highlight' or emphasize certain critical features of a system, while simultaneously de-emphasizing other aspects of the system. The third issue concerns the nature of a model. Some researchers regard a model as a conceptual thing. Jayaratna (1994, 242), for instance, concludes that a model "is a complete and coherent set of concepts, which can underpin our understanding and actions". Some others argue that a model is a linguistic thing. Krogstie (1995, 476), for instance, defines a model to mean "an abstraction externalized in a language". Rosemann *et al.* (2002) define a model to be "a representation of a relevant part of the real world".

The issues discussed above reflect three viewpoints to the notion of a model. The viewpoints are teleological, semantic and semiotic. We argue that a definition of a model should always highlight aspects from these three viewpoints. What we do next is present a general definition for the notion and then specify it further from the viewpoints. Generally speaking, a *model* is a thing that is used to help or to enable the understanding, communication, analysis, design, and/or implementation of some other thing to which the model refers. To specify it further, we first say that a model is always produced for some specific purpose (Teleological viewpoint). Its value comes about from the benefits it brings to its users. It may help the users better understand reality, design options for changes, foresee consequences of changes, reason on information and knowledge carried by the model, etc. (cf. Kangassalo 2002, VI). Second, a model can be seen as a perception and an abstraction of certain things in reality (Semantic viewpoint). Perception and abstraction are enacted and guided by the intended purposes and the applied point of view (cf. the UoD in Section 3.8). Third, a model can appear in one of three forms, namely as a conceptual construct, as a linguistic expression, or as a physical construct (Semiotic viewpoint).

Implied from the semiotic viewpoint, we distinguish between a concept model, a model denotation and a physical model[124] (see Figure 65). A *concept model* is composed of concepts and conceptual constructs referring to certain things in reality. To enable the communication about a concept model, it has to be represented in some language. A precise and unambiguous representation of a concept model in some language is called a *model denotation* (cf. Falkenberg *et al.* 1998, 55). A *physical model* consists of physical parts, which, as an organized whole, resemble some other thing(s) (e.g. small copies of airplanes or ships). The physical models are also called the empirical models or the analog models. To differentiate the physical models from the others, we call the other models the linguistic models. In this study we consider only the linguistic models.

[124] This arrangement is revised from the taxonomy originally suggested by Bertels *et al.* (1969) and Dietz (1987).

When it is not necessary to make the distinction between a concept model and a model denotation, we use the generic term 'model'.

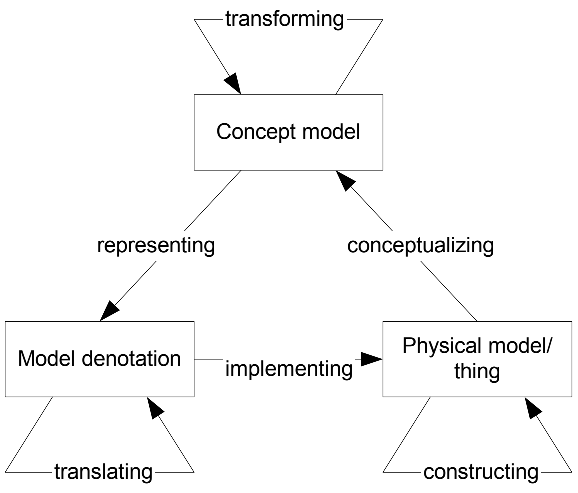


FIGURE 65 Main types of models and modeling actions

Figure 65 also presents the main actions of producing models of three kinds (cf. Dietz 1987; Brinkkemper 1990). The physical models are *constructed* from the physical things by moulding, building or engineering. The concept models can be produced in two ways: (a) by perceiving and *conceptualizing* the relevant features of the physical thing(s), or (b) by *transforming* from some other concept model(s). The model denotations are produced (a) by *representing* concept model(s) by signs of some language, or (b) by *translating* them from some other model denotation. A model denotation can be *implemented* as a physical model. An example of the implementation of a model is a CASE tool. The whole process of yielding an externalized model denotation for a certain purpose is called *modeling.*

7.1.2 Classifications of Models

The models are classified in numerous ways in the literature. Shubik (1979), for instance, distinguishes between verbal, analytic or mathematical, iconic, pictorial or schematic, and simulation models. Gigch (1991, 125) divides the models into explanatory, hypothetical, experimental, predictive, innovative, and epistemological models. Our intention is to produce much more comprehensive and structured classifications. We distinguish between three contexts that are related to a model. The contexts are: (a) the modeling context, (b) the modeled context, and (c) the model utilizing context[125]. For instance, a model of an electric system for a building (the modeled context) is designed by an engineering company (the modeling context) to support the construction

[125] This division into three kinds of contexts corresponds to the categorization into the epistemological, ontological, and social context questions related to the paradigmatic assumptions of data modeling in Hirschheim *et al.* (1995, 156-157).

work of the electric system (the model utilizing context). In the above case, the contexts are mainly separate. In information system development (the modeling context), a model is built about an information system (the modeled context) with the aim to guide information processing in the information system (the model utilizing system). In this case, the contexts are more coupled with one another. In the following we consider the contexts one by one in order to recognize and define more refined concepts related to a model (see Figure 66).

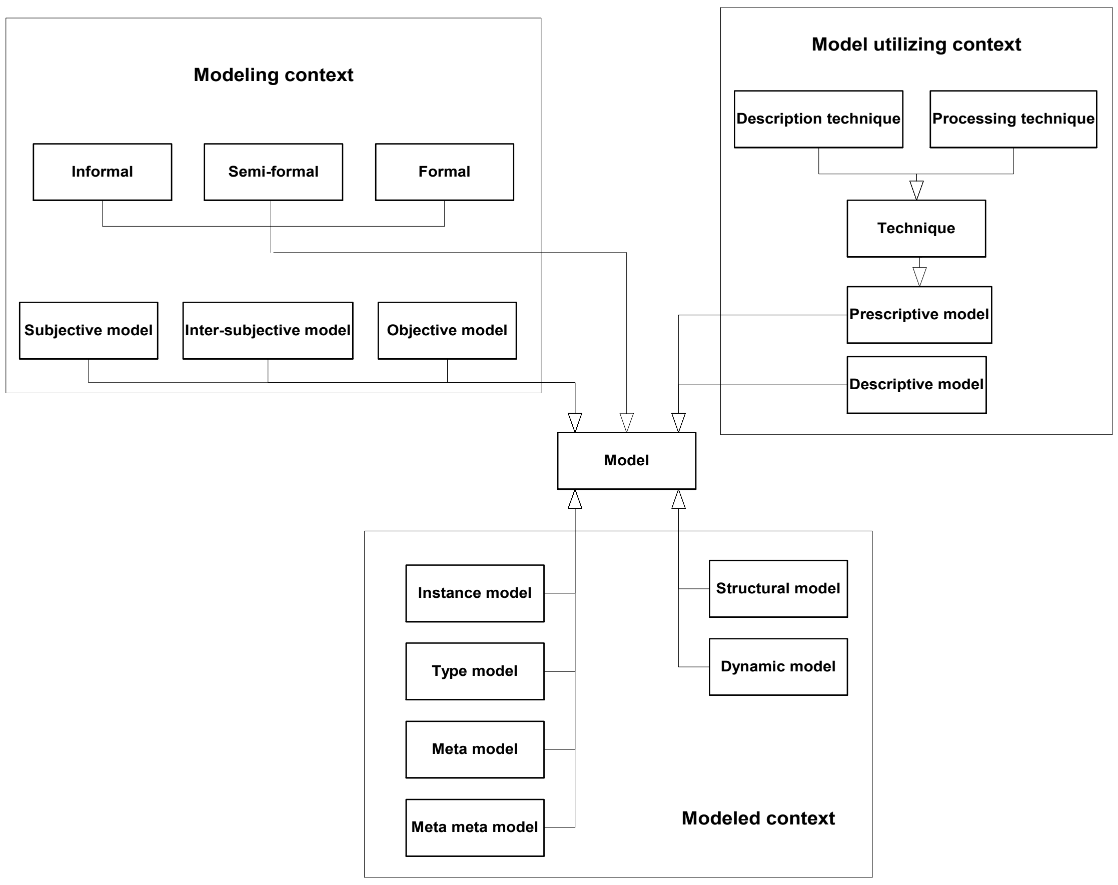


FIGURE 66 Classifications of the models within three contexts

The *modeling context* means a context the purpose of which is to produce a model for a model utilizing context. We already recognized three types of models and six types of processes of modeling. On the basis of the language used to represent a model, we categorize the models into informal models, semi-formal models, and formal models (cf. the categorization of the languages in Section 3.6). *Informal models,* also known as free models (Wijers 1991, 15), are restricted in their structure only by the modeler's imagination. Typical examples are verbal and pictorial models (e.g. rich pictures in Checkland 1981). Informal models are less accurate but more flexible and subtle than the other models. *Semi-formal models,* like diagrams, tables, matrices and structured texts, are constrained by the syntax of the language(s). *Formal models* are represented in a formal language, such as a programming language, or by logical and mathematical constructs with rigorously defined syntax and semantics. Independently of the level of formality, the models can be classified by their presentation style into graphical, matrix, textual, tabular, or form based models. The more formal the models are, the larger the portion is for which the computerized tools can serve as support in manipulation and reasoning from them.

On the basis of actors involved in modeling, we can distinguish between subjective models, inter-subjective models, and objective models. As a single actor conceives reality in an individual manner, it is natural that the outcome of modeling process reflects the modeler's subjective conception about the subject matter. Depending on an abstraction level and a language used in modeling, *subjective models* about the same things in reality can substantially diverge from one another. Modeling through sharing the conceptions within a community and negotiating on them yields an *inter-subjective model.* In some very rare situations, we can consider a model to be *objective* in the sense that there is no room for differing interpretations (e.g. a formal model of the Euclidean space).

The *modeled context* is a context which a model is about. The models can be categorized here according to which kinds of concepts they are composed of. First, the models are categorized into structural models and dynamic models. *Structural models* are composed of concepts that refer to static phenomena in the modeled context. The structure may concern information (e.g. ER model (Chen 1976)), a social organization (e.g. organisational chart (Ouchi 1981)), software system (e.g. application architecture (Booch *et al.* 1999)), hardware, or any other part of the organizational infrastructure. *Dynamic models* are composed of concepts that refer to the behavior in, or the evolution of, the modeled context (e.g. DFD model (Yourdon *et al.* 1979); action diagram (Jackson 1983); activity diagram (Booch *et al.* 1999)). The concepts widely used in the dynamic models are activity, process, event, trigger, and state transition (cf. Sol *et al.* 1992).

Second, we can apply abstraction by classification to divide the models into instance models and type models[126]. An *instance model* is a model, which is mainly composed of concepts that are instances of the concepts of the other model, the *type model.* For example, an ER model contains concepts like Entity type, Relationship type and Attribute. The corresponding instance concepts, included in an instance model (called an ER schema), may be Person, Marriage, and Age. There are also models that consist of concepts that are types of the concepts in a type model. We will consider them in Section 7.2.

Modeling is guided by the knowledge on the intended utilization of the resulting model. Thus, the *model utilization context* is of vital importance to modeling. Modeling may aim, for instance, to help us understand complex phenomena in reality, to design changes in reality, to foresee consequences of those changes, etc. Modeling may also aim, through mathematical, statistical,

126 The Frisco Report (Falkenberg *et al.* 1998, 57) uses the terms 'extensional' and 'intensional' models, correspondingly.

axiomatical, etc. machineries used in the models, to facilitate translatability, interpretability, verifiability, optimizability, traceability, etc. of the models.

Based on the primary purpose of the utilization, models can be divided into descriptive models and prescriptive models (cf. McChesney 1995; Pohl 1996; Ramesh *et al.* 2001; Rossi *et al.* 2003)[127]. *Descriptive models,* like traceability models, are used to portray or predict the relevant features of the modeled context to support the analysis of the existing reality or the design of the future reality. *Prescriptive models* are conceived to be sets of normative statements, which specify what is permitted, forbidden or obliged in certain situations. The prescriptive models can comprise plans, rules and/or commands (cf. Section 4.6.4). There is an important ontological difference between the descriptive models and the prescriptive models. The descriptive model should match, at a given level of abstraction, the modeled context. If it fails to do so, the model is false. In contrast, if the matching between the prescriptive model and the modeled context fails, corrective actions are required to get the modeled context to fit the prescriptive model. The descriptive models can be further divided into explanatory, hypothetical, experimental, predictive, innovative, or epistemological model in the dimension of knowledge inquiry and acquisition (Gigch 1991, 125).

A prescriptive model aimed to guide the behavior in the modeled context is called a *technique.* It is a precise programme of action leading to a desired result (cf. Checkland 1981). There are two kinds of techniques: description techniques and processing techniques. A *description technique* is a technique to create a model and represent it as a model denotation (e.g. diagramming technique (Martin *et al.* 1985)). A *processing technique* is a technique to create, transform, translate, analyze, validate and/or verify one or more models. Examples of processing techniques are the normalization technique (Codd 1972) and the affinity analysis technique (Martin 1982). A technique may be composed of procedures and guidelines. A *procedure* is an explicitly specified manner of proceeding in an action or process (cf. Webster 1989). A *guideline* is any advice or guide to reach a goal (cf. Webster 1989). There are many categorizations for guidelines: e.g. minor guidelines, template guidelines, and style guidelines (Anda *et al.* 2001).

Let us illustrate the difference between the descriptive model and the technique with the ER model (Chen 1976). The model is composed of concepts and constructs for abstracting and structuring the modeled context. It does not provide the modeling context with instructions concerning actions, actors, or facilities of modeling. There are special conceptual modeling (CM) techniques (like the one in Benyon 1990) that prescribe how to apply the ER model to produce an ER schema. The concepts in the ER model refer to the modeled context, whereas a CM technique comprises concepts and rules that refer to the modeling context.

---

[127] Lonchamp (1993) distinguishes still another type of models, the proscriptive models (descriptions), which state what is not allowed.

Information system development needs several models and techniques. To facilitate their integrated use, they are composed to form a whole called a method. A method also is a model, prescriptive on one part and descriptive on the other part. Guided by a method, ISD proceeds, step-by-step, in sketching, specifying, elaborating, transforming, validating and verifying IS models on several levels of abstraction. It ends up with the implementation of those organisational and technical changes that have been described by the models and seen beneficial. In Chapter 9 we elaborate further the notion of a method from what has been said above.

To summarize, the notion of a model is always associated with three contexts, the modeling context, the modeled context and the model utilizing context. This basic division becomes visible in the definitions and classifications of the models. This same division is also vital to the quality criteria defined for the models (cf. Krogstie 1995). In the next section we will discuss models that are composed of meta concepts, meta meta concepts, etc. All that has been said about the models in this section holds for those models as well.

7.2 Levels

In Section 3.4 we defined the semiotic ontology distinguishing between three kinds of things: concepts, signs and referents. In Section 3.5 (the intension/extension ontology) the notions of an instance concept and a type concept were defined. Grounding on these ontologies we will here first define the notions of a meta concept and a meta level and from them derive the notions of a meta model and a model level ontology. At the end of this section, we will also present classifications for the meta models based on the classifications of models, contextual domains and perspectives.

Let us first consider the arrangement of three levels in Figure 67. On the lowest level, known as the instance level, the sign 'John' signifies the concept John, which in turn refers to the referent "John". John is assumed to be a concrete instance concept. On the next level, called the type level, the sign 'Person' signifies the concept Person that refers to all the possible persons ("Persons"). Person is a concrete type concept. Its extension contains the referred persons. There is the instanceOf relationship between the concepts John and Person.

Now suppose that in another context Person is considered an instance concept for some type concept, e.g. for Entity type. In this case we can distinguish the third level, on which the sign 'Entity type' (or the rectangular in a graphic notation) signifies Entity type that refers to the referent things, called Entities. Person is an instance of Entity type, which is called a meta concept for

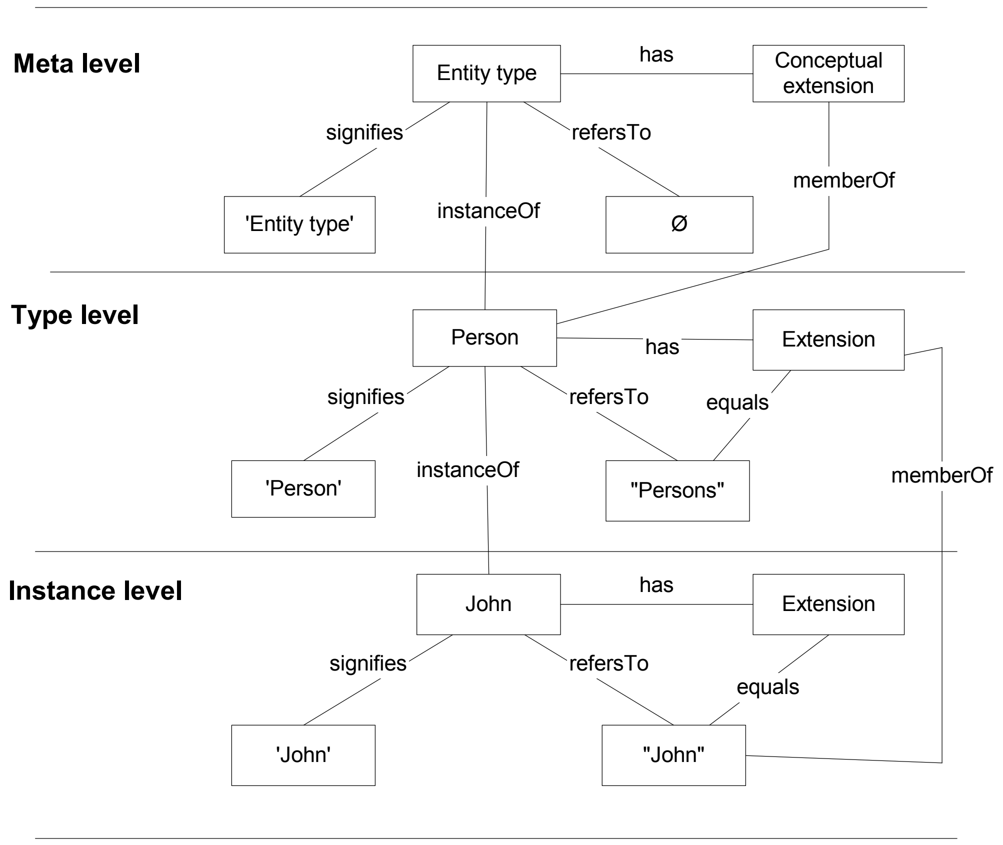


FIGURE 67 Concept levels

John. To put it more precisely, a *meta*[128] *concept* is a concept an instance of which is a type concept for some other instance concepts. The meta concept tells something about the concepts (Bergheim *et al.* 1989, 272). Correspondingly, the level of meta concepts is called the *meta level.*

Let us have a closer look at the relationships between the concepts at the type level and the concepts at the meta level. According to its intensional definition, Entity type defines "a set of entities that have the common attributes" (Elmasri *et al.* 2000, 49). Entity means a thing in reality, having an independent existence. Person is a proper instance of Entity type. Other possible instances of Entity type are Copy, Book, Loan, Reservation etc. Let us look at the concept of extension at each of the levels. At the lowest level, the extension of John is "John". At the type level, the extension of Person is a set of all possible persons, including "John". Entity type is an abstract concept with no concrete referents. Therefore at the meta level there is no (real) extension. However, we can say that Entity type has the conceptual extension (see the definition of the conceptual extension in Section 3.5) that means all those type concepts that apply to the intensional definition of the meta concept Entity

128 Meta is a Greek prefix meaning 'after', 'along with', 'beyond', 'behind' (Webster, 1989).

type[129]. Implied from the above, we can state that there is the memberOf relationship between Person and the conceptual extension of Entity type. This view brings out the manifestation of the vertical shift of the semiotic framework (see the horizontal shift in Section 3.4), meaning that a thing that is seen as a concept in one context can be seen as a referent in another context (cf. Sowa 2000, 194).

In Figure 67 the three concept levels are distinguished. A *concept level* is composed of concepts between which there are no instanceOf relationships. In fact, there may be still more levels. Above the meta level, there can be the meta meta level, above which there may be the meta meta meta level, and so on. At the lowest level, called the root level, the concepts are usually concrete and individual. At the type level, the concepts are generic, often concrete. At the meta level and at the levels higher than that, the concepts are always abstract with no real extension. Concept levels constitute a hierarchy, which we call a system of concept levels.

In the previous section, we defined an instance model and a type model. Now we can define a meta model. A *meta model* is a model that is composed of meta concepts. A meta model is always an intensional model (Falkenberg *et al.* 1998, 58). Like the concepts, also the models with the instanceOf relationships between one another constitute a hierarchy of levels, which we call a *system of model levels.* A *model level* is composed of models that comprise concepts on the same concept level. We distinguish between the following model levels: the instance level, the type level, the meta level, and the meta meta level. Besides the instanceOf relationships, the levels are also related in another way: a model on a higher level describes / prescribes models on the next lower level.

Now we are in the position to define the notion of a model level ontology. The *model level ontology* provides concepts and constructs for conceiving, understanding, structuring, and presenting things in models within a system of model levels. Figure 68 presents the main part of the model level ontology. The ontology also contains those specialized concepts that are included in the classifications in Figure 65 and Figure 66.

To have a more concrete conception about the model levels, let us consider the following example. At the root level, there is a database that is an extensional model of the relevant features of the object system. At the next level, the type model level, there is an ER schema, which describes / prescribes the allowed structure and contents of the data base. The concepts in the data base are instances of the type concepts in an ER schema. At the highest level, the meta model level, there is the ER model, which in turn describes / prescribes the allowed structures and contents of the ER schemas.

129 The term ′conceptual extension′ in exactly this meaning is used in the IS/ISD literature. Hautamäki (1986, 37) defines the notion to be ″the set of concepts to which a given concept is a characteristic. Bergheim *et al.* (1989) come close to our concept with their definition: If ″X″ is an expression whose intension is a meta-concept, then the concept is a member of the extension of ″X″ (ibid p. 293).

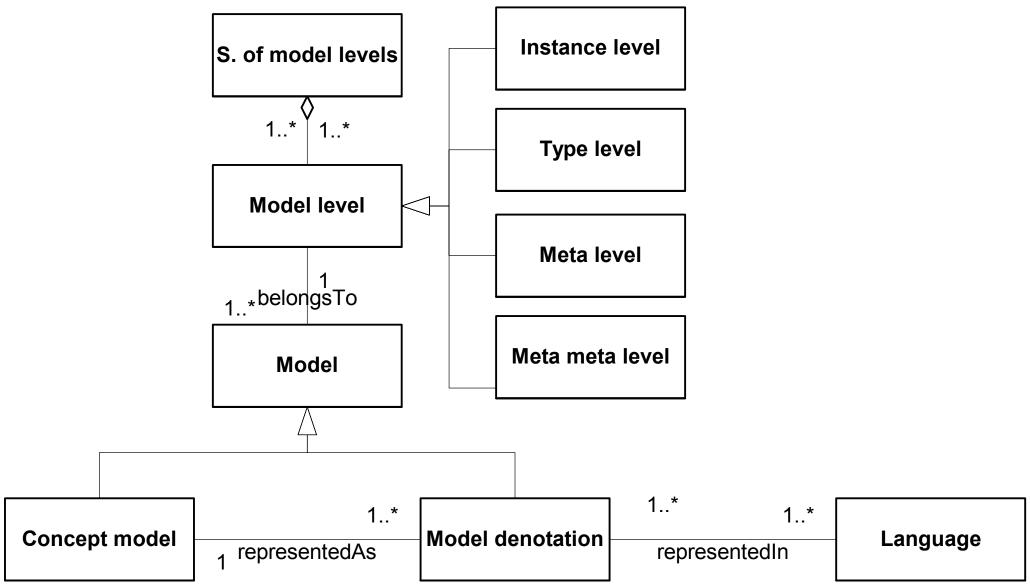


FIGURE 68 Main part of the model level ontology

The selection of the root level determines what the model levels contain. Instead of the data base, we could regard an ER schema to be at the root level. In that case, the meta model would be a model which describes/prescribes the concepts and constructs of the ER model. In the previously considered system of model levels, this model would be called a meta meta model.

In the literature there are three different approaches to define a system of meta levels: (a) the model-based approach, (b) the language-based approach, and (c) the technique-based or the method-based approach. In the *model-based approach* the levels in the system are derived from the instanceOf relationships between the concepts within the models at two adjacent levels in the hierarchy. This kind of approach is applied by e.g. Bergheim *et al.* (1989, 272), Jarke (1992), and OMG (2002). Also we have applied the model-based approach as seen above. In the *language-based approach* the system of meta levels is established for the languages such that a language used to present another language is called a meta language. A meta language, in turn, is represented in a meta meta language, and so on. The meta level hierarchy continues upward until, at some level, a self-descriptive language is used, i.e. a language is reached that is sufficiently expressive to be used to formulate its own rules (Falkenberg *et al.* 1998, 58). The third approach to define the meta levels is called the *technique-based approach* or the method-based approach. The domain in the approach is ISD work or part of it (e.g. IS modeling like in Wijers (1991) and use of a technique in Brinkkemper (1990)). Brinkkemper (1990, 29), for instance, defines a meta model to be a model of a modeling technique. The purpose of this approach is commonly to produce a structural framework for an information base of method knowledge (e.g. Harmsen 1997) or project knowledge (e.g. ConceptBase in Jarke (1992)).

Next, we show how the views of the model-based approach and the language-based approach are related to one another. The key issue in relating the views is the fact that each language used for representing a model has a conceptual foundation consisting of a set of basic concepts and a set of rules, in other words, an abstract syntax (see Section 3.6). This conceptual foundation may also be viewed as a model, called a meta model (Falkenberg *et al.* 1998, 58). To illustrate this we present the levels of models and the levels of languages in the same figure (see Figure 69)[130].

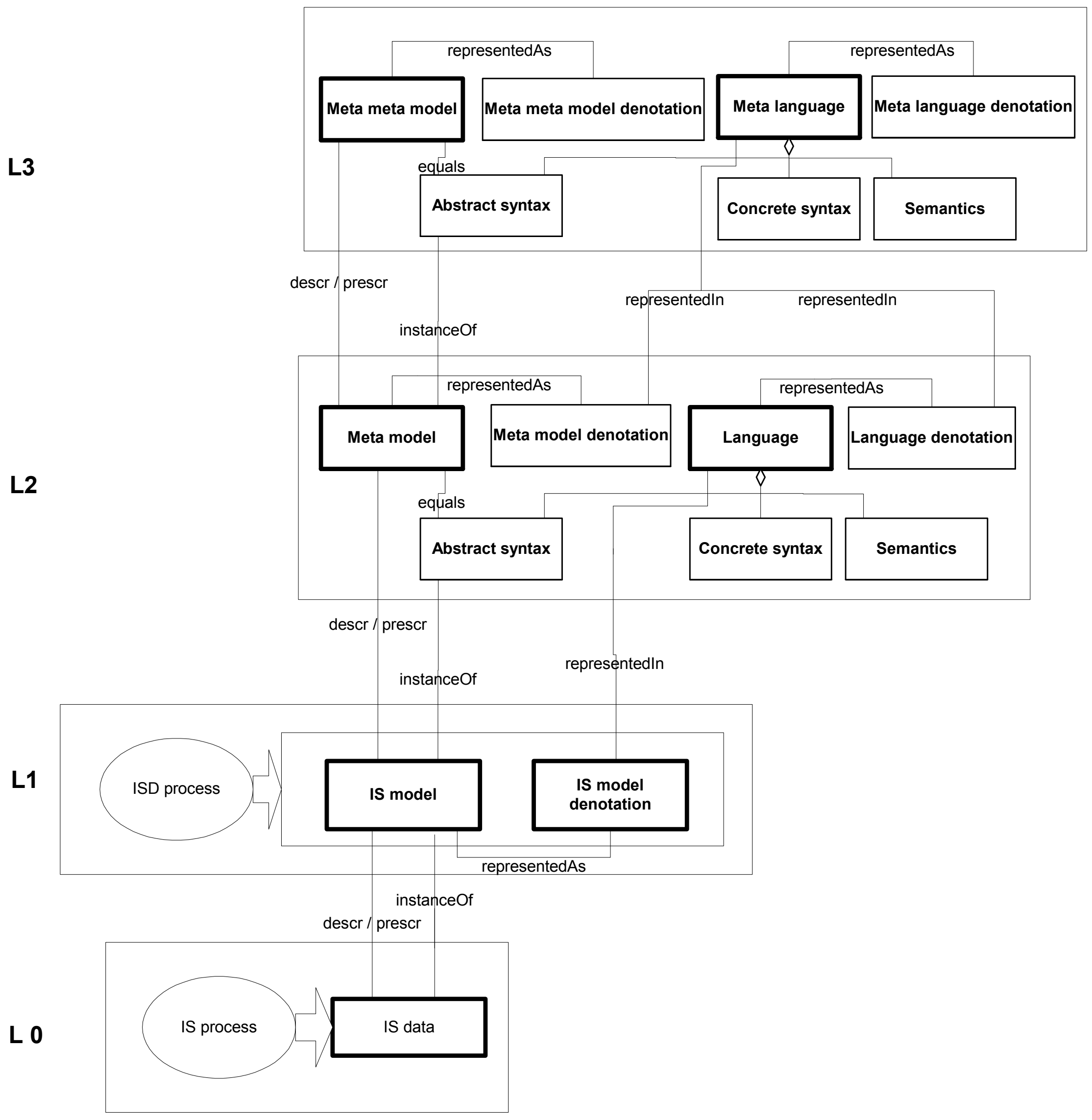


FIGURE 69 Levels of models and languages

Figure 69 contains four levels. Two lowest levels correspond to IS data (L0) and ISD data (L1). ISD data means here IS models and IS model denotations. The next higher level (L2) contains meta models describing / prescribing the IS

[130] To keep the figure simple enough, the multiplicities are not included in it.

models (e.g. an ER schema) produced by ISD, and languages used to represent the IS models as the IS model denotations. The highest level (L3) comprises meta meta models describing / prescribing the meta models, as well as meta languages used to represent the meta models and the languages, both at the next lower level. The figure shows how the hierarchy of model levels is anchored onto the IS data and established via IS models, meta models, and meta meta models. It also shows how the hierarchy of language levels comprises languages and meta languages. The connection between these hierarchies is formed by the fact that a meta model is an abstract syntax of the language (Oei 1995, 113) used to represent a model as the model denotation. The elements in both of the hierarchies are conceptual, and become visible only through their denotations. A meta language denotation is expressed by using the language itself. In the hierarchy of model levels the concepts at the lower levels are instances of the concepts at the next higher levels. On the other hand, we can say that the models at the higher levels describe / prescribe models at the next lower levels. In our study we mainly consider the meta levels from the conceptual viewpoint, meaning that for us the models, the meta models, and the meta meta models are more important than the languages and the meta languages.

As a meta model is a model, most of the characteristics and classifications presented for the models in Section 6.1 apply to the meta models, too. Due to the fact that a meta model is an intensional model, there are, however, some exceptions to this strict "inheritance of the predicates". In the following, we consider the meta models with the predicates of the models.

A meta model is a thing that is used to help or enable the understanding, communication, analysis, design and / or implementation of models. A meta model is an abstract model in the sense that that it is composed of abstract concepts. An action by which a meta model is produced is called *metamodeling.* It takes place on one level of abstraction (by classification) higher than modeling. It comprises several sub-actions: (a) abstracting from existing models, (b) transforming from other meta models, (c) translating from other meta model denotations, (d) revising an existing meta model, and (e) integrating two or more other meta models or parts thereof. Sub-actions of metamodeling will be considered in more detail in Section 10.3.3. A meta model is a formal or semi-formal model reflecting objective (cf. the objective model) or inter-subjective (cf. the inter-subjective model) views. It is also worth of noticing, that a meta model is always needed when interpreting, analyzing, designing, or implementing corresponding (type) models.

The meta models are structural models. Depending on the nature of corresponding (type) models, the meta models can be classified in various ways. Applying the contextual domains and the perspectives, we classify the meta models into two sets of categories, that are the contextual (meta) models and the perspective meta models (see Figure 70). One of the contextual meta models is a meta purpose model, which is composed of concepts and relationships defined within the purpose domain (see Section 4.4.1). An example of the perspective meta models is an infological meta model, which consists of concepts and relationships defined within the IS infological perspective (see Section 6.3.2).

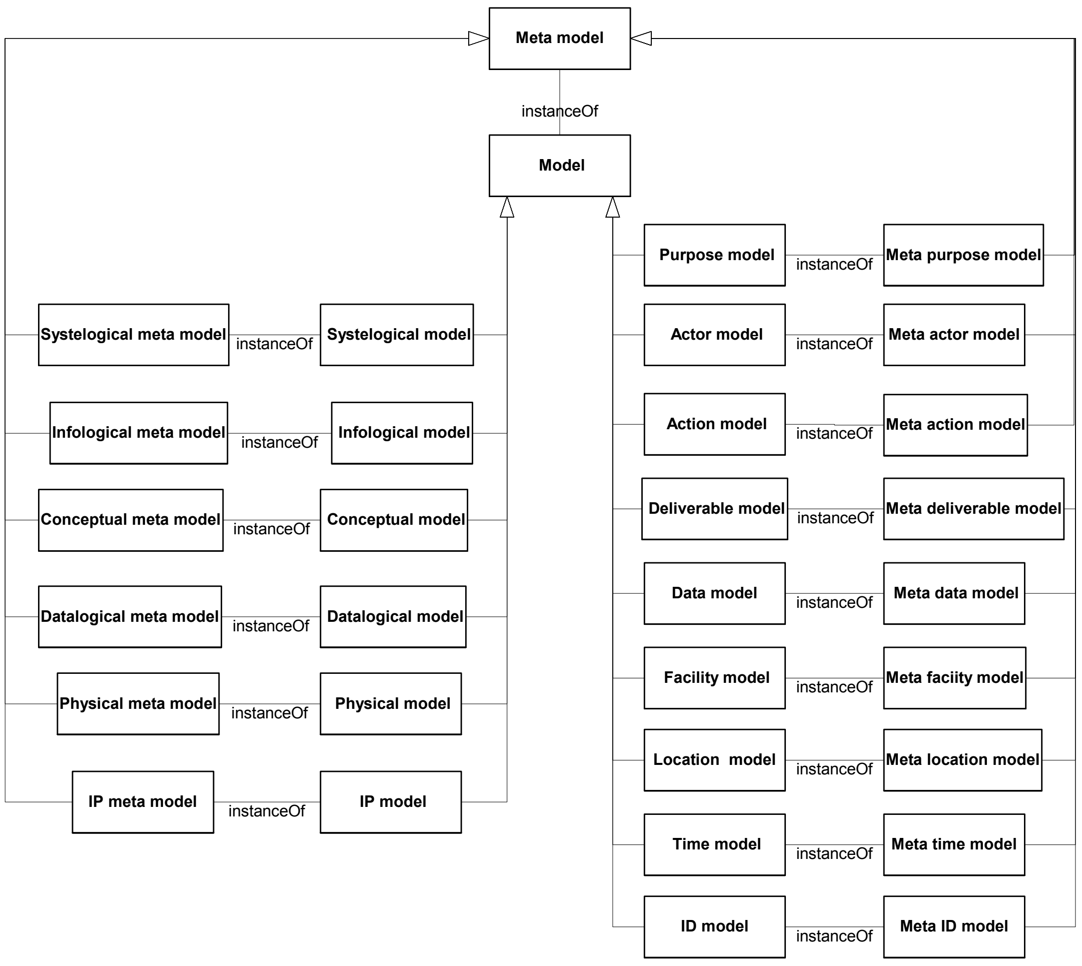


FIGURE 70 Categorizations of models and meta models based on the contextual domains and the perspectives (IP = inter-perspective, ID = inter-domain)

As said in Section 6.1, informational objects in the contexts can be conceived as linguistic objects or through their conceptual contents (see also Section 4.4.4). To bring out the view applied in modeling, we distinguish between deliverable models and data models. A *deliverable model* describes / prescribes the structure and presentation of informational objects (e.g. a relational scheme with data types). A *data model* describes/ prescribes the conceptual contents of informational objects (e.g. an ER schema). Correspondingly, we have a meta deliverable model (e.g. the relational model, Codd 1970, 1979) and a meta data model (e.g. the ER model, Chen 1976).

Second, besides the "pure" models and meta models, which are exclusively based on the concepts of one perspective or one domain, there are also inter-domain (ID) and inter-perspective (IP) models and meta models. Examples of the inter-domain meta models are a meta activity model (e.g.

Brinkkemper 1990) and a meta process model (or a process metamodel in Henderson-Sellers (1999) and Firesmith *et al.* (1999)), which are composed of concepts and relationships, not only within the IS action domain but also within the IS object domain and the IS time domain.

7.3 Comparative Analysis of Systems of Levels

In this section we shortly describe, analyse and compare presentations given in the ISD literature for systems of levels. The systems of levels concern either the concepts or the models. The results of the comparative analysis are summarized into Table 22. In the table for each presentation, the object system (OS) at the root level as well as the names of levels are brought forward.

We have selected twelve well-known presentations for the analysis. Some of them consider the information system (Bergheim *et al.* 1989; Brinkkemper 1990; Falkenberg *et al.* 1998) or CIS data (ISO 1990; OMG 2002) to be the object system. Some others anchor their system of levels on ISD work (Heym *et al.* 1992a; Jarke 1992; Saeki *et al.* 1993; Harmsen 1997; ter Hofstede *et al.* 1997), or on part of it (IS modeling (Wijers 1991)). In the latter ones, the aim is to specify and structure ISD method knowledge into a method base. Gigch (1991) advocates a different approach applying a general view of problem solving in the IS, ISD or any other human action. Next, we first describe the system of levels in each of the presentations and consider which of the approaches (i.e. the model-based, language-based, technique-based or method-based approach) defined in Section 7.2 is applied in them.

Bergheim *et al.* (1989) present a taxonomy of concepts of the science of information systems to distinguish between four meta-levels: ω-level, α-level, β-level, and γ –level. The lowest level, the operational level, concerns the changes of states in the application. The next meta-level, also known as the application level, contains descriptions about a specific application (e.g. a data flow diagram or a Pascal program). The β-level is about how to make instances at the α-level (e.g. a DFD model or the language Pascal itself). The highest level is about how to make instances at the β-level, that is, about the ways to make different formalisms. For each level, a universe, constructs, a theory, an interpretation, valuations, a model, a description, and a method are considered. The discussion in Bergheim *et al.* (1989) about the levels is comprehensive, and considering when it was published, it was in advance of one's time. It is a pure representative of the model-based approach to establishing the system of the levels.

ISO (1990) launched the Information Resource Dictionary Standard (IRDS), which is composed of four levels: application data, IRD level, IRD Definition level, and IRD Definition Schema level. The first level includes data and program execution. The next level stands for a data base schema and

TABLE 22 Comparative analysis of the systems of levels in the ISD literature

Reference	**OS**	**Level 0**	**Level 1**	**Level 2**	**Level 3**
Bergheim *et al.* (1989)	IS	ω-level	α-level	β-level	γ–level
ISO (1990)	CIS data	Application data	IRD level	IRD Definition level	IRD Definition Schema level
Brinkkemper (1990)	IS	System to be modeled		Modeling technique	Meta-modeling technique
Gigch (1991)	Problem solving	Implementation level	Modeling level	Meta level	
Wijers (1991)	IS modeling	Application level	Meta level	Theory level	
Heym *et al.* (1992a)	ISD	Project level	Method level	Methodology level	
Jarke (1992)	ISD	Instance level	Class level	Metaclass level	
Saeki *et al.* (1993)	ISD	Instance level	Object level	Metalevel	
Harmsen (1997)	ISD	IS engineering level	IS engineering method level	Method engineering level	
ter Hofstede *et al.* (1997)	ISD	Operational level	Application level	Method level	
Falkenberg *et al.* (1998)	IS	-	Meta-level 0	Meta-level 1	Meta-level 2
OMG (2002)	CIS data	M0: Data	M1: Model	M2: UML metamodel	M3: MOF

application programs. The IRD Definition level specifies the models and languages by which schemata and programs are described. The IRD Definition Schema level specifies a meta meta model, according to which things at the ISD Definition level are associated and described. The IRDS is an outcome of the model-based approach, although a language as a means of description is recognized. Brinkkemper (1990, 28) distinguishes between three levels: system to be modeled, modeling technique, and meta-modeling technique. In the hierarchy of levels, "the system of concepts of a modeling technique is considered as a concrete system on an abstraction level higher than the application of modeling in the development of an IS" (ibid p. 28). The approach is clearly technique-based. Brinkkemper (1990, 28) provides a figure describing the relationships between the aforementioned concepts and between the notions of modeling notation and meta-modeling notation. The figure is obscure for several reasons: e.g. relationships denoted with unnamed arrows are ambiguous, and actions and outcomes are missing.

Gigch (1991, 17) differentiates knowledge needed to solve a problem into three levels of inquiry. The level of implementation or intervention contains e.g. citizens, clients and practitioners participating in activities involving real world problems. At the modeling level understanding and solving problems requires formulation of models. At the meta level or metamodeling level, people are involved in the design of the methods and approaches to be used at the other levels of inquiry. A meta model is considered a model of the modeling process (ibid p. 255-256). Although the approach in Gigch (1991) is mainly model-based, its scope also comprises modeling processes.

Wijers (1991, 31) divides the knowledge needed in modeling into three levels: application level, meta level, and theory level. At the application level actual ISD processes and products (ISD models) are dealt with. Modeling knowledge concerning the ways of working and of modeling, as well as acquisition of modeling knowledge are included at the meta level. The theory level is concerned with a theory applicable at the meta level. A meta-model is defined to encompass a concept structure (for a way of modeling) and a task structure (for a way of working) as well as constructs interrelating those two. A meta-language is a language used at the meta level (ibid p. 426). Wijers (1991) clearly apply the model-based approach but the scope also contains the process of modeling, which should not be situated at the same meta levels as models. Wijers also considers the integration of the process-oriented and product-oriented views.

Heym *et al.* (1992a) suggest a methodology reference model that is based on three levels of abstraction in which each level applies the notation or specification model from the next higher level. This means that an object type on the higher level is instantiated on the next lower level. The levels are: methodology level, method level and project level. The methodology level describes a methodology reference model, which contains all object types as well as their relationships necessary to describe information systems development methods, e.g. activities, phases, deliverables, or actors. The method level specifies an ISD method by a number of description objects of object types defined at the higher level. The project level describes a particular project to which a certain method is applied, by creating instances of special method description objects from the method level. The scope in Heym *et al.* (1992a) is very broad, covering the whole ISD knowledge. The levels of abstraction are not pure model levels, because at the method level, for instance, part of knowledge concerns ISD process and therefore it is not at the meta level.

Jarke (1992, 57) uses Telos' metaclass hierarchy (Mylopoulos *et al.* 1990) in ConceptBase (DAIDA's metadata management and reasoning system) to document data of projects at three levels: instance level, class level, and metaclass level. The instance level consists of concrete development projects within the environment. The metaclass level describes the development environment at hand. The class level defines the basic structure for development processes. The metaclass hierarchy applies the model-based approach. The object system in the hierarchy is an ISD project, not an IS.

Saeki *et al.* (1993, 150) defines a meta model as a data model or scheme for representing design methods, expressing a concept structure common to various methods. To specify the structural relationships among a meta model, formal representations of design methods (called object models), and actual specification processes, Saeki *et al.* (1993, 151) distinguish between three levels: instance level, object level and meta level. The instance level corresponds to actual products and design activities. The object level stands for the formal representations of a design method. The meta level contains a model for the representations at one level lower as well as the relationships between the representations of design methods. The considerations of the hierarchy of levels remain on a general level. Nevertheless, in our opinion, the meta levels contain knowledge of processes that are located at the wrong level. Saeki *et al.* (1993) clearly apply the method-based approach.

Harmsen (1997) considers the allocation of methodological knowledge onto three levels: method engineering level (ME), IS engineering methods level (ISEM), and IS engineering level. The ME level describes classes of ISEM concepts, that is to say, concepts of any ISD method. The IS engineering methods level describes instances of concepts at the method engineering level. The IS engineering level addresses the actual models, reports, steps, tools etc. used in a ISD project. There are the type/instance relationships between the levels. Meta-modeling is located at the ISEM level, and meta-meta models at the method engineering level. Harmsen (1997) also applies the method-based approach, and, as is typical for the adherents of this approach, he leaves the specification of the hierarchy on too general a level.

ter Hofstede *et al.* (1997, 404) distinguish between three levels of abstraction at which method knowledge can be viewed. The levels are: method level, application level, and operational level. The method level is concerned with knowledge which enables to control the ways how information modeling process may be performed and to define which products may result from those processes. The application level is concerned with information which results from projects for specific organizations and applications. It is an instantiation of the method level. The operational level is an instantiation of the application level and as such it consists of concrete entities, relationships, process traces, etc. Most of what is said about the systems of levels by Saeki *et al.* (1993) and Harmsen (1997) also applies here. The method-based approach of ter Hofstede *et al.* (1997) is applied with too general a view of e.g. the elements at and the relationships between the levels.

In the Frisco Report (Falkenberg *et al.* 1998, 57-58) three meta-levels are distinguished: meta-level 0, meta-level 1, and meta-level 2. At each meta-level, a model and a model denotation are specified. The models are: a base model (meta-level 0), a language (meta-level 1) used to represent the base model, and a meta-language (meta-level 2) used to represent the language. A base model may be a particular model consisting of states and transitions. The corresponding base model denotation is a graphical representation of this model (i.e. a state-transition diagram). The language in this case is like the one in Booch *et al.* (1999). The meta-language can be the MOF (OMG 2002). The representing relationships establish the relations between the meta-levels. Frisco applies a mixed approach considering both the relationships between the languages and the relationships between the models, although the latter relationships become only implicitly specified.

OMG (2002, Kleppe *et al.* (2003, 85-87); Frankel (2003, 105-107)) uses a four-layered architecture for its standards. In the OMG terminology these layers are known as M0, M1, M2, and M3. At the M0 layer there is the running system in which the actual instances exist. The M1 layer contains models of a CIS (e.g. a UML class diagram of a software system). The M2 layer contains meta models (e.g. the UML meta model and the CWM (Common Warehouse Meta model)). At the highest layer, called M3, there are meta meta models (e.g. MOF (Meta Object Factory)) that are used to define meta models. Every meta model is an instance of some meta meta model, and every model must be an instance of some meta model. The object system of the four-layered architecture of OMG is a CIS and especially its data. The architecture has been fully built according to the model-based approach.

As the descriptions above concretely show, there are large varieties in terms and meanings with which models and languages at different levels are specified. The presentations consider the systems of levels from different viewpoints: e.g. from the viewpoint of the science of information systems (Bergheim *et al.* 1989), of metamodeling (Brinkkemper 1990; Wijers 1991; Falkenberg *et al.* 1998), of method engineering (Heym *et al.* 1992a; Saeki *et al.* 1993; Harmsen 1997; ter Hofstede *et al.* 1997), of problem solving modeling (Gigch 1991), of metadata management system (Jarke 1992), and of standardization of development environments (ISO 1990; OMG 2002). This partly explains the varieties perceived in terms and meanings. But there still remain many particularities that should require an in-dept analysis to be revealed, explained and compared. Unfortunately, such an analysis goes beyond this study. As a general finding we can, however, say that presentations that apply the model-based approach seem to be more precise in defining the levels and the relationships between them. In contrast, in those presentations applying the technique-based or method-based approach the levels are defined more generally and the processing layers and the model levels are commonly confused. In the next chapter, we show in a concrete fashion how models at the same model level can situate at different processing layers and a processing layer can contain models from different model levels.

7.4 Models at the Processing Layers

Up till now we have considered the contextual ontologies one by one without paying too much attention to the relationships between them. Because the model level ontology is the last contextual ontology in the order we have introduced them in this study, it is time to examine in more detail how these ontologies are related to one another. The examination will be carried out in two parts. In the first part we use the arrangement, presented at the beginning of each concerned chapter to show its focus (see Figures 44, 52, and 64), to depict the key concepts of the ontologies. In the second part we present a still more detailed view of the ontologies and relationships between them with the figure, which exhibits models about different contextual domains, at different layers, and at different levels.

Figure 71 presents the context ontology, the layer ontology, the perspective ontology and the model level ontology in terms of their key concepts and constructs. The figure also reveals how the ontologies are related to one another. We can distinguish between the following relationships: (a) a model describes / prescribes a context, (b) a model is produced at a processing layer, (c) a model views from a perspective, (d) a context is located on a processing layer, (e) a context is perceived from a perspective, and (f) a perspective is applied at a processing layer. Some of the relationships can be inferred from the others, e.g. if a context is perceived from a certain perspective, then a model of the context is made from that perspective. In addition to those mentioned, there are other relationships that can also be inferred from others. For instance, if a context is perceived from a certain perspective and is located at a certain processing layer, then the processing layer is also perceived from that perspective.

Although the view presented in Figure 71 is quite complicated, it is still on a general level. In Figure 72 we portray all the models, which are distinguished at the processing layers, describing/prescribing any of the domains, and being at any of the model levels. Due to the scarcity of space, the names of the models are written without the term 'models'. In the figure we can see four processing layers (IS, ISD, ME and RW). At each layer there are eight "boxes" standing for the domains (purpose, actor, action, object, facility, location, time, ID = inter-domain) of a context. The models are objects or deliverables resulting from

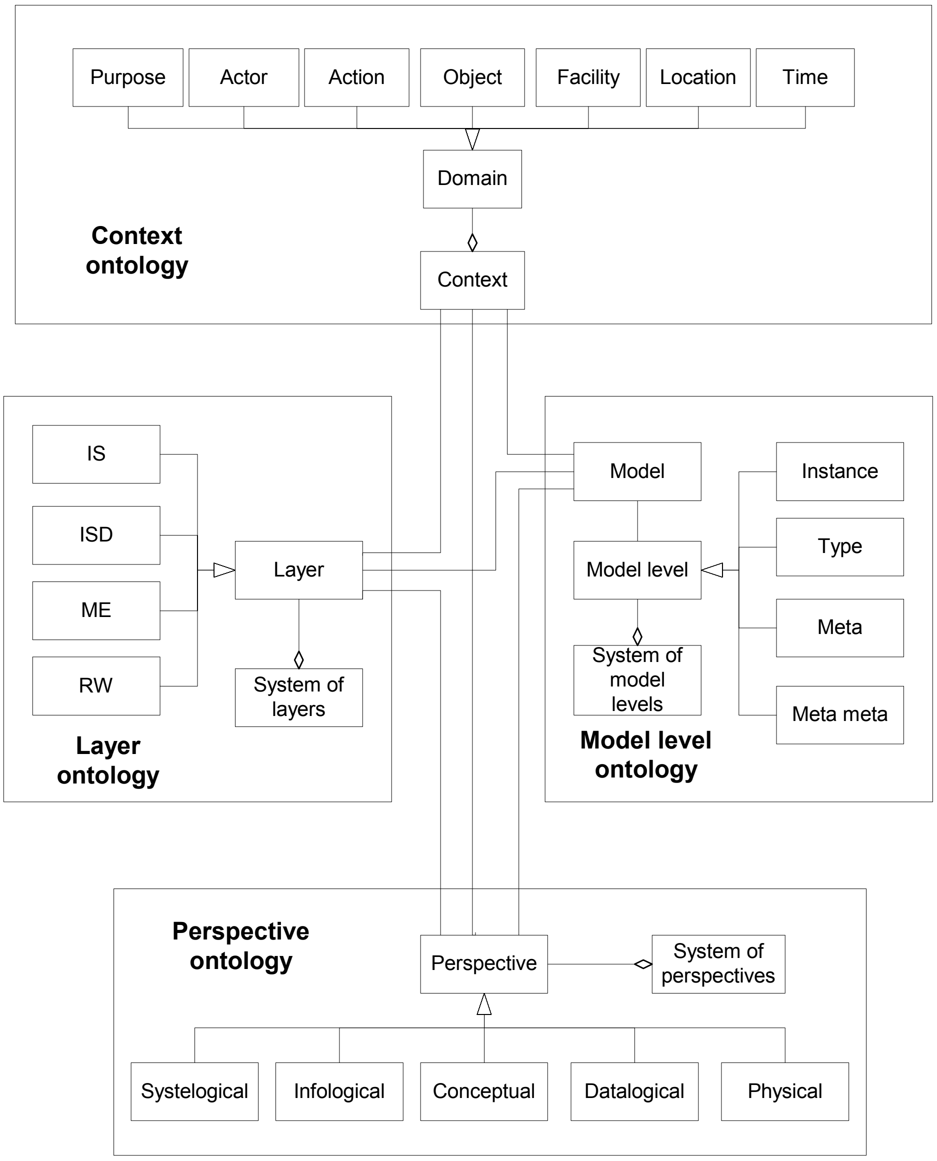


FIGURE 71 Essence of and relationships between the contextual ontologies

actions in the contexts at each processing layer. The arrows denote the instanceOf relationships between the models and the corresponding instance-level phenomena. To illustrate concretely arrays of models, the "boxes" corresponding to the object domains are extended at the ISD, ME and RW layers. Let us next consider the models in more detail.

At the ISD layer, the ISD deliverables comprise nine kinds of models, namely the IS purpose models, the IS actor models, the IS action models, the IS deliverable models, the IS data models, the IS facility models, the IS location models, the IS time models, and the IS ID models. At the ME layer, the ME deliverables also comprise nine kinds of models, in this case the ISD models. The ISD data models mean the IS meta models, which describe / prescribe the set of possible IS models of the concerned types. The IS meta purpose model, for instance, may be the goal graph model (Loucopoulos *et al.* 1998), which is to be used during ISD work to represent IS goals / means hierarchies. An example

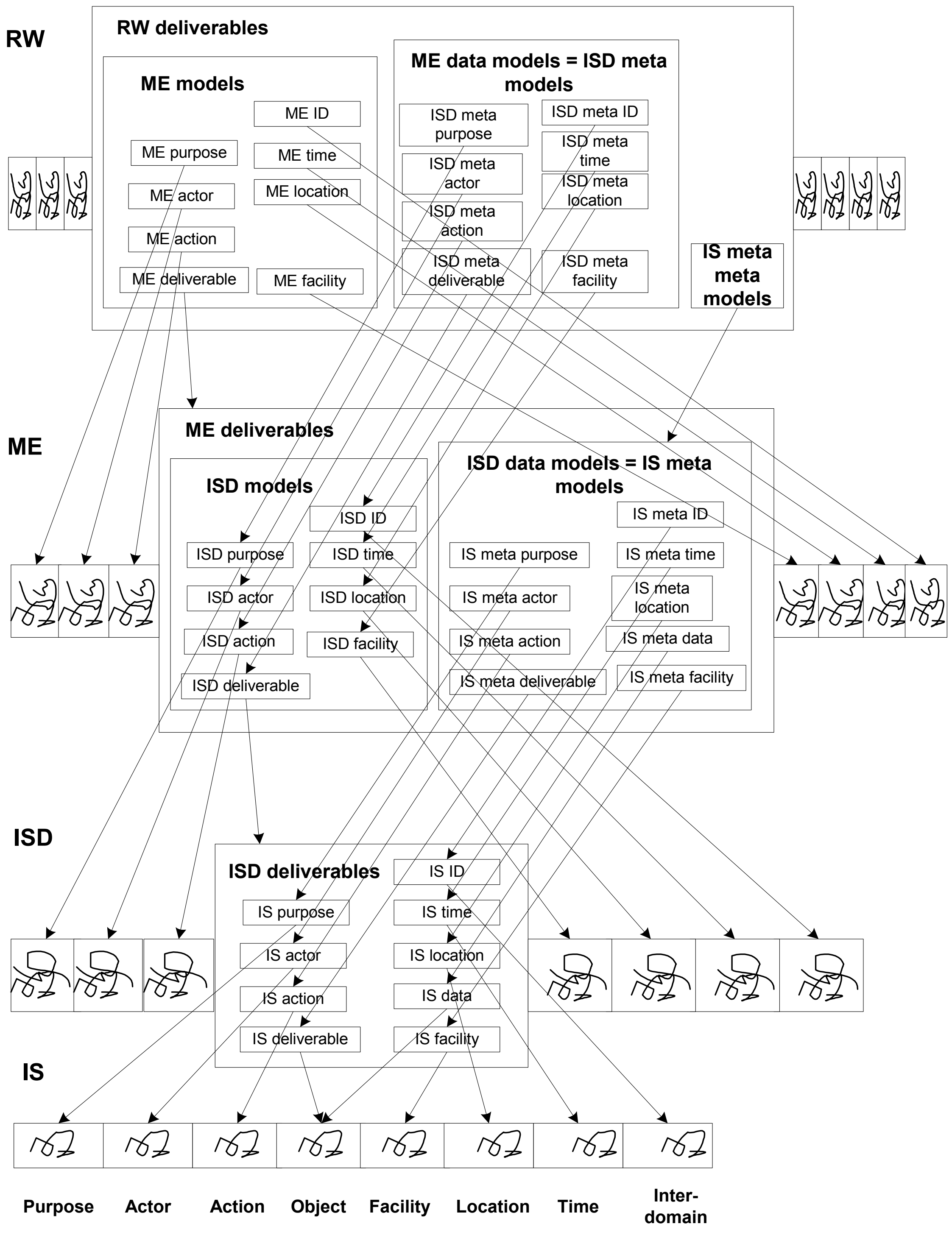


FIGURE 72 An integrated view of the contextual models at three processing layers and on three model levels

of the IS meta data model is the ER model (Chen 1976), which is used to present an ER schema.

The RW deliverables at the RW layer comprise, besides models and meta models, also meta meta models. The models, called the ME models, describe / prescribe the structure and behavior of the current ME context or the desired

ME context, covering all its contextual domains. The meta models, also known as the ISD meta models, specify the concepts and constructs of the ISD models at the lower model level. The meta meta models are the IS meta meta models, of which the IS meta models are instances. An IS meta meta model contains the abstract syntax of those meta languages that are used to represent the IS meta models (cf. Section 7.2). In this study we use a sub-set of UML as the meta meta model.

Three remarks should still be made on the figure. Working at some processing layer always involves models at three model levels, namely the resulting model, the concerned meta model and its meta model. For instance, in designing an object database, a class diagram is produced according to the object class model. To fully understand and deploy the concepts and notation of the object class model, it is necessary to know the semi-formal language in which it is presented, in other words the meta meta model. In addition, it is necessary to some extent to understand phenomena at the next lower level to be able to present abstractions of them in the form of a class diagram.

Second, Figure 72 concretely manifests how analogous the processing layers in terms of models and meta models really are. The same kinds of models can be recognized at the ISD, ME and RW layers, and also the meta models at the ME and RW layers can be quite equal. Of course there are some differences in details and emphasis of certain contextual phenomena at the layers, but there is no need, in principle, to construct and deploy a large variety of meta models and meta meta models for the layers.

Third, we have not included the "perspective dimension" in the figure. Having done this would have made the figure perhaps too complicated. At each processing layer, it would have been possible to show that there are six kinds of models, meta models and meta meta models, depending on the applied perspective.

The integrated view of the models, processing layers and model levels in Figure 72 serves as a useful foundation to position and relate the issues that will be discussed in the next chapters. In Chapter 8 we aim to define the ISD ontology, which addresses the contextual domains and the perspectives at the ISD layer. The purpose of Chapter 9 is to define the ISD method ontology. We have already generally defined a method to be a model that describes / prescribes an ISD context. Now we can see that it decomposes to several contextual ISD models, including the IS meta models. In that chapter we will elaborate this view to specify the contents and structure of an ISD method. In Chapter 10 we will define the ME ontology and the ME method ontology that are positioned at the ME layer and the RW layer, respectively. In both of the chapters the approaches, views, models, concepts and constructs will be further refined from what we have presented here.

7.5 Summary and Discussions

In this chapter we defined the model level ontology. The ontology provides concepts and constructs to conceive, understand, structure and present phenomena in reality in terms of models within a system of model levels. Generally, a model is a thing that is used to help or enable the understanding, communication, analysis, design, and/or implementation of some other thing to which the model refers. The notion of a model can be specialized according to the aspects of the modeling context, the modeled context, and the model utilizing context. To facilitate communication about a model, it is presented in some language. A language is composed of a concrete syntax, an abstract syntax, and semantics. Essential to the model level ontology is the recognition of hierarchical meta levels of concepts and models. A meta model is a type model, which describes / prescribes another type model that is at the next lower model level. In the hierarchical system of model levels, there are instanceOf relationships between the models at one level and the models at the next lower level.

Second, we described and compared systems of levels presented in the ISD literature. Concluding from the analysis we can say that there exist large varieties in terms and meanings with which the levels are called and specified. The presentations have been established for several purposes, e.g. for specifying the conceptual contents of the science of information systems, meta modeling, method engineering, metadata management systems, and standardization of development environments. The root level in these presentations varies from the IS layer to ISD layers (CIS data, IS modeling, ISD). There are also different approaches to defining the systems of levels. In particular in the method-based approaches and the technique-based approaches there are confusions in distinguishing the model levels and the processing layers. This confusion can be avoided with the use of a unified conceptual foundation, such as OntoFrame.

Third, we provided an integrated view of four contextual ontologies, including the context ontology, the layer ontology, the perspective ontology, the model level ontology. The ontologies were integrated via the focal concepts of context, layer, perspective and model. We also exhibited and related a large variety of models, which concern four processing layers, seven contextual domains and three model levels.

8 ISD ONTOLOGY

Information system development (ISD) means the accomplishment of organizational and technical changes in an IS context. It aims at improving an IS, that is to say, to make it more effective, efficient, reliable, easy-to-work, user-friendly, etc. Small-scale changes and improvements in an IS are carried out with daily work. But accumulation of problems, becoming conscious of new technological potentials, or decisions on new business strategies and policies may trigger a special effort to design and implement more profound changes in an IS. This is a kind of ISD we consider it here.

The purpose of this chapter is to present the ISD ontology that provides fundamental concepts and constructs for conceiving, understanding, structuring, and representing essential phenomena in ISD. The ontology is specialized from the underlying ontologies, in particular from the context ontology and the perspective ontology (see Figure 73). The context ontology provides a basis for a theory-based classification of the concepts of ISD into seven contextual domains. With the perspective ontology it is possible to manage the complexity of the target system by viewing its phenomena from well-defined perspectives. Applying the perspectives to ISD helps us understand how conceptions about the ISD can develop step by step in method engineering. Resulting from the underlying ontologies the ISD ontology has been constructed from two main parts: ISD domains and ISD perspectives.

The chapter is organized as follows. First, we discuss and classify ISD paradigms and ISD approaches that affect views of and conceptions about what ISD really is. Second, we give a comprehensive definition for the notion of ISD. Third, we present the first main part of the ISD ontology. It is composed of meta models and concept definitions within four ISD domains. The domains considered are: the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain. Also an overview of the inter-domain relationships is given. Fourth, we present the second main part of the ISD ontology, which is composed of four ISD perspectives. The perspectives are: the ISD systelogical perspective, the ISD infological perspective, the ISD conceptual perspective, and the ISD datalogical perspective. Also inter-perspective

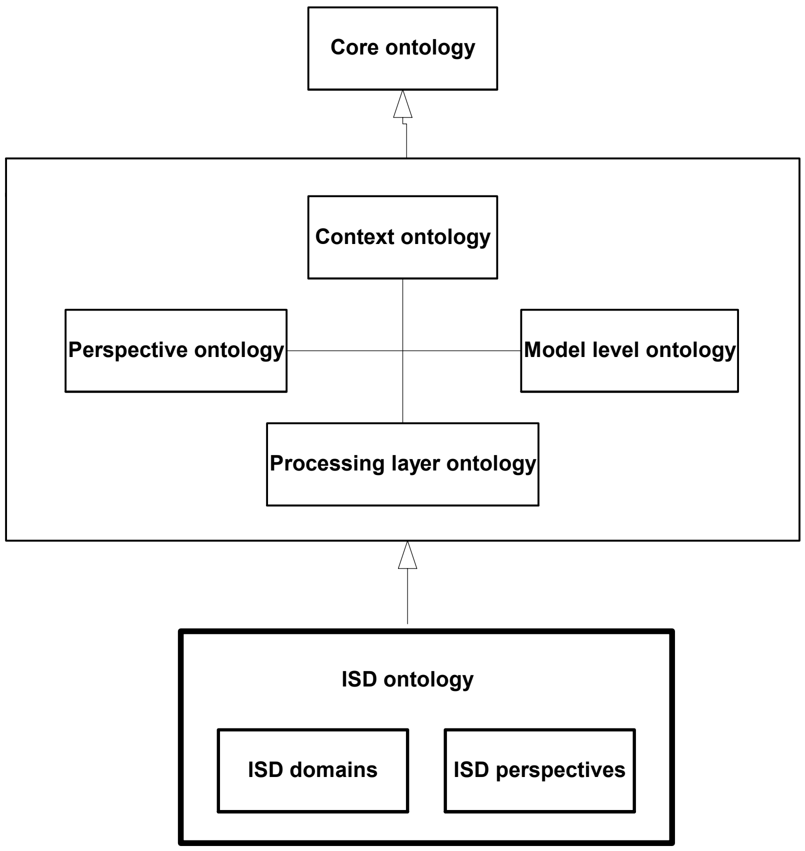


FIGURE 73 Basis and structure of the ISD ontology

relationships are discussed. Sixth, we make a comparative analysis of artifacts (i.e. frameworks, meta models and the like) presented in the literature. The purpose of the analysis is to obtain an overview of the artifacts, to find out how they differ from one another, and compare them with the ISD ontology in terms of comprehensiveness and focus. The chapter ends with a summary and conclusions.

8.1 ISD Paradigms and ISD Approaches

In the ISD literature, there are highly divergent conceptions about the nature, purpose, structure, and behavior of ISD. Conceptions can be, on a general level, categorized, analyzed and compared through ISD paradigms and ISD approaches underlying them. Basic assumptions, views and principles of ISD paradigms and ISD approaches formulate our views of information systems development and thus affect through which concepts and constructs we conceive, understand, structure and represent phenomena in ISD. To get a firm foothold for defining the notion of ISD and later for establishing the ISD ontology, we briefly discuss and classify the ISD paradigms and the ISD approaches in the following sub-sections.

8.1.1 ISD Paradigms

The notion of paradigm has been a controversial concept ever since Kuhn (1970) introduced it (Iivari *et al.* 1998a). Kuhn defined a paradigm to mean "universally recognized scientific achievements that for a time provide model problems and solutions to a community of practitioners". Burrell and Morgan (1979) state that paradigms are "meta-theoretical assumptions about the nature of the subject of study". In the ISD field, a paradigm is defined as e.g. "a specific way of thinking about problems, encompassing a set of achievements that are acknowledged as the foundations of further practice" (Avison *et al.* 1995a, 447). We share the conception presented by Hirschheim *et al.* (1992b)[131] according to which a *paradigm* means "the most fundamental set of assumptions adopted by a professional community which allow it to share similar perceptions and engage in commonly shared practices" (ibid p. 305).

Burrell and Morgan (1979) distinguish between four types of assumptions: ontological assumptions (assumptions about the world), epistemological assumptions (i.e. assumptions about the knowledge), methodological assumptions (i.e. assumptions about the appropriate mechanisms for acquiring knowledge), and human nature issues. On the bases of the types of assumptions, Burrell and Morgan (1979) establish two dimensions: the order-conflict dimension and the subjectivist-objectivist dimension. They also identify four paradigms of sociology and organizational research: functionalism, interpretivism, radical structuralism, and radical humanism. Hirschheim *et al.* (1989) extend the notion of paradigm further and show that the four paradigms of organizational research also exist in the literature of ISD. They refer to the paradigms with the following terms: functionalism, social relativism (interpretivism), radical structuralism, and neohumanism (radical humanism).

Iivari (1991, 255) refines the paradigmatic framework and introduces ethics of research as the fourth constituent of the framework. Ethics concerns the responsibility of a scientist for the consequences of his research and its results. Iivari (1991) and later Iivari *et al.* (1998a) applied the framework to analyze the schools of IS development. Hirschheim *et al.* (1992a) and Hirschheim *et al.* (1995) used the framework to make a paradigmatic analysis of ISD approaches.

Besides those mentioned above, there are also other authors who have contributed to the discussion about the paradigmatic categories in the IS/ISD fields (e.g. Floyd 1987; Nurminen 1988; Orlikowski *et al.* 1991; Dahlbom *et al.* 1993; Stamper 1992b; Jayaratna 1994). It goes beyond our aims to consider them here more closely. We merely state that we apply the paradigmatic framework of Hirschheim *et al.* (1989) and Hirschheim *et al.* (1992a), because it is firmly established on the philosophical traditions and widely applied in the ISD literature. In the following we present short characterizations of the four paradigms with words of Hirschheim *et al.* (1989, 1203-1210) and Hirschheim *et*

131 This definition is also adopted in Hirschheim *et al.* (1989, 1201), Hirschheim *et al.* (1992a, 305) and Iivari *et al.* (2001).

al. (1992a, 308-309). For each paradigm the nature of the IS and the roles of, and relationships between, various stakeholders of the IS are outlined.

The *functionalist paradigm* is concerned with providing explanations of the status quo, social order, social integration, consensus, need satisfaction and rational choice. ISD goals are dictated by a "technological imperative". ISD work proceeds by applying formal concepts through planned intervention with rationalistic tools and methods. Managers are responsible for providing the systems goals. The system developer is the expert who takes the goals and turns them into a constructed product. Users operate or interact with a system to achieve organizational goals.

The *social relativist paradigm* seeks explanation within the realm of individual consciousness and subjectivity. Any goals or values that are consistent with social acceptance are legitimate. ISD work proceeds by improving subjective understanding and cultural sensitivity through adapting to internal forces of evolutionary social change. Users are the organizational agents who interpret and make sense of their surroundings. The systems developer is the change agent who helps users make sense of the new system and its environment.

The *radical structuralist paradigm* has a view of society and organizations that emphasizes the need to overthrow or transcend the limitations placed on existing social and organizational arrangements. All goals other than those that further the class interests of the workers are considered illegitimate and reactionary. ISD work proceeds by raising ideological conscience and consciousness through organized political action and adaptation of tools and methods to different class interests. The two antagonistic classes, the owners of the productive resources and labor, are engaged in a classic struggle. The owners become the beneficiaries of IS's while labor becomes the victim of system rationalization. The management acts as the agent of the owners. The systems developer chooses between being an agent for the management or member of the labor force.

The *neohumanism paradigm* seeks radical change, emancipation and potentiality and stresses the role that different social and organizational forces play in understanding change. Only goals that survive from maximal criticism and thus are shown to serve generalizeable human interests are legitimate. ISD work proceeds by improving human understanding and the rationality of human action through emancipation of suppressed interests and liberation from unwarranted natural and social constraints. The stakeholders, comprising customers, management, labor and owners of the productive resources, exist within an intertwined set of social relationships and interactions. They take part in communicative action. The systems developer acts as a social therapist in an attempt to draw together the various stakeholders.

8.1.2 ISD Approaches

In the ISD literature hundreds of ISD approaches have been suggested with varying contents and motives. One reason for the flavor of the term 'approach' is its vagueness: it can be used to mean almost anything, at any level of detail. For instance, at the initial stage of a research effort when no concrete method, model or technique can yet be presented, ideas can be packaged and "marketed" with a named approach. Sometimes, an approach is not even defined, but used as a kind of label attached to obscure ideas.

Another reason for the favor of the term is that it can be used to emphasize and highlight specific features (e.g. goal and scenario based approach (Liu *et al.* 2002), context-based approach (Kashyap *et al.* 1996), process-oriented approach (Mylopoulos *et al.* 1992)), or certain underlying theory or discipline (e.g. speech-act-based approach (Auramäki *et al.* 1988), contingency approach (e.g. Zhu 2002), activity theory approach (e.g. Boer *et al.* 2002), semiotic approach (e.g. Calway 1995), genre-based approach (e.g. Päivärinta 2002)). Further, with the use of approach one can inform that one's suggestion is related to a certain model (e.g. UML-based approach (Briand *et al.* 2002), ER-approach (Batini *et al.* 1992), conceptual graph approach (Moulin *et al.* 1992)) or a certain technique (e.g. conceptual modeling approach (Motik *et al.* 2002), meta-modeling approach (Chiu *et al.* 1999), meta model transformation approach (Oei 1995)). Finally, there are approaches that reflect specific ways of carrying out some actions (e.g. top-down approach (Peristeras *et al.* 2000), unified approach (Potter *et al.* 1988), and formal approach (Hong *et al.* 1993)).

To manage fuzziness that troubles the understanding and use of the notion of ISD approach, we first present a general definition of an ISD approach and then subdivide the ISD approaches into three categories. An *ISD approach* is defined to mean a generic way of conceiving certain aspects of ISD, or a generic way of working in ISD. The first category in our classification is composed of those ISD approaches that are some kinds of "schools of thought". The second category comprises approaches that take a specific view on ISD. The third category is composed of ISD approaches that are suggested to emphasize certain features related to specific ISD domains. In the following we first discuss the ISD approaches as the categories A, B and C. After that we briefly consider relationships between the approaches of these categories.

ISD Approaches in Category A

The *category A* contains ISD approaches that are kinds of "schools of thought". Iivari (1991) states that schools of thought have identifiable founders and scientific community to enable their institutionalization. Accordingly, an ISD approach here means "a set of goals, guiding principles, fundamental concepts, and principles for the ISD process that drive interpretations and actions in the ISD" (Iivari *et al.* 1998a, 166). The goal specifies "the general purpose of the approach". Guiding principles form "the common 'philosophy' of the approach". The fundamental concepts define "the nature of an IS and the focus and unit of analysis and design in ISD". The principles of the ISD process express essential aspects of the ISD process in the approach (Iivari *et al.* 1998a, 166).

According to Iivari *et al.* (2001) the group A comprises the following approaches (a reference to a representative of each approach is given in parenthesis): structured approach (e.g. Yourdon 1989), information modeling approach (e.g. Martin 1989), decision support systems approach (e.g. Keen *et al.* 1978), socio-technical approach (e.g. Mumford 1983), object-oriented approach (e.g. Henderson-Sellers *et al.* 1995), infological approach (e.g. Lundeberg *et al.* 1981), interactionist approach (e.g. Kling 1987), speech act –based approach (e.g. Auramäki *et al.* 1988), Soft-Systems Methodology approach (e.g. Checkland 1981), trade unionist approach (e.g. Bjerknes *et al.* 1987), and professional work practice approach (e.g. Andersen *et al.* 1990).

There are also other classifications of ISD approaches that can be seen, at least partly, to belong to this category, though they are not referred to as schools of thought by the authors. Wood-Harper and Fitzgerald (1982) distinguish between the general systems theory, the human activity, the systems approach, the participative approach, the traditional approach, the data analysis approach, and the structured approach. Benyon and Skidmore (1987) recognize the software systems approach, the structured systems analysis and design, the traditional approach, the data-centered approach, and the participative approach. Also some of the approaches recognized in the taxonomies of Lyytinen (1986) and Hirschheim *et al.* (1995) belong to this category.

ISD Approaches in Category B

The *category B* contains ISD approaches that adopt a specific view of ISD as a context. An adopted view determines concepts and constructs through which ISD is perceived and structured. We distinguish between six approaches: the transformation approach, the decision making approach, the problem solving approach, the learning approach, the political approach, and the knowledge work approach. Next, we characterize and give examples of these view-based approaches.

According to the *transformation approach,* ISD is seen as sequential steps of transforming ISD deliverables on one level of abstraction into ISD deliverables on the next lower level of abstraction (e.g. Lundeberg *et al.* 1981; Lehman 1984; Fickas 1985; Turski *et al.* 1987; Wand 1988a; Moynihan 1993; Tracz *et al.* 1993; Jacobson *et al.* 1999, 24). Abstraction is commonly performed according to the principles of predicate abstraction based on the realization criterion (cf. Section 3.8.3). This means that during the ISD process requirements specifications are transformed through analysis and design deliverables into implementational deliverables.

According to the *decision making approach,* ISD is seen as a decision making process in which knowledge is acquired, options are specified, and the "best" options are selected (e.g. Iivari 1983; Iivari *et al.* 1987; Jarke *et al.* 1990; Wild *et al.* 1991; Jarke *et al.* 1992; Grosz *et al.* 1997). An example of realization of the decision making approach is suggested in the NATURE approach to requirements engineering (Jarke *et al.* 1993; NATURE Team 1996), according to which a requirements engineer is in a situation that he/she considers with some specific intention. His/her decision depends on the context he/she is placed in. The NATURE approach, slightly evolved, has been later applied in e.g. Rolland *et al.* (1996), Pohl *et al.* (1997), Pohl *et al.* (1999) and Rolland *et al.* (2000). The ISD process involves also claims on and arguments for decisions (Conklin *et al.* 1988; Ramesh *et al.* 1994).

According to the *problem solving approach,* ISD is seen as a problem solving process in which problems at several levels of details are identified and solved (Bodart *et al.* 1983; Dasgupta 1989; Sol 1992; Blum 1994; Jayaratna 1994). Problem solving can be viewed as utilizing available means to reach desired ends while satisfying "laws" existing in the environment (Hevner *et al.* 2004). Bodart *et al.* (1983), for instance, suggest an analysis framework of the ISD process, which is based on distinguishing between three classes of problems: abstraction problems, decision problems, and control problems. Sol (1992) suggests that ISD problems can be approached through viewing them from a certain point of view. Based on the views he subdivides the problems into systelogical, infological, datalogical, and technological problems.

According to the *learning approach,* ISD is seen as a learning process by which knowledge on application domain, technology and ISD work is acquired, elaborated and disseminated (e.g. Iivari 1982; Ramesh *et al.* 1994, 296). ISD is enabling and is enabled by personal and organizational learning. Systems development aims at achieving growing cognitive and interpersonal skills coupled with better self-awareness among all the participants. Organizations can view themselves as learning organizations, able to learn from their experience and to effect changes in their own actions (Lyytinen *et al.* 1999). Some approaches, such as prototyping (Floyd 1984; Bai 1998), facilitate learning better than others.

According to the *political approach,* ISD is seen as a cooperative process composed of negotiations, bargaining, power and social interactions (Newman *et al.* 1990). Interactions are based on political machinations and result in manifestations of power (Avison 1996). Markus (1983) considers the implementation of IS and its impact on power shift in the organization in the light of the political variant of the interaction theory. Keen (1981) analyzes the political games in ISD. Chang *et al.* (2002) identify 41 kinds of political games in the analyzed ISD projects based on thematic interviews. They also discuss the relationships between political games and stages of ISD as well as organizational factors affecting political games.

According to the last approach in this category, ISD is viewed as *knowledge work* (Iivari *et al.* 2001, 205). For knowledge work there are certain characteristics (Iivari *et al.* 1999): (a) there must be a clearly identified body of knowledge, (b) work must be concerned with creating or manipulating representations rather than the physical objects of work, (c) it must require a deep understanding of the objects of work, and (d) it must result in products that entail knowledge as their essential ingredient. Iivari *et al.* (2001, 206) distinguish between three components that make up the body of knowledge for

ISD: knowledge of information technology, application domain knowledge, and system development (process) knowledge.

ISD Approaches in Category C

The *category C* contains ISD approaches that have particular views of some specific contextual domain(s) of ISD. They are more skeletal than the ones in the other categories. A variety of the ISD approaches in this category is so large that we only give some examples of the approaches. References primarily point to pioneers in establishing and applying the approaches.

First, there are ISD approaches that differ from one another on how they perceive and emphasize the features of the IS to be developed. We can recognize four specific approaches (cf. Bracchi *et al.* 1984; Barbic *et al.* 1985, 150; Vessey *et al.* 1994): the IS data-oriented approach, the IS process-oriented approach, the IS user-oriented approach, and the object-oriented approach. The *IS data-oriented approach* regards data as the fundamental part of the IS. It is claimed that identifying and classifying the conceptual entities in the OS_{IS}, or the set of data elements in the IS, one can establish the core nature of the IS. It is assumed that the fundamental structure of data remains, to a high degree, the same although the IS may face many kinds of changes in the environment, or at least it will change much less likely than the actions applied to it (cf. Wood-Harper *et al.* 1982, 13). All other constituents of the IS are subordinated to the data. For example, actions are seen as a series of operations on data (Bracchi *et al.* 1984, 163).

The *IS process-oriented approach* considers information processing actions or processes to be the most essential parts of the IS. Processes are viewed as concrete and easy to understand. IS analysis and design are carried out in a top-down, structured, and modular manner. The other parts of the IS are perceived through their relationships with actions. The IS actions are modeled with data flow diagrams, action decomposition models, Petri nets, etc. (cf. Gane *et al.* 1979; Yourdon 1989; Zisman 1977). The *IS user-oriented approach*, or the use-centered approach, puts the major emphasis on human beings, their needs, views and interactions in the IS and in the US. This is seen as a prerequisite for the proper recognition and establishment of the structure, behavior and evolution of the IS. A conception about the 'objective' data, shared and agreed by the stakeholders (i.e. consensus or standard data), collapses. The meaning of data is firmly attached into a subject, and the existence and justification of different views are acknowledged. Roles and positions are built up from the premises of meaningful jobs, which provide more challenges and less routine, potential to self-control, and a sense of personal achievements (e.g. Mumford 1981, 1983).

The object-oriented approach, with roots in SIMULA (Dahl *et al.* 1968; Nygaard *et al.* 1978) and in abstract data types (Parnas 1972; Morris 1973; Liskov *et al.* 1974; Guttag 1977), also belongs to this category of ISD approaches. Instead of focusing on phenomena in one contextual domain, the *object-oriented approach* considers data and processes as encapsulated compositions, known as objects. In ISD data and processes are designed and elaborated in parallel (Booch 1991; Henderson-Sellers 1992; Jacobson *et al.* 1992). In its latest variant, called the agent-oriented approach, objects are seen as agents with autonomy, social ability, reactivity, and pro-activeness (Wooldridge *et al.* 1995).

Second, there are ISD approaches that differ from one another in how they structure the ISD process. We can distinguish between the life cycle approach, the prototyping approach, the incremental approach and the evolutionary approach. The approaches differ from each other in three essential aspects (cf. Vlasblom *et al.* 1995, 598): (1) whether ISD actions are carried out iteratively or in linear fashion, (2) whether the delivery is monolithic or incremental, and (3) the degree to which the systems functionality is defined beforehand. In the *life cycle approach*, the ISD work is decomposed into discrete phases to be accomplished in an order that is comparable to sequential waterfalls. Each phase should be satisfactorily completed before the next one begins (Royce 1970). This implies, for instance, that user requirements must be frozen in the early phase. In the *prototyping approach* requirements are engineered in parallel with their implementation (Gomaa *et al.* 1981). The purpose is to increase, through prototypes, the understanding of those issues on which there exists some uncertainty, and thus to decrease risks related to the ISD process or its outcome (Floyd 1984). The target of prototyping can be information requirements, user-interface, technical architecture, or the like (Bai 1998). There are several variants of the prototyping approach (Iivari 1982; Budde *et al.* 1984; Boehm 1988; Iivari 1990b).

The *incremental approach* means the process of constructing a partial implementation of the total system and slowly adding increased functionality or performance (Graham 1989; Vonk 1990). The approach reduces the costs incurred before an initial capability is achieved. Also of the incremental approach there are several variants (Iivari 1982; Graham 1989). The *evolutionary approach* means that the information system is an incremental outgrowth of evolution and learning and it continues to evolve over time owing to new learning experiences (Lucas 1978; Lyytinen 1986). It is assumed that requirements will constantly change and, therefore, the iterative process never ends. The initial version of the system as a prototype is delivered to the intended users and it continues to be improved until it becomes the system.

Synthesis

The ISD approaches are often defined in an insufficient manner in the literature, which makes it difficult to fully understand them and to construct a clear-cut categorization for them. Our sub-division of the ISD approaches into three categories is an attempt to bring some structure among the approaches. There are many kinds of relationships between the approaches in the same category and in the different categories. Contrary to Benyon *et al.* (1987) who use the 'map' metaphore (cf. national maps vs. major routes vs. street plans) in sub-dividing the ISD approaches into hierarchical categories, we do not claim that the approaches within the three categories constitute a strictly hierarchical structure. However, for most of the approaches it holds that an approach in the "upper" category is realized by applying views and principles of some approach(es) in the "lower" category (cf. Iivari *et al.* 2001). In the following we give some examples of this kind of relationship.

First, the information modeling approach (e.g. Martin 1989) and the structured approach (e.g. Yourdon 1989) belonging to the category A, clearly apply views and principles of the data-oriented approach and the process-oriented approach, respectively. To the interactionist approach (e.g. Kling 1987) and the trade-unionist approach (e.g. Bjerknes *et al.* 1987), in turn, the principles and ways of working contained in the political approach are particularly important.

Second, the transformational approach in the category B mainly applies views of the data-oriented approach and / or the process-oriented approach. In the former case, information requirements are first transformed into a conceptual schema, then changed into a relational schema and further implemented into physical files. In the latter case, based on information requirements a context diagram is produced, from which data flow diagrams are derived and further decomposed into more detailed process descriptions. The prototyping approach and the evolutionary approach, in turn, are the most essential means of implementing the learning approach to ISD.

8.2 Definition of ISD and ISD ontology

It is surprising how seldom the notion of information system development is defined in the ISD literature. Either it is just "taken as granted" or only characterized with general outlines. Where definitions are provided, conceptions about the nature, structure and behavior of ISD vary, partly due to commitments to different paradigmatic assumptions or ISD approaches. Quite naturally, whether seeing ISD, for example, as a transformation process, as a decision making process, or as a learning process becomes more or less visible in a way specific concepts and views are adopted, emphasized and organized in the definitions. The purpose of this section is first to briefly review definitions presented in the literature for ISD and to bring out our definition for the notion. Second, we define the ISD ontology and describe its overall structure.

ISD is regarded as a "systematic" (Baskerville 1996, 9), "collective" (Korpela *et al.* 2000, 198), "consistent and effective" (Harmsen 1997, 314) action. It is seen as a change process (Welke 1982; Lyytinen 1986; Mathiassen 1998; Tolvanen 1998) that is composed of actions, such as "identifying, analyzing, designing and implementing" (Jayaratna 1994, 214), or "analysis, design, technical implementation, organizational implementation and [..] evolution" (Iivari 1991, 250). Heym *et al.* (1992a) recognize, not only the main ISD stages, but also supportive actions in their definition: "….covers all aspects such as systems specification, project management, quality assurance or risk management from strategic planning, analysis, design, construction, and installation to maintenance of an information system" (ibid p. 215).

One of the most comprehensive definitions is presented in Welke (1981). The definition has been further elaborated e.g. in Lyytinen (1986), Mathiassen (1998) and Tolvanen (1998). In its most commonly used form, the definition resembles that of Lyytinen (1986): ISD "is a change process taken with respect to object systems (target) in a set of environments by a development group to achieve and/or maintain some objectives" (ibid p. 74). Although this definition addresses many important aspects of ISD, it is still limited.

In this work we pursue a definition that is comprehensive and neutral. Comprehensiveness means that the definition should address all the contextual aspects of ISD. Neutrality means that the definition should not exclude the use of any paradigm or ISD approach. This is an important property because the definition should serve as a foundation for engineering the ISD ontology that can be used equally, regardless of the selected ISD paradigm or ISD approach. That does not, however, mean that the definition must address views of all the paradigms and approaches. The definition goes as follows:

> *Information system development* is a context in which ISD actors carry out ISD actions, ranging from requirements engineering to implementation and evaluation of an IS, to produce ISD deliverables that contribute to a renewed or a new IS, by means of ISD facilities in a certain organizational and spatio-temporal context, in order to satisfy ISD goals set by ISD stakeholders.

The definition above provides a versatile view on the contextual aspects of a situation in which an information system is developed. Consequently, ISD is much more than a composition of ISD actions[132]. ISD work is guided by ISD requirements and goals that, through elicitations and negotiations, become more and more complete, shared and formal (Pohl 1993, 279). ISD work is carried out by ISD actors with different motives, skills and expertise, acting in different roles in situationally established organizational units. ISD work is composed of various ISD actions, structured in concordance with the selected ISD approaches and ISD method, and following conventions of the organization. The views and principles of the ISD approach are realized with different structures of ISD actions. The application of the transformational approach, for instance, results in sequential IS modeling actions. The use of the decision making approach or the problem solving approach becomes visible in recursive structures of ISD actions that correspond to intelligence, design, and choice (Simon 1960). The learning approach causes frequent iterations of ISD actions. The political approach manifests itself through ISD actions that highlight stages of collaboration and negotiation concerning IS requirements and design options.

[132] We use the term 'ISD context' when we want to emphasize the contextual nature of information system development. Otherwise, we refer to it with 'ISD'.

The final outcome of ISD is a new or improved information system, composed of interacting social and technical components (cf. Chapter 5). In addition, ISD yields a wide range of plans, memos, decisions, diagrams, testing reports, etc. as intermediate and supportive material. ISD work consumes resources (money and time) and is supported by computer-aided tools (e.g. CASE tools). ISD actors, ISD deliverables and ISD facilities are situated in certain locations (e.g. in work sites, rooms, buildings or geographical sites), and are present in certain times.

Based on the definition of ISD above as well as on the underlying ontologies presented in the preceding chapters, we can now define the ISD ontology as follows: the *ISD ontology* provides concepts and constructs for conceiving, understanding, structuring and representing contextual phenomena in ISD.

To give an overview of the basis and structure of the ISD ontology, we present the meta model of the ISD ontology in Figure 74. In the figure we can see that the most focal concept of the ISD ontology is an ISD context. It is a specialization of the generic notion of a context (cf. Chapter 4). An ISD context is a highly complicated conceptual construct, which is composed of concepts of seven contextual ISD domains. Conceptions about an ISD context are influenced by paradigmatic assumptions adopted, as well as by ISD approaches applied. ISD paradigms and ISD approaches, in turn, are specializations of the generic notion of a point of view (cf. Chapter 3). Conceptions are also affected by ISD perspectives based on some system of perspectives (cf. Chapter 6). The paradigmatic framework and the system of ISD perspectives are kinds of rigid frameworks.

In the following sections we elaborate the view of the ISD ontology given above. First, we will define the concepts and constructs within four ISD domains and the most essential inter-domain relationships (Section 8.3). Second, we will specify ISD perspectives and main inter-perspective relationships (Section 8.4).

8.3 ISD Domains

The purpose of this section is to present the ISD ontology through the meta models of the ISD domains and define the concepts and constructs included in the meta models. Due to the scarcity of space, we focus here only on four ISD domains: the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain. In addition, we present an overview of ISD intra-domain relationships. The meta models and the definitions have been derived from those presented in the context ontology in Chapter 4. Concepts that are adapted as such from the underlying ontologies, will not be explicitly repeated here.

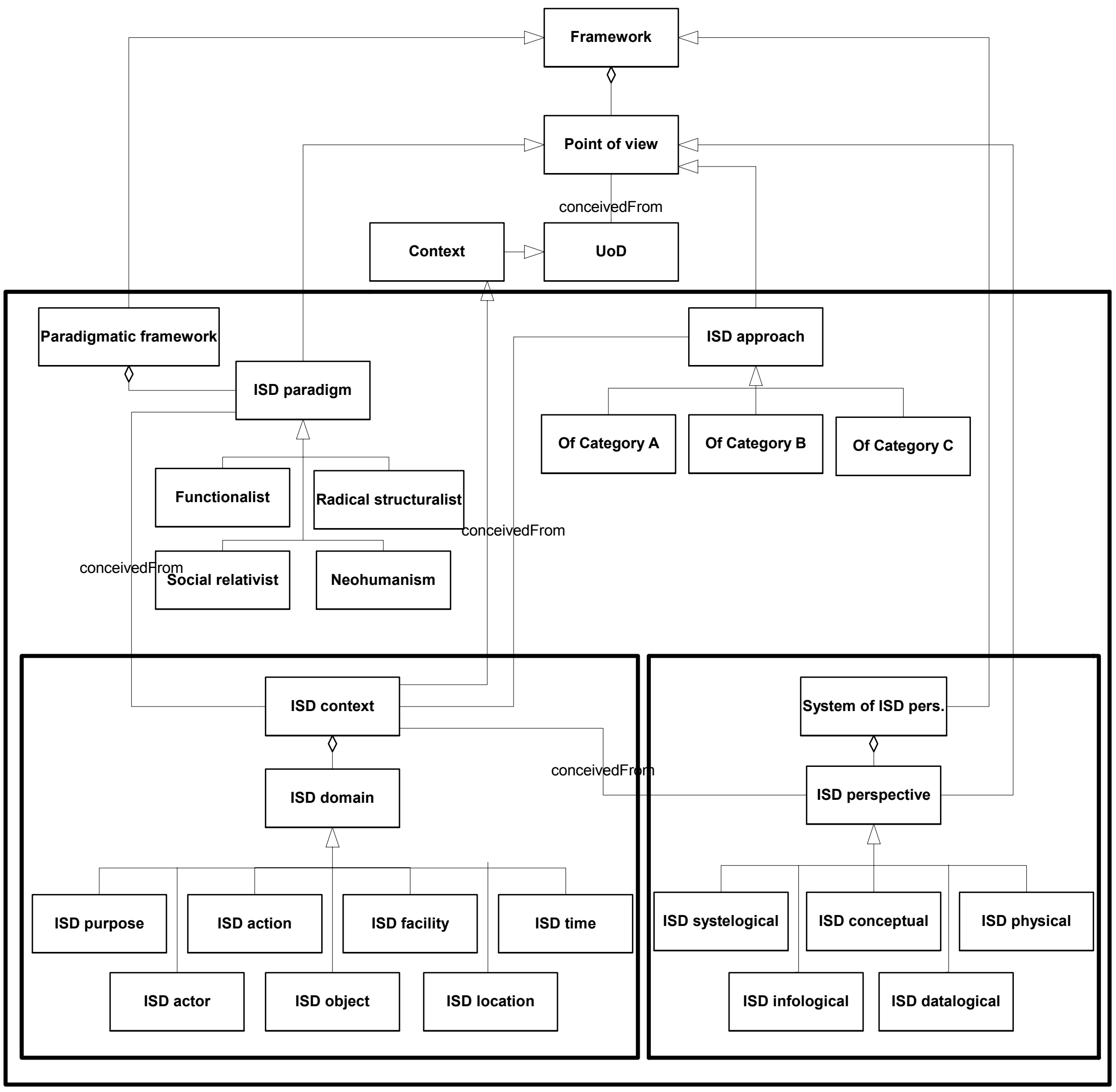


FIGURE 74 Overview of the structure of the ISD ontology

8.3.1 ISD Purpose Domain

The *ISD purpose domain* embraces all those concepts and constructs that refer to goals, motives, or intentions of someone or something in the ISD context. The concepts may show a direction toward which to proceed, a state to be attained or avoided, and reasons for them. Reasons can be expressed in terms of requirements, problems, etc. The ISD purpose domain is highly important because only through its concepts it is possible to demonstrate "Why" an ISD effort is necessary to be accomplished. Correspondingly, reasons can be used to express why certain goals have been set up (from the historical point of view). In the following, we define the main concepts of the ISD purpose domain presented in the meta model (Figure 75).

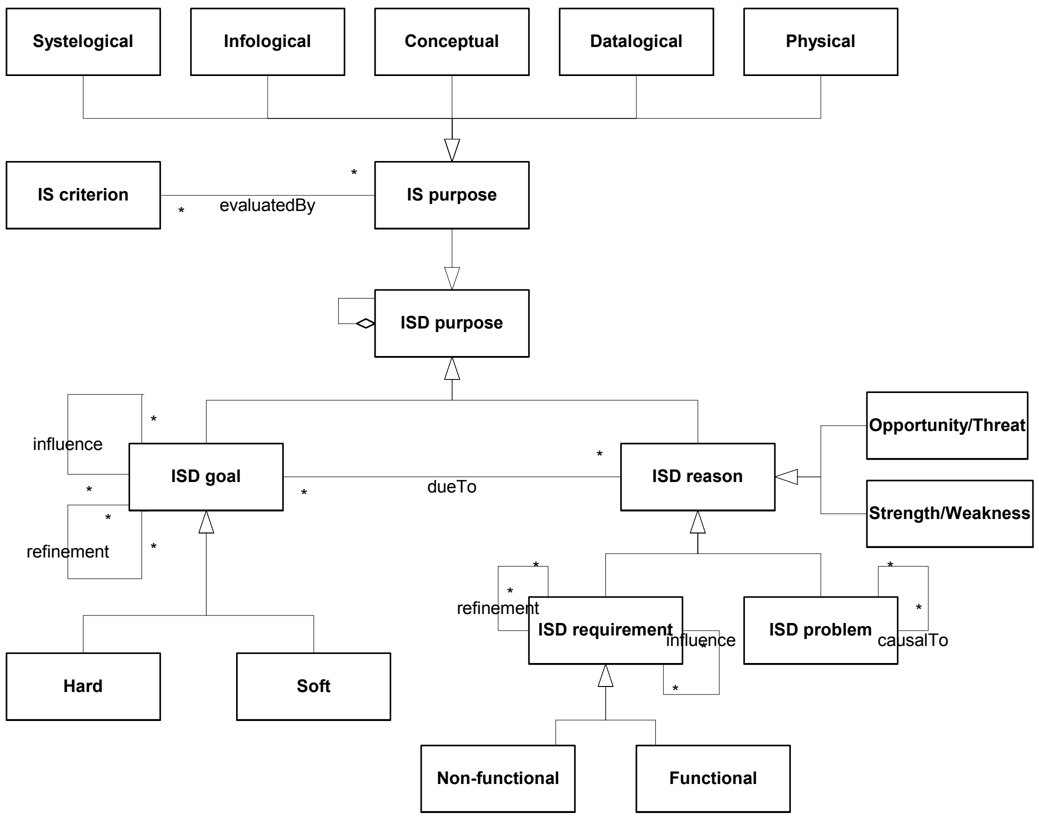


FIGURE 75 Meta model of the ISD purpose domain

An *ISD goal* expresses a desired state or event with qualities and quantities, related to an ISD context as a whole, or to some parts thereof. *Hard ISD goals* have pre-specified criteria for the assessment of the fulfillment of ISD goals, while *soft ISD goals* have not (Mylopoulos *et al.* 2001; Lin *et al.* 1999). An *ISD requirement* is some quality or performance demanded in and for an ISD context. It is a statement about the future (NATURE Team 1996). According to Pohl (1993), ISD requirements can be classified along three orthogonal dimensions: specification, representation, and agreement. In the specification dimension the requirements range from opaque to complete. The representation dimension categorizes requirements into informal, semi-formal and formal requirements. The agreement dimension reflects the fact that ISD requirements initially are personal views, which are negotiated and agreed on to achieve a common view. ISD requirements become goals in an ISD context after having been agreed on. All the requirements cannot be accepted to be goals, since their fulfillment may, for instance, go beyond the resources available. An *ISD problem* is the distance or mismatch between the prevailing ISD state and the state reflected by the ISD goals. ISD problems can be structured, semi-structured or non-structured.

Some of the ISD purposes concern an IS. They are called the IS purposes (cf. Section 6.3.2) and are further sub-divided into IS goals and IS reasons, and furthermore into IS requirements, IS opportunities/threats, and IS strengths/IS weaknesses. IS goals are defined to guide the ISD actors in selecting and implementing IS requirements. For the evaluation and comparison of IS designs, implementation and use, a large variety of IS criteria can be used. An *IS criterion* is a standard of judgment presented as an established rule or principle for evaluating some feature(s) of an IS in terms of IS purposes.

Next, we consider the IS requirements more closely. An *IS requirement* means a condition or capability of the IS needed by an IS client or an IS worker to solve a problem or achieve a goal (cf. IEEE 1990, 62). IS requirements are divided into functional requirements and non-functional requirements. *Functional IS requirements* specify what the IS should do and for whom (cf. Pohl 1993, 280). An example of a functional requirement is: "A user must be able to check his account balance with the help of the CIS". A *non-functional IS requirement* constraints or sets some quality attributes upon the services or functions offered by the IS (Pohl 1994, 247, Cysneiros *et al.* 2001, 97). A non-functional requirement specifies how the IS should function. It can be expressed in terms of performance, safety, quality, maintainability, portability, usability, reliability, confidentiality, security, accuracy, etc. (see Chung *et al.* (2000) and Cysneiros *et al.* (2001, 100) for more comprehensive lists of non-functional requirements).

There are also other classifications for IS requirements. Sage and Palmer (1990) distinguish between technical requirements and managerial requirements (i.e. costs, time constraints as well as quality factors). IEEE (1990) recognizes functional requirements, performance requirements, interface requirements, design requirements, implementation requirements, and physical requirements. The NATURE Team (1996, 516-517) divides the IS requirements into two subtypes: user-defined and domain-imposed requirements. User-defined requirements arise from clients' requests, whereas domain-imposed requirements are facts of nature and form the connection between the real world and the system to be built. These requirements include e.g. social, organisational and technical contexts. The IS requirements can also be classified on the basis of the rationale, i.e. reasons for which they are presented and considered important. These reasons associate the ISD context with other contexts. Sutcliffe (1996) distinguishes between policy-driven requirements, problem-initiated requirements, requirements by example, and the requirements imposed by the external environment.

In this work we apply the IS perspectives to categorize the IS purposes into IS systelogical, IS infological, IS conceptual, IS datalogical, and IS physical purposes. Next, we present the perspective-based definitions for the IS requirements. The *IS systelogical requirements* concern (e.g. the benefits and costs of) information services the IS should provide to its utilizing system[133]. These requirements are specified by senior management (e.g. financial constraints in

[133] ISD is here seen a context which primarily aims to acquire or improve an information system. Often in parallel to ISD there is an effort going on, which pursues to change the utilization system (cf. business process re-engineering). The IS systelogical requirements may reflect needs for improvements in an existing IS as well as in the utilization system.

terms of budget) and IS clients (e.g. required functionalities). The *IS infological requirements* express demands on the type and quality of information needed as well as actions with which the information is to be processed. The *IS conceptual requirements* pertain to the contents of information to be processed in the IS. The IS infological and IS conceptual requirements are specified by IS clients. The *IS datalogical requirements* concern e.g. how to present information, how to divide information processing between persons and computers, and how to organize responsibilities for information processing into IS roles and IS positions. These requirements are affected by the IS workers' views. The *IS physical requirements* are detailed demands on physical structures and behavior of the HIS and the CIS. These are derived and further refined from non-functional requirements expressed in terms of job satisfaction, response time, memory use, security level, etc.

The ISD goals, as well as the ISD requirements, are related to one another through refinement relationships and influence relationships. A *refinement relationship* means that an ISD goal can be reached when certain ISD goals, also known as satisfying or argumentation goals (Cysneiros *et al.* 2001, 102), below it in the ISD goal hierarchy are fulfilled (Rolland *et al.* 1998, 1056). An *influence relationship* means that an ISD goal has impacts on the achievement of another ISD goal (Loucopoulos *et al.* 1998, Kavakli *et al.* 1999, 192). The influence can be positive or negative. The ISD goals with negative interrelationships are referred to as conflicting requirements (Chung *et al.* 2000, Lee *et al.* 2001). A *causalTo relationship* between two ISD problems means that the appearance of one ISD problem (e.g. lack of human resources) is at least a partial reason for the occurrence of another ISD problem (e.g. delays in ISD deliveries).

The ISD requirements and the ISD goals exist with different status (cf. the agreement dimension in Pohl (1993)). 'Proposed' means that an ISD requirement or an ISD goal is brought out by an individual or a group. 'Signed off' means that an ISD requirement, or an ISD goal, is agreed on. 'Frozen' means that no changes are accepted in an ISD requirement or an ISD goal without new negotiations and agreements.

In the ISD / SE literature, a large variety of requirements for requirements specifications are presented. Sommerville (1998), for instance, states that the requirements should fulfill the requirements of validity, consistency, completeness, realism, verifiability, comprehensibility, traceability, and adaptability. According to Lang *et al.* (2001, 162), the requirements should be concise, design-independent, feasible, precise, complete, consistent, and verifiable. IEEE (1991) states that requirements specifications should be unambiguous, complete, verifiable, consistent, modifiable, traceable and usable during operations and maintenance (see also Krogstie (2002), Firesmith (2003a), and Firesmith (2003b)).

8.3.2 ISD Actor Domain

The *ISD actor domain* consists of all those concepts and constructs that refer to human and active part of an ISD context. Actors own, communicate, transform, design, interpret, code, etc. objects in an ISD context. They are responsible for or responsive to trigger and cause changes in the states of objects. They are also aware of their intentions and capable, at least to some degree, of reacting to fulfill their goals. Next, we define the most essential concepts and relationships in the ISD actor domain presented in the meta model in Figure 76.

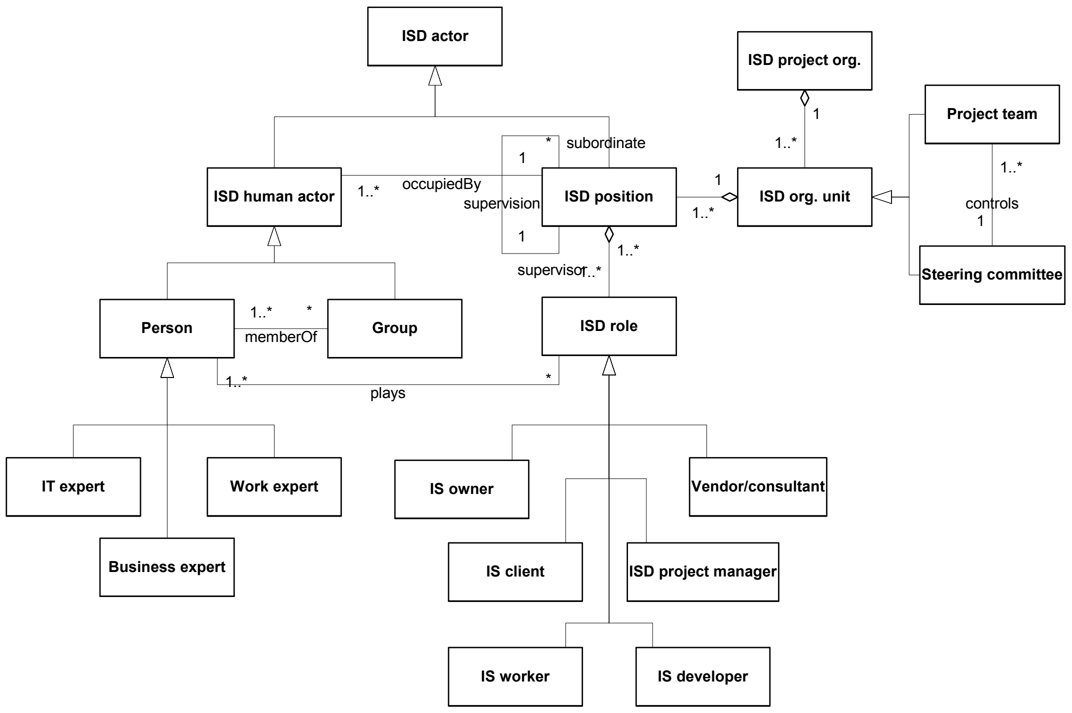


FIGURE 76 Meta model of the ISD actor domain

An *ISD actor* is an ISD human actor or an administrative actor that is, one way or another, involved in an ISD context. An *ISD human actor* means an individual person or a group of persons contributing to ISD work. An ISD administrative actor is an ISD position or a composition of ISD positions. The *ISD position* is a post of employment occupied by a human ISD actor in an ISD context. It is identified with a title, composed of the defined ISD roles, and equipped with a set of skill or capability characterizations (i.e. expertise profile).

A capability means a skill or attribute of the personal behavior, according to which action-oriented behavior can be logically classified (Acuna *et al.* 2004, 678). An *ISD role* is a collection of ISD responsibilities and authorities, stipulated in terms of ISD actions. An ISD position can be hold by several persons. There may exist ISD roles that are not included in any ISD position but are anyhow played by one or more persons. A person may play in several ISD roles.

In the ISD literature, the ISD roles are categorized in various ways. The suggested categorizations can be divided into two groups depending on whether they are based on so-called social roles or technical roles (Constantine 1991). In the following, we first give some examples about the categorizations in the literature and then define the set of ISD roles used in this work.

Divisions into the social ISD roles result from ways of viewing ISD as a problem solving process, a change process, a political process, or a learning process. If ISD is seen as a problem solving process, the major ISD roles are a problem owner and a problem solver (e.g. Vessey *et al.* 1994). If ISD is regarded as a change process, it involves a change facilitator (or a change agent or a shepherd) and a change implementator (Welke *et al.* 1982; Rettig *et al.* 1993, 49). According to the political models, ISD involves self-interest agents employed to perform some services on behalf of the principals (Robey 1984; Markus *et al.* 1987). If ISD is seen as a learning process, it involves a mentor and a student or an apprentice. Further, applying organizational metaphors on ISD (Kendall *et al.* 1993), we can distinguish between the following social roles (the corresponding metaphor is mentioned in parentheses): a player (game), a part or an interchangeable cog (machine), a captain and his crew (journey), a head and members (family), a leader with his troops and an enemy (war).

Hirschheim *et al.* (1989, 1203-) distinguish between four roles of an IS analyst based on paradigmatic assumptions. The IS analyst can be seen as a systems expert, a facilitator, a labor partisan, or an emancipator. An expert takes the objectives and turns them into a system. A facilitator helps users to make sense of the new system and its environment. For a labor partisan there are two antagonistic classes, the owners of the productive resources and labor, and he/she has to choose between being an agent for the former or the latter. An emancipator acts as a social therapist in attempting to draw together the various stakeholders in the ISD.

The divisions into the technical ISD roles result from applying the stakeholder view, the software business view, or the organizational view. In the first case (e.g. Macauley 1993), the roles are based on the division of stakeholders into (1) those with financial interest in, and responsibility for, the systems sale or purchase, (2) those who have an interest in the use of the system, and (3) those who are responsible for the development, introduction, and maintenance of the system. From the viewpoint of the software business, we can distinguish between two partner roles: a customer and a supplier (Franckson 1994). The organization-based ISD roles derive from which system they are representatives of: (1) representatives of the business system, (2) representatives of the information system, or (3) representatives of the object system. The latter role stands for those about whom information is stored and processed in the information system.

In this work, we base our categorization of ISD roles on the works of Checkland (1988), Baskerville (1989, 246), Sabherwal *et al.* (1995, 312) and Mathiassen (1998, 82). We distinguish between five major ISD roles that unify social and technical natures in ISD work. The roles are: an IS owner, an IS client, an IS worker, an IS developer, an ISD project manager, and a vendor/consultant.

An *IS owner* in his/her role has a financial interest in the IS and, thereby, the responsibility for, and the authority of, making decisions on the IS as though it were his/her property. In some cases, e.g., in small software houses, he/she may be the real owner. More commonly, an IS owner might in many cases behave as if he/she possessed the IS, e.g. while, at a general level, were just making the final decisions on goal settings for, and acceptance on, the deliverables. In these cases, he/she acts like a commissioning agent (Brinkkemper 1990, 15). This implies that he/she also has a major power to decide when the current IS should be abandoned and replaced by a new one (Graham *et al.* 1997, 85). An IS owner does not directly intervene in ISD project work, unless the project is so large and important that it has a major impact on the organization.

An *IS client* is the ISD role player for whom the IS is to be developed. He/she is a beneficiary or a 'victim' of the IS (Graham *et al.* 1997, 85). Therefore, he/she is expected to be active in specifying information requirements for the IS in terms of contents, form, time, and media. An IS client also acts as an informant for inquires on business processes, and as an acceptor of the designs of ISD deliverables (cf. the so-called client tests) and plans of re-engineering business processes and work contents (Brinkkemper 1990, 15-16). IS clients usually come from inside the organisation for which the ISD project has been launched. Sometimes, they are stakeholders from the environment, e.g. representatives of the bank needing the salary data in an electronic form.

An *IS worker* works with the current IS and/or is going to work with the new IS. He/she collects, records, stores, transmits, and processes data with or without the help of the computerized information system, in order to produce information needed by IS clients. During an ISD effort, IS workers are expected to express their experience from and requirements for the functionalities, user interface, and information contents of the CIS, as well as to give their opinions about re-arrangements among IS roles and positions designed for a new human information system (HIS). They also actively participate in user tests.

An *IS developer* attempts to meet the needs and requirements put forward by ISD actors in the other roles. For that purpose, he/she analyses IS requirements and IS goals expressed and refines them into more realization-dependent specifications, searches for social and technical solutions and implements those selected. He/she also strives for ensuring that the specifications, designs and implementations are technically acceptable (cf. developer's tests)[134].

An *ISD project manager* makes plans on how to organize an ISD effort. This includes making plans on ISD phases, schedules, milestones, base lines, resource allocations, etc. He/She also participates in making decisions on the execution of the plans. Moreover, he/she is responsible for motivating and inspiring IS workers and IS developers, resolving disagreements, developing standards of performance and methods for the assessments, and establishing reporting and monitoring systems.

A *vendor / consultant* role is played by a person from outside the organization. With the role more expertise on some specific organizational or technical issues are imported to an ISD project. Expertise may be related to

[134] IS developers are referred to as workers and actors in the frameworks of Jacobson *et al.* (1999) and Graham *et al.* (1997, 85), respectively.

technologies (e.g. J2EE platforms, web services), methods (e.g. agile methods), techniques (e.g. TQM) or the like, that is something new to the organization.

The IS clients and the IS workers are users of the IS. Each role can be further specialized. For instance, an IS developer can be specialized e.g. into a business analyst, an IS analyst, an IS designer, and an IS constructor (cf. Olle *et al.* 1988a)[135]. We call the ISD actors who are potentially affected by the IS or ISD and therefore are invited to act in some of the aforementioned ISD roles the *ISD stakeholders.*

The ISD work is mostly organized in the form of a project. An *ISD project* is a temporary effort with the well-defined objectives and constraints, the established organization, the budget and the schedule, launched for the accomplishment of ISD. An *ISD project organization* is a composition of ISD positions, ISD roles and ISD teams wherein the responsibility, authority and communication relationships are defined (cf. Fife 1987).

A large project organization is composed of several organisational units. An *ISD organizational unit* is a composition of ISD positions with a coherent set of organizational goals, authorities and responsibilities. The most common units in ISD are a steering committee, also known as a guidance team (Rettig *et al.* 1993, 46), and a project team. A *steering committee* carries the responsibility for the overall management of the ISD project. The day-to-day management is delegated to the project manager, who directs and controls the actions of specialists in various disciplines. A *project team* is collected for the execution of an ISD effort. If a project is large, there may be a need for several teams acknowledging their share in the common responsibility for developing the IS. The division into teams may follow the sub-systems structure, the phase structure and/or expertise collections (e.g. the technical team, the quality assurance team, etc.). Managerial accountability and responsibilities inside and between the teams may vary depending on the organizational pattern selected. Within each team, the position of a *leader* is devoted to the management of the team and to ensure proper communication between the members, as well as between the team and the other teams. Another essential position in a team is that of a *secretary.*

Some of the positions and roles in an ISD project are full-time vacancies due to the amount of responsibilities and time they require. This is commonly the case for the ISD project manager in a large project. Some other positions and roles do not require full-time commitment. For instance, IT experts can participate in more than one ISD project, and IS clients can participate in projects while they carry out, at least partly, their daily work included in their positions in the business system, in the information system, or in another organization.

For each ISD position the most suitable person is sought. For being suitable the person's skill and experience profile has to match with the expertise profile stated for the ISD position (cf. Acuna *et al.* 2004). Sometimes, no person with the required qualifications can be found from inside the organization, and

---

135 See Kruchten (2000, 263) for 28 different sub-roles of an IS developer.

thus an expert (e.g. a consultant) from another organisation is hired. According to their expertise, the persons involved in ISD can be categorized into IT experts, business experts and work experts. *IT experts* are persons whose education, skills, experience, as well as their former positions, are related to information technology and/or ISD methods. B*usiness experts* are knowledgeable in business strategies, policies, markets, competition, trends, legislation, etc., in other words, in matters relating to how to make business, in general or in the organization. *Work experts* master daily routines, e.g. in making orders, invoicing, production planning, inventory control, goods deliveries, etc. The work experts can be further categorized according to whether their expertise concerns the US_{IS}, IS, or OS_{IS}.

Participation of IS clients and IS workers in an ISD context can take a number of various forms. Traditionally three main forms are distinguished: consultative, representative and consensus (Mumford 1981). With the consultative development, a project team consults with IS clients and IS workers, particularly about their information requirements and job satisfaction needs. The primary actions of analysis, design and implementation are, however, carried out by IT experts. With the representative development, a project team is formed of representatives of the IS clients and the IS workers. The aim of these representatives is to actively participate in the development of a new system. The consensus development attempts to involve, not only the representatives, but all the IS clients and IS workers into the active ISD work. More refined types of participation are presented by Cotterman *et al.* (1989) based a cube with three dimensions (operation, development, control) and Heller (1991) based on different ways of sharing power and influence (see also Krogstie *et al.* 1996, 286). More recently, Markus and Mao (2004) propose theoretical foundations to recognize different roles in which IS clients and IS workers may participate in ISD.

8.3.3 ISD Action Domain

The *ISD action domain* comprises all those concepts and constructs that refer to deeds or events in an ISD context. *ISD actions*, also known as ISD functions, ISD activities, ISD tasks, and ISD operations, are carried out to manage and execute a part of an ISD effort. They customize, incorporate, and implement given procedures, rules and policies to produce desirable ISD deliverables. The ISD actions may involve e.g. knowledge acquisition on problems encountered in the existing IS, generation of design options and selections among them, creation and representation of IS models at various levels of abstraction, verification and validation of their consistency, implementation of models into concrete software and hardware architectures, etc. To manage this extensive variety of ISD actions, we apply a set of views to establish a fundamental categorization of ISD action structures. Each view guides one to conceive the ISD action domain from a particular perspective. In the following, we make a short review of the categorizations presented in the ISD literature. Then we introduce our categorization and define the concepts and constructs for each ISD action structure based on this categorization.

Several categorizations of ISD actions and ISD processes have been presented in the literature. One of the most well-known is the perspective-based categorization of software process models by Curtis *et al.* (1992, 77). The perspectives are: functional, behavioral, organizational, and informational. The functional perspective represents what process elements are being performed, and what flows of informational entities (e.g. data, artifacts, products) are relevant to these processes. The behavioral perspective reveals when process elements are performed (e.g. sequencing), as well as aspects of how they are performed through feed-back loops and iterations. The organizational perspective represents where and by whom (agents) in the organization process elements are performed, physical communication mechanisms used for transfer of entities, and the physical media and locations used for storing entities. The informational perspective represents the informational entities produced or manipulated by a process, their structure and the relationships among them. As we can see, the categorization of Curtis *et al.* (1992) involves, not only the ISD actions, but also various phenomena (e.g. ISD actors, ISD deliverables, ISD locations etc.) related to the ISD actions. This way of categorization is unsuitable for our purposes here.

Dowson (1987) presents a categorisation of process models that is more focused on the ISD action domain. He distinguishes between three kinds of process models: the activity-oriented models, the product-oriented models, and the decision-oriented models. The activity-oriented models derive from an analogy with problem-solving-in-large in which finding and executing a plan of actions leads to the solution (cf. Schmitt 1993, 233). The most common approach in the activity-oriented models is the transformation approach according to which ISD is seen as a sequence of transformation steps from an initial representation of the required real world, through a sequence of intermediate representations, culminating in the delivered system (cf. Moynihan 1993; Tracz *et al.* 1993). Transformation can also be interpreted from the viewpoint of feedback control systems (Weide *et al.* 1993). Examples of this kinds of models are the waterfall model (Royce 1970), the spiral model (Boehm 1988), the hierarchical spiral model (Iivari 1990b) and the fountain model (Henderson-Sellers *et al.* 1993). In the product-oriented models, outcomes of ISD are used to define an IS as being in a particular state of evolution (Glasson 1989; Tomiyama *et al.* 1989). According to the decision-oriented models, ISD is seen as a complicated design decision composed of many smaller ones. Execution of these small sub-decisions creates dependencies among them corresponding to input and output components (cf. White 1982; Iivari *et al.* 1987; Potts 1989; Wild *et al.* 1991; Jarke *et al.* 1990).

More specific categorizations of actions are specified on the bases of group behavior (Hutching *et al.* 1993), dialog (Finkelstein *et al.* 1988), and cooperation (Jarke *et al.* 1993). Building on group behavior Hutching *et al.* (1993) distinguish between five phases of group development: forming, storming, norming, performing, and adjourning. Finkelstein *et al.* (1988) propose a formal framework for understanding program development as cooperative work. The framework is based on a dialogue paradigm with three essential constructs: acts, events, and commitments. Acts are used to reflect assertions, denials, questions, withdrawals, challenges, and resolution demands occurring in the conversations. Jarke and Pohl (1993) see ISD as cooperation between agents. Their model organizes this cooperation to conversation structures, leading to decisions that define and select actions.[136]

All these views help decomposing the ISD work into action structures but in quite different ways. It is difficult or even impossible to integrate them into a unified view on the ISD action domain. We recognize five fundamental ISD action structures that together are comprehensive enough to cover all the essential aspects of the ISD action domain, and deploy them as orthogonal to one another. The ISD action structures are: the ISD management – execution structure, the ISD workflow structure, the ISD phase structure, the ISD problem solving structure, and the IS modelling structure. In addition to these, the generic action structures (i.e. the decomposition structure, the control structure, and the temporal structure) defined in Section 4.4.3 are "inherited" by the ISD action domain. The aforementioned ISD action structures give a natural basis for specializing and decomposing ISD work into more specific ISD actions. Each ISD action is governed by one or more *ISD rules* with the ECAA structure composed of ISD events, ISD conditions and ISD actions (cf. Section 4.4.3). An instance of an ISD action is called an *ISD process*.

Next, we will define the ISD action structures and the concepts contained in them. After that we form an integrative view by considering how the ISD action structures are intertwined with one another. In each part we refer to the literature to show how the ISD structures are recognized, named and decomposed there. An overall view of the ISD action structures and concepts is presented in Figure 77 in the form of the meta model of the ISD action domain.

A. ISD Management–Execution Structure

From the viewpoint of the *ISD management–execution structure* ISD is seen as a functional and behavioral unity, composed of two kinds of actions, ISD management actions and ISD execution actions. *ISD management actions* aim to organize, staff, direct, implement and control ISD work. These actions comprise, for instance, making a project plan (i.e. a work breakdown, a schedule, resource allocation, etc.), determining its adequacy, consistency and feasibility, establishing a quality control mechanism, and re-organizing the plan in cases in which it does not match the reality. ISD management actions also involve planning, acquiring and allocating the resources for an ISD project.

ISD execution actions aim to produce the required ISD deliverables under the guidance and control of ISD management. These actions include, for instance, knowledge acquisition about the existing IS and problems

[136] See more categorizations in e.g. Barros (1991, 539) and Rubenstein-Montano *et al.* (2001).

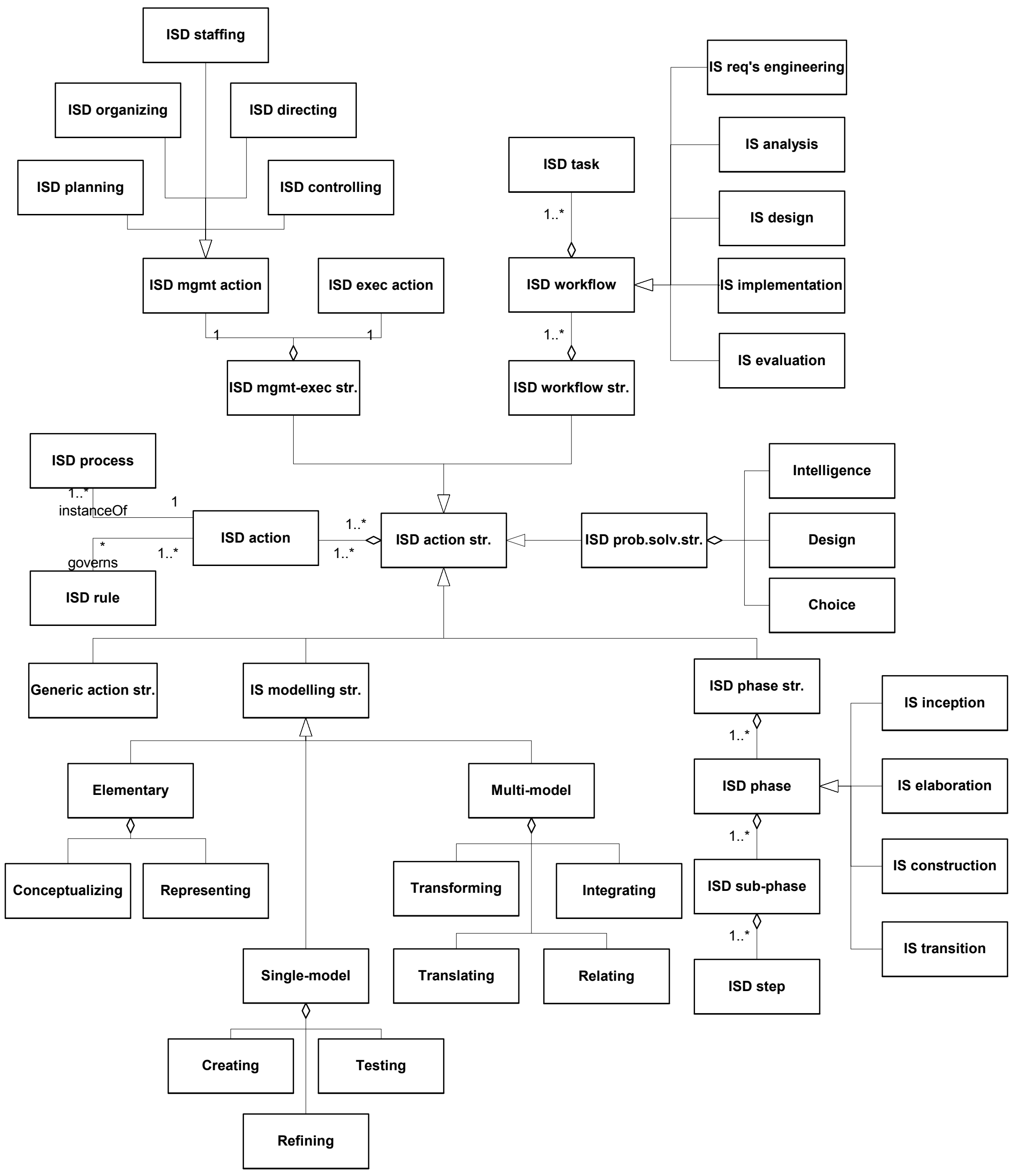


FIGURE 77 Meta model of the ISD action domain

encountered there, requirements specification for a new IS, and design and implementation of specifications into a working system. Besides the actions directly contributing to the deliverables, ISD execution actions comprise supporting actions, for instance training and guidance of users, installation of computer-aided development environments, etc.

The ISD management actions and the ISD execution actions constitute a highly complicated structure in which they appear recursively within one another on multiple levels. These management or control levels are identified with different names. Essink (1988, 361), for instance, distinguishes between two control levels, which he calls the process of IS planning and managerial control, and the process of approach selection. The former corresponds to daily activities of project management, and the latter to the decision making on approaches to project management, modelling, and validation. The SYDPIM model (System DYnamics Project-management Integrated Model) Rodrigues *et al.* (1997, 56) distinguishes two levels of management processes that are the operational level and the strategic level. At the operational level ISD work processes are monitored and new or revised plans with estimations and risk analysis are produced for engineering processes. At the strategic level strategic decisions and risk analysis are made.

The ISD management actions can be further specialized into ISD planning, ISD organizing, ISD staffing, ISD directing, and ISD controlling (see Section 4.4.3). *ISD planning* refers to all those ISD management actions that specify the goals of an ISD project and the strategies, policies, programs and procedures for achieving them (cf. Thayer 1987, 21). These involve partitioning managerial and technical requirements into measurable actions and tasks, determining milestones, priorities and schedules, estimating necessary resources and figuring them as a budget.

ISD organizing refers to all those ISD management actions that are needed to design a formal structure of ISD execution actions and authority relationships between them. These comprise aggregating actions into ISD roles and ISD positions, establishing an organisational structure, and specifying titles, scope, duties, qualifications and relationships of ISD positions.

ISD staffing refers to all those ISD management actions that are needed to fill the ISD positions of the ISD project organization and to keep them filled. These comprise recruiting qualified people, orientating them into technical and social environment, educating them in required methods, skills and equipment, evaluating personnel, determining salary scale, promotion policy etc.

ISD directing refers to all those ISD management actions that are needed for clarifying the assignments of ISD personnel, assigning actions to organisational units, teams and individuals, motivating and inspiring personnel, resolving disagreements between personnel and between the ISD project and outer stakeholders.

ISD controlling refers to all those ISD management actions that are needed for ensuring that actual actions are executed according to the plans. These develop standards of performance and methods for the assessments, establish reporting and monitoring systems, measure and audit progress and status of a project as well as quality and quantity of deliverables, and initiate corrective actions.

In the literature, the ISD management–execution structure is commonly recognized but with different views and concepts. Van Slooten and Brinkkemper (1993, 179) divide the ISD processes into three categories: primary processes that transform inputs to output, regulative processes, like policy making, planning and control, and maintenance processes that obtain and maintain the means of the organizations. For structuring the system development, Wijers (1991, 14) presents a framework for categorizing the types of activities for the solutions of ISD problems. In the framework he distinguishes between 'way of working' and 'way of controlling'. The former deals with the identification of the relevant tasks in the development process and determining their feasible order. The latter includes planning for the ISD project and plan evaluation. Rodrigues *et al.* (1997, 56) distinguish between the engineering process and the management process. Mathiassen *et al.* (1988, 9) and Mathiassen (1998), in the basic activity model of software development, divide the systems development into two categories of activities: performance and management. The performance activities are oriented towards software products and services, whereas the management activities are oriented towards the process of producing them. Cronholm *et al.* (1999, 222) distinguish between the target domain and the project domain. The former corresponds to the execution part of ISD, and the latter stands for project management.

B. ISD Workflow Structure

According to the *ISD workflow structure* ISD is composed of various ISD workflows. An *ISD workflow* is a coherent composition of ISD actions, which are organised to accomplish some ISD process, which share the same target of action, and which produce valuable results for stakeholders[137]. A part of an ISD workflow is called an *ISD task*. ISD workflows can be identified among the ISD management actions as well as among the ISD execution actions. In the following, we will consider them in the context of the ISD execution actions. We distinguish between five core ISD workflows: IS requirements engineering, IS analysis, IS design, IS implementation, and IS evaluation[138].

IS requirements engineering means an ISD workflow, which aims at the identification and elicitation of IS clients' and IS workers' requirements on the IS, as well as establishing and maintaining, at least to some extent, agreement on what the information system should do and why. This necessitates that the ISD actors have general understanding of the problem area and the scope of the IS (cf. Kruchten 2000, 155). The IS requirements engineering is commonly decomposed into feasibility study, requirements analysis, requirements definition, and requirements specification (Sommerville 1998, 67).

IS analysis means an ISD workflow, which models the problem domain. The focus of this workflow is to represent the business system in a manner that is natural and concise enough, and to achieve an overall description of the information system that is easy to maintain. The IS analysis aims to ensure that the information system's functional requirements are covered. In this sense, analysis starts with looking at the system from outside (Mathiassen *et al.* 2000, 13).

---

137 Note that the term 'workflow' is here used in a different meaning from the one used in the UML process model (Jacobson *et al.* 1999) or in the workflow management literature (cf. Workflow Management Coalition 1999; Mentzas *et al.* 2001).

138 In some frameworks (e.g. Iivari 1991, 250) there are also workflows for maintenance or evolution of an IS. We regard them as being composed of ISD actions of the core ISD workflows and ignore them here

IS design means an ISD workflow, which models the solution domain. It involves elicitation, innovation and evaluation of design options in the form of IS models on various levels of abstraction. Thus, the IS design looks at the system from inside. A decision is made on which part of the system will be automated (cf. CIS) and which part is to be implemented as a manual system (cf. HIS). The workflow aims to acquire an in-depth understanding of issues regarding non-functional requirements and constraints related to components reuse, software architectures, hardware platforms, user-interface technologies, etc. Good designs are not deduced, they are invented. Thus, a creative and intellectual element is most essential for design (Fairley 1985; Lanzara 1983).

IS implementation means an ISD workflow, which fleshes out the architecture and the system as a whole, by carrying IS design models into effect. There are two kinds of implementation actions. Technical implementation, known as construction in Iivari (1991, 250), involves all those actions that are necessary to construct/acquire and carry into effect technical components of the CIS. These generate and code software procedures, acquire and assemble hardware components into computer and communication systems, specify and load files and databases, etc. Organizational implementation, referred to as institutionalisation in Iivari (1991, 250), means all those actions that are necessary to create and change social norms, conventions, procedures and structures to be embedded in the HIS.

IS evaluation means an ISD workflow, which aims at the assessment of an existing system, as well as of all the specifications, designs and implementations made for the future system. Evaluation is based on quality criteria derived from the functional and non-functional requirements. Evaluation comprises verification and validation. Verification is a process of determining whether or not the ISD deliverables, produced by ISD work, fulfill the established requirements. Validation is a process of evaluating the IS at the end of the ISD work to ensure compliance with the requirements (cf. Boehm 1984).

Besides the core workflows defined above, there are supporting workflows, like configuration and change management (cf. Kruchten 2000). These are not discussed here.

As can be seen from the definitions above, there are no clear-cut borderlines between the workflows. In the ISD literature, analysis and design are the most commonly referred parts of the ISD work. To illustrate differences in their meanings and deviations in the conceptions about how they are related, we present a short review of the literature. We can distinguish between five views on which the dichotomy of analysis and design is defined in the literature. According to the first view, analysis continues until a decision can be made on whether design is necessary or profitable. This means that analysis also yields at least informal requirements for the IS (e.g. Brodie *et al.* 1982) or even narratives of required functions and building blocks of the system (e.g. Maddison *et al.* 1984; Colter 1984). According to the second view, analysis means finding out "Why" and "What" the system is supposed to do, and design means "How" this should be happened (Alabiso 1988; Wand 1988a, 203; Zultner 1993; Vidgen 2002, 249). The third view makes a difference between analysis and design on the basis of the object system of the actions: analysis involves an existing system, and design focuses on a new system (cf. Olle *et al.* 1988a; Mathiassen *et al.* 1988, 7; Jayaratna 1994, 244). The fourth view considers analysis to be something which takes apart and describes things, whereas synthesis is a constructive action by which the known parts are put together in a new way (cf. Mathiassen *et al.* 2000, 14; Harmsen 1997, 138). The fifth view emphasizes the seamlessness of analysis and design, meaning that analyzing a problem leads automatically to thoughts of a like solution (cf. Graham *et al.* 1997, 41-42).

The first view is based on the phase-oriented perspective, whereas in the third view a distinction is made based on the target of the action (an existing system vs. a new system). These kinds of conceptions are common in conjunction with the waterfall model (Royce 1970). In recent years, the concepts of workflow and phase have been clearly separated (cf. Jacobson *et al.* 1999). This is best enabled by the second view. According to this view, the analysis workflow as well as the design workflow contains analytical and synthetic actions, yet from different perspectives. This is the view we advocate here.

C. ISD Phase Structure

According to the *ISD phase structure,* the ISD is seen as being composed of sequential phases. An *ISD phase* means an ISD action, executed between two milestones, by which a well-defined set of goals is met, ISD deliverables are completed, and decisions are made on to move or not to move into the next phase (cf. Kruchten 2000, 276). *Milestones* are synchronization points where ISD management makes important business decisions and ISD deliverables have to be at a certain level of completion (Heym *et al.* 1992a, 230). Major milestones are used to establish baselines (see Section 8.3.4 for the definition of a baseline).

In ISD methods, a large variety of phases with different names are presented. Without wanting to commit to any of them, we have selected the set of phases, suggested by Jacobson *et al.* (1999) and Kruchten (2000), as an example of the ISD phase structure[139]. It comprises four phases: IS inception, IS elaboration, IS construction, and IS transition.

In the *IS inception phase* the focus is on understanding the overall requirements and determining the scope of the development endeavor. The scope is needed to understand what the architecture has to cover, what the critical risks are, and to provide the boundaries for costs and schedule, as well as the return-on-investment estimates. All in all, the IS inception phase determines the feasibility of the proposed system development.

[139] We are fully aware of a large variety of process models (e.g. life cycle model, spiral model, fountain model) and approaches (e.g. prototyping approach, evolutionary approach, incremental approach, agile approach), as well as of the fact that in each of them different phase structures are applied. Although the phase structure of Kruchten (2000) is not conceptually the best, we are here satisfied with that.

In the *IS elaboration phase* the focus is on detailed requirements engineering, but some systems design and implementation actions aimed at prototyping can also be done. Prototyping is deployed to better understand IS requirements, to test the established architecture, thus mitigating certain technical risks, and/or to learn how to use certain tools and techniques. The phase ends with the baseline for the next phase.

The *IS construction phase* focuses on design and implementation of the system. During this phase a software product is produced, which is ready for the initial operational release that fulfills the given requirements. Also plans for organizational changes are "operationalized" for realization.

The *IS transition phase* is entered when at least some part of an ISD baseline is mature enough to be deployed. The phase comprises beta testing, fixing bugs, adjusting features, parallel operations with the legacy system, conversion of operational databases, training of users and maintainers, etc. At the end of the phase the final product (CIS) has been delivered and the new organizational arrangements (HIS) are fully in operation.

The ISD phases comprise, besides ISD actions described above, also some method engineering actions. Especially in the first phase but also at the beginning of the other phases it is common to customize the selected method to make it better fit the ISD context at hand (Nuseibeh *et al.* 1996; Mathiassen 1998; Tolvanen 1998). These ME actions are discussed in Chapter 10. In some ISD approaches an ISD phase structure is established to include some IS actions as well. In the prototyping approach (Budde *et al.* 1984) and especially in the evolutionary approach (Iivari 1982; Falkenberg *et al.* 1992a) ISD actions and IS actions are highly intertwined. We do not discuss this issue any further here.

D. ISD Problem Solving Structure

The *ISD problem solving structure* is the result of seeing the ISD as a series of interrelated decisions, which involve the identification and articulation of problems, alternative solutions, decisions and justifications (cf. Wild *et al.* 1991, 18). There are approaches which lay more emphasis on problems (e.g. Bodart *et al.* 1983; Sol 1992; Blum 1994; Jayaratna 1994), and approaches for which the decision is the focal element (e.g. Jarke *et al.* 1990; Wild *et al.* 1991; the NATURE Team 1996). With the ISD problem solving structure we aim to cover both of these approaches[140].

According to Simon (1960), problem solving is composed of three kinds of stages: intelligence, design, and choice. *Intelligence* means actions that search the environment for conditions calling for a decision. In the ISD context, this means the recognition of problems and the acquisition and analysis of knowledge relevant to resolution of the problems. *Design* consists of the actions of inventing, shaping and specifying alternatives for possible courses of action in

[140] Considering each choice in the problem solving structure to be a decision establishes a structure of decisions, which are associated with one another with relationships that are derived from the corresponding relationships of the problems.

ISD work. If the available information is found to be insufficient, the problem solver (e.g. an IS analyst) may choose to go back to the intelligence stage before making any further move. *Choice* means the evaluation and comparison of each alternative design option and the selection among them. If needed, more information is collected, more options are specified and/or specifications are further refined or revised. Hence, the stages constitute an iterative rather than sequential process.

Simon's framework can be recursively applied within each of the three stages (cf. Cooper *et al.* 1979). This enables us to distinguish between the first-order problems and the second-order problems (cf. Eloranta 1974). To solve the original problem, known as the first-order problem, it is first necessary to find an answer to how to solve it. Intelligence within intelligence, for instance, means the collection of information about possible approaches, objectives and procedures to collect and analyse the information. Correspondingly, design and choice within intelligence means the generation and assessment of alternative means to collect and analyse the information, and the selection of the best one to be applied. In the ISD work, both the IS problems (i.e. the first-order problems) and the ISD problems (i.e. the second-order problems) have to be tackled.

The ISD problem solving structure is seen as being embedded in the ISD at several levels of detail in the literature. Jayaratna (1994, 37), for instance, considers an ISD method as a problem-solving mechanism that shows how to perform problem solving in ISD through three phases: a problem formulation phase, a solution design phase, and a design implementation phase. Wild *et al.* (1991) consider software development as a series of interrelated decisions which involve the identification and articulation of problems, alternatives, solutions and justifications. The design process is characterized by a search by decision dependency directed backtracking. In the NATURE approach (NATURE Team 1996) the requirement engineering process is structured as contexts in which requirement engineers with certain intentions have several options to select from when making decisions on actions.

E. IS Modeling Structures

Modeling has incontrovertibly a focal role in the whole range of the ISD actions. It is a necessary and frequently used means equally in the ISD management actions (cf. organization charts, time tables, etc.) and in the ISD execution work. Here, we focus on modeling in the latter case, and refer to it as *IS modeling.* The target of IS modeling can be an existing IS or a new IS, seen from different IS perspectives. The significance of modeling to ISD appears the most evident in those ISD approaches that regard ISD work as a transformation process by which IS models are transformed into more realization-dependent models (e.g. Wand 1988a; Tracz *et al.* 1993; Moynihan 1993; Jacobson *et al.* 1999). Although modeling does not exert influence on the macro structures of the ISD actions, it is intrinsically present at all lower levels of ISD work. We refer to the structures of actions targeted at the IS models as the *IS modeling structures.*

There are three kinds of IS modeling structures: the elementary modeling structure, the single-model action structure, and the multi-model action structure. The *elementary modeling structure* comprises IS modeling actions that are always present in IS modeling. These actions are conceptualizing and representing (cf. Chapter 7). By *conceptualizing,* relevant perceptions of the existing reality and conceptions of the imagined reality are interpreted, abstracted and structured according to some conceptual model (cf. Falkenberg *et al.* 1988, 47). *Representing* is an ISD action by which conceptions are made "visible" and proper to communicate about them. Representing yields a model denotation from a concept model.

The *single-model action structure* comprises IS modeling actions that involve a single model at a time. These actions are creating, refining and testing. *Creating* is an ISD action by which an IS model is conceptualized and represented for some specific use. It is an initializing action which starts without any previous version of the model. After making the first version of the model, some corrections, modifications and enlargements are often required. Also, actions of abstraction and concretization may be needed. These IS modeling actions are called the model *refining. Testing* is an ISD action by which a concept model or a model denotation is checked against the given quality criteria (cf. Krogstie 1995). Testing comprises validation and verification. Validation means checking that the proper fit between the model and the existing or imagined (conceptions of) reality exists. Verification means checking whether there are any inconsistencies within the model (or among the models).

The *multi-model action structure* comprises IS modeling actions that involve, some way or another, two or more IS models at the same time. These actions are transforming, translating, relating, and integrating. *Transforming* is an ISD action by which conceptions structured according to one IS model are transformed into conceptions structured according to another IS model. For instance, conceptions about data flows structured by the concepts of the DFD model (e.g. Gane *et al.* 1979) can be transformed to conceptions structured by the concepts of the ISAC activity model (Lundeberg 1982). Transforming can also be done through derivation, by strict rules or some heuristics, from one or more IS models to another IS model. For instance, an ER schema is transformed into a relational schema by following a set of simple transformation rules (cf. Elmasri *et al.* 2000). *Translating* is an ISD action by which conceptions represented in some language are translated into another language. For instance, a description of the goal /means relationships can be translated from a graph form to a matrix form. In translating, the semantic contents of the IS model (i.e. the concept model) are supposed to remain the same, while only the presentation (i.e. the model denotation) is changed. This of course is not, strictly speaking, true.

Two or more IS models are *related,* or mapped, to one another by finding common concepts within the models or defining some “bridging” relationships between the concepts of the models. Relating can be total or partial. It does not create any new model. *Integrating* means an ISD action by which a new model is made by assembling together concepts and constructs of two or more other IS models. Integration requires that conflicts in naming (e.g. synonyms, homonyms) and structures are resolved.

Each of the IS modeling actions defined above can be further decomposed and/or specialized. For instance, creating is composed of the following steps: delimitation, identification, defining, characterization, relating, decomposition, specialization, etc. (Goldkuhl *et al.* 1993).

In Figure 78 ISD actions of the IS modeling action structure are illustrated in the setting of two dimensions (cf. Goldkuhl *et al.* 1993, 8). The vertical dimension stands for the perspectives, and the horizontal dimension is established along the IS domains. Transforming concerns two IS models which represent the same or different perspectives. For instance, the transformation of a relational schema from an ER schema means a shift of the view from the IS conceptual perspective to the IS datalogical perspective. Relating and integrating IS models yields a more comprehensive view on the IS context. They can involve models from the same or different contextual domains (e.g. integration of a goal/activity model (Kueng *et al.* 1996) and a data flow model (Yourdon 1989)). In some cases the IS model to be related and integrated can be made from different IS perspectives as well. The other IS modeling actions concern the models of the same IS perspective and the same IS domain.

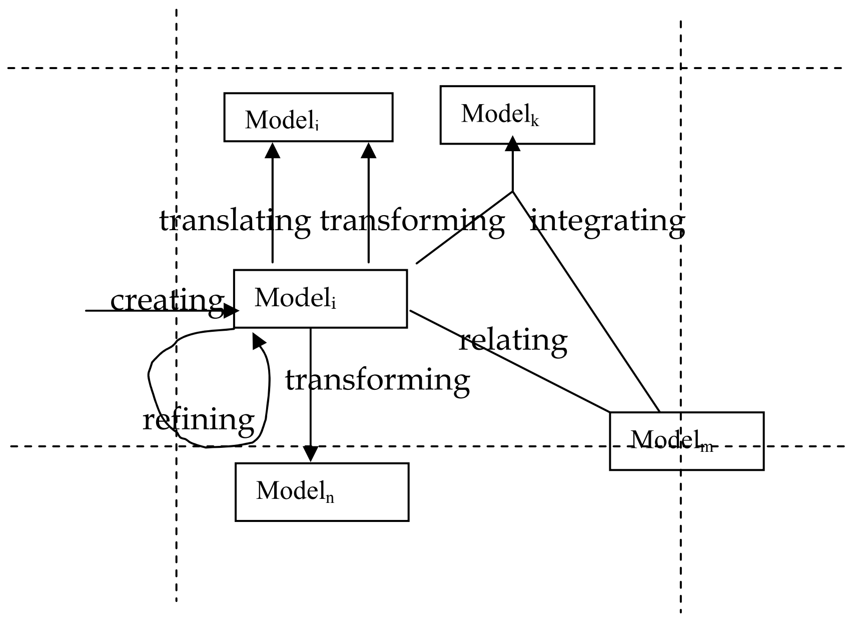


FIGURE 78 IS modeling actions in the vertical (perspective) and horizontal (contextual domain) dimensions

F. Synthesis

The ISD action structures are highly inter-twined with one another. In an ISD project, ISD work may be structured, for instance, into five ISD phases each of which is decomposed into several sub-phases and numerous steps. These phases contain ISD actions from several ISD workflows. Each workflow in turn comprises different IS modeling actions. Because in ISD it is basically a question about decision making, in all ISD phases, ISD workflows, and IS modeling actions we can recognize parts of solving primary and secondary IS/ ISD problems. Also, among ISD actions there appear a multitude of branches and iterations. Besides being succeeded by one another, ISD actions can overlap or be executed in parallel. Keeping this in mind, we can imagine the difficulties encountered in modeling and managing this complexity in ISD, as well as in ISD methods.

In the ISD literature, only one or two of the ISD action structures defined above are usually identified. There are, however, some exceptions. Next, we review two of them. Iivari (1990b) distinguishes between three main categories of IS/SW design structures in the hierarchical spiral model for information system and software development. The categories are: decision making dynamics concerning IS/SW products, learning dynamics, and IS/SW design acts. The first structures are further divided into evolution dynamics and main phase dynamics. Evolution dynamics consists of successive life cycles of the operational IS/SW product at the levels of modeling (i.e. perspectives in our terminology). Main phase dynamics corresponds to our ISD phase structure. Learning dynamics takes into account the fact that the IS/SW design process normally involves continuous learning that presumes making iterations explicit. The category of IS/SW design acts corresponds to lower-level acts, which are completely or partially ordered in time.

Mathiassen *et al.* (1988) present a basic model of software development in terms of seven intrinsic relations: management and performance, reflection and action, analysis and design, knowledge and practice, quality and resource, formal and natural, and actors and bystanders. Four of these are related to the ISD actions. The division into management and performance is based on the process-oriented and product-oriented views on software engineering, respectively. The dichotomy of reflection and action highlights the necessity of effective learning in the cycle of design and realization. Distinguishing analysis and design means that the actions directed at the present reality are separated from those concerning the future possibilities. Knowledge and practice correspond to technical rationality and reflection-in-action (Schön 1983).

8.3.4 ISD Object Domain

The *ISD object domain* comprises all those concepts and constructs that refer to something to which ISD actions are directed. In ISD frameworks these are commonly called deliverables (Glasson 1989; Heym *et al.* 1992a; Cimitile *et al.* 1994), artifacts (Song 1997; Hruby 2000b, 23; Jacobson *et al.* 1999, 21), decisions (Rose *et al.* 1990; Wild *et al.* 1991), products (Aoyama 1993; Saeki *et al.* 1993; Hazeyama *et al.* 1993), work products (Hidding 1997, 105; Firesmith *et al.* 1999; Henderson-Sellers *et al.* 1999c 40), design products (Olle *et al.* 1988a, 2), and increments (Graham 1989). To emphasise the linguistic nature of the ISD objects and our orientation to ISD objects in the execution part of the ISD, we use the generic term *ISD deliverable.*

An ISD deliverable inherits all the predicates of an informational object specified in Section 4.4.4. This means, for instance, that an ISD deliverable can be, on the elementary level, an assertion, a prediction, a plan, a rule, or a command, concerning the ISD itself, the existing information system, the new information system, the OS_{IS} or the US_{IS} (cf. Chapter 5). We use the term *'OS_{ISD} construct'* to denote all these parts in the object systems of ISD. The signifies relationship expresses a relationship between an ISD deliverable and an OS_{ISD} construct.

In the following, we first define the essential classifications of the ISD deliverables and then specify the most substantial relationships between the ISD deliverables. Figure 79 gives an overview of the concepts and relationships within the ISD object domain in the form of the meta model.

The ISD management actions aim to plan, organize, staff, direct, and control ISD work. They produce plans for, decisions on, directives for, and assessments of goals, positions, actions, deliverables, locations, etc. in an ISD context. We refer to these objects as the *ISD management deliverables.* Examples of the ISD management deliverables in the form of documents are: Definition study action plan and schedule, Statement of work for detailed system design, Conversion and installation plan, and Subsystem detailed design report. The ISD management deliverables are intended for the ISD actors, who are in charge of carrying out the corresponding ISD execution actions. Some of the deliverables (e.g. budgets and assessment reports) are for persons on the strategic or tactical level in the US organization. Besides the deliverables disseminated by formal documents, persons in charge of the ISD management guide, motivate, inspire and support their subordinates and colleagues informally through discussions, advice and messages.

The ISD execution actions aim to implement plans and prescriptions got from the ISD management. These actions result in a large variety of descriptions and prescriptions about why, what, and how information processing is carried out or is to be carried out in the current IS context or in a new IS context, respectively. We call these ISD deliverables the *ISD execution deliverables.* The ISD execution deliverables comprise informal drafts and scenarios, as well as more formal presentations. The former include instructions and guidelines, produced for IS actors in the form of training materials, handbooks, and manuals. The latter are presented in *IS models* (e.g. ER schemes, DFD's, program structure charts) or they are *IS implementations* of those models (e.g. software modules, prototypes, files, data bases).

Some of the ISD execution deliverables are specified to be parts of the ISD baselines with milestones in a project plan. An *ISD baseline* is a set of reviewed and approved ISD deliverables that (1) represents an agreed basis for further evolution and development, and (2) can be changed only through a formal procedure such as configuration and change management (Jacobson *et al.* 1999, 443). Because a variety of ISD deliverables is too large to be dealt with here, we will concentrate on the IS models.

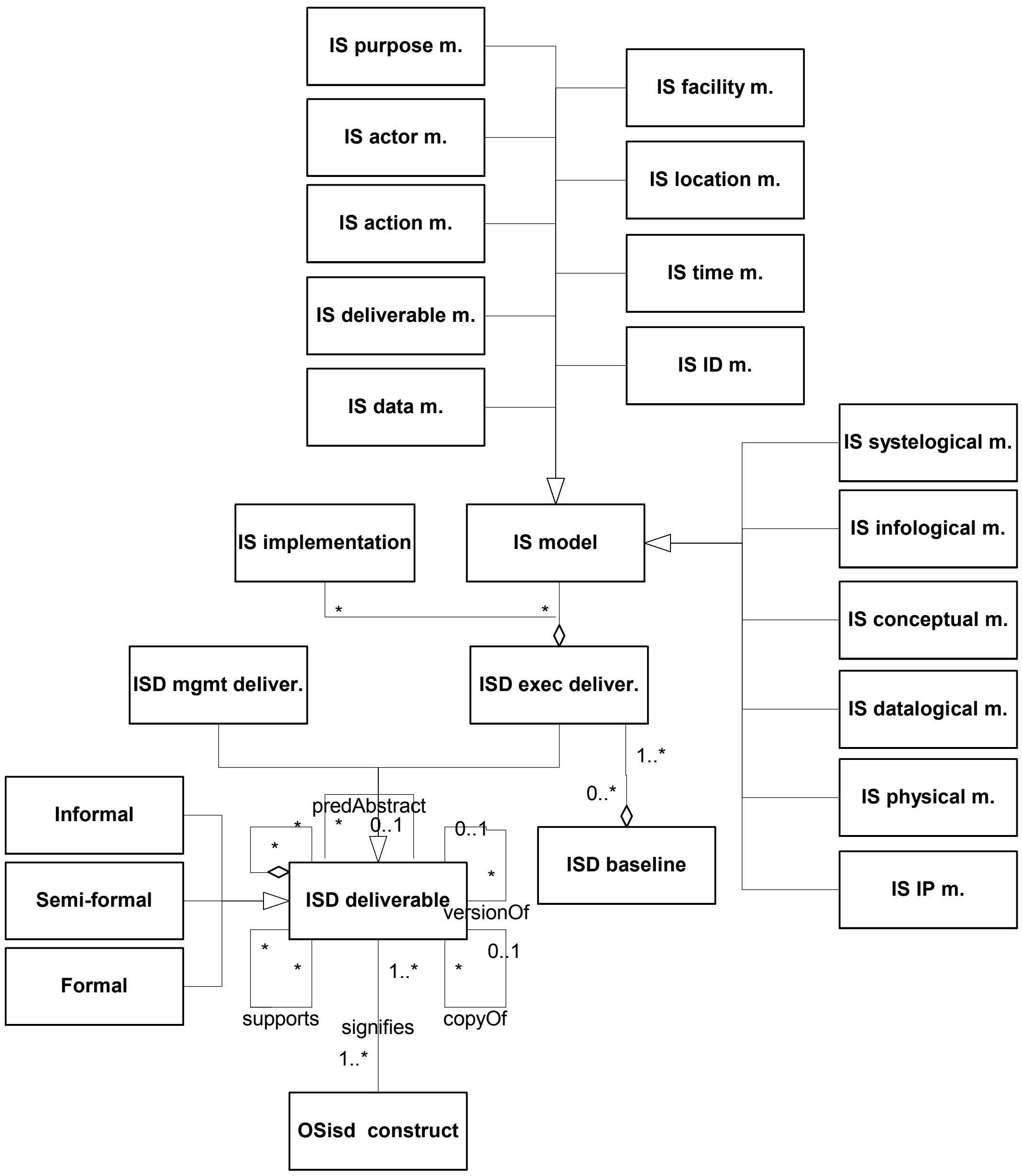


FIGURE 79 Meta model of the ISD object domain

On the basis of the perspective ontology (see Chapters 6 and 7), we distinguish between systelogical, infological, conceptual, datalogical, physical, and inter-perspective (IP) IS models. Furthermore, in accordance with the context ontology (cf. Chapters 4 and 7), we classify the IS models into IS purpose models, IS actor models, IS action models, IS deliverable models, IS data models, IS facility models, IS location models, IS time models, and IS inter-domain (ID) models.

As the ISD deliverables are informational objects, they can be perceived from different semiotic viewpoints. An ISD deliverable can be a conceptual, linguistic, or physical object. Above, we have considered the ISD deliverables mainly to be conceptual with the aim of revealing their conceptual contents.

The ISD deliverables are presented in some language(s). Presentations may be informal, semi-formal, or formal, including texts, lists, matrices, program codes, diagrams, charts, maps, pictures, voices, videos, etc.

Implied from the meta model of the object domain (Section 4.4.4), the ISD deliverables are related to one another with five kinds of relationships. First, an ISD deliverable can be composed of other ISD deliverables. Second, an ISD deliverable can be used as an input to, or as a prescription for, another ISD deliverable (i.e. the supports relationship). For instance, an ER schema is a major input to a relational schema. Third, an ISD deliverable can be the next version of another ISD deliverable (i.e. the versionOf relationship). Fourth, an ISD deliverable may be a copy of another ISD deliverable. Fifth, an ISD deliverable can be more abstract than another ISD deliverable in terms of predicate abstraction (i.e. the predAbstract relationship).

In the literature, there are only few presentations in which the ISD deliverables are addressed in a comprehensive manner. Most commonly the ISD deliverables are classified according to ISD actions, or alternatively in a more or less non-systematic way. Examples of the latter case are classifications in Harmsen (1997, 141) (i.e. requirements statements, specifications, operational items, plans, reports) and in Heym *et al.* (1992a, 227) (system specifications, planning documents, reports or documentation, and decisions). From the relationships between the ISD deliverables the partOf relationship (e.g. 'contains' in Glasson (1989) and Song (1997), 'is_part_of' in Song *et al.* (1992), 'has' in Prakash (1997, 1999), 'composed of' in Schmitt (1993)) and the supports relationship (e.g. 'depends_on' in Glasson (1989), 'output usage' / 'input usage' in Heym *et al.* (1992a)) are most commonly distinguished.

8.3.5 ISD Inter-Domain Relationships

In the sections above the ISD concepts and constructs have been discussed from the viewpoint of one ISD domain at a time. The ISD domains are, however, inter-related in many ways. Figure 80 presents the general-level meta model, which illustrates the most essential inter-domain relationships. In the meta model one or few essential concepts from each of the ISD domains are depicted and related to concepts of the other domains. We omit the cardinality constraints associated to the relationships in the figure to keep it simple. The meta model has been derived from the one in Figure 43 in Section 4.5. It is neither possible here to discuss all the inter-domain relationships in Figure 80, nor to give explicit definitions for them. Instead, we refer to the definitions given in Section 4.5. There are, however, two ISD inter-domain relationships, which we consider here in more detail. These are the viewedBy relationship and the strivesFor relationship. With these relationships we can highlight ISD as an organizational context in which ISD stakeholders have different views and opinions and ISD actions are guided by certain design rationale.

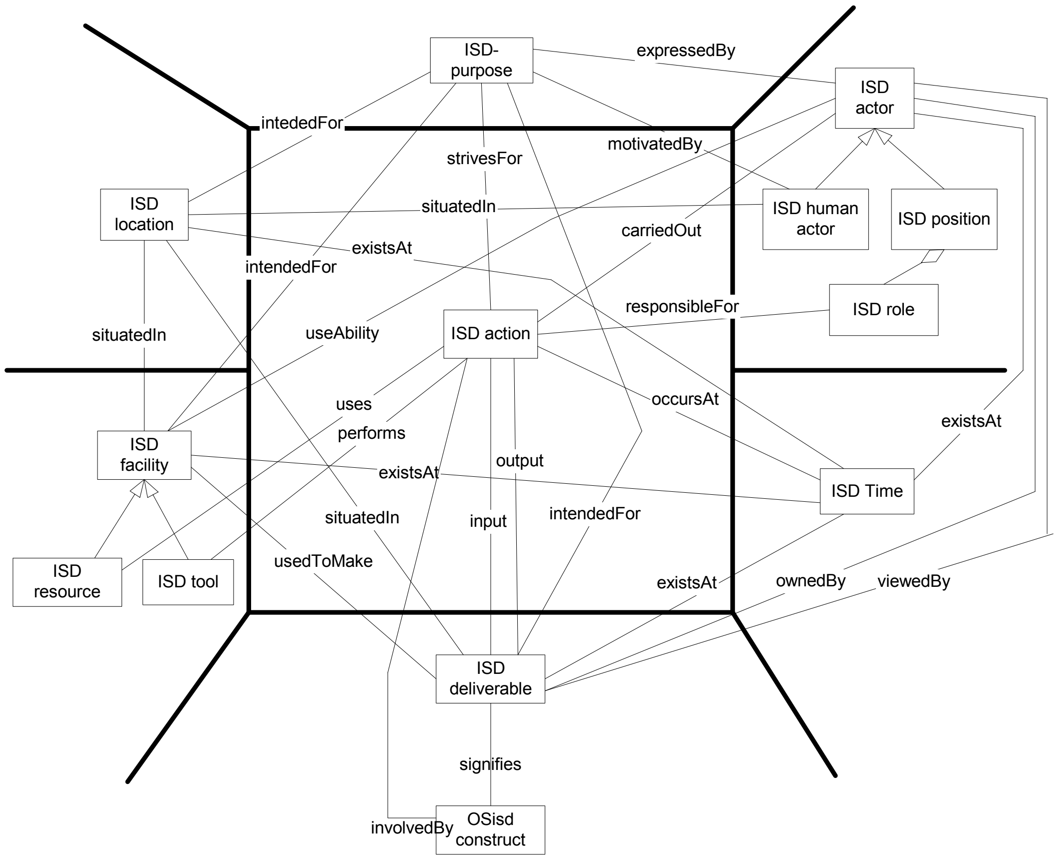


FIGURE 80 Meta model of ISD inter-domain relationships

The viewedBy relationship between an ISD deliverable and an ISD actor means that an ISD deliverable represents views, insights, opinions, etc. of a specific ISD actor. If associated with a person or a group of persons, an ISD deliverable represents subjective or inter-subjective views, whereas if it is associated with an ISD position, an ISD deliverable reflects an organizational view or a so-called 'official' view. According to Stamper (1992b) there is no knowledge without an agent. With this relationship an ISD deliverable can be tied to the person or organization concerned. Through this relationship it is also possible to present differences between, and conflicts among, the views[141]. The significance of this relationship is acknowledged especially in the requirements engineering literature. Lang *et al.* (2001, 166) identify the 'proposes' relationship between 'Stakeholder' and 'User requirement' in the meta model for RM-tool (Requirements Management Tool). Lee *et al.* (2001) argue that the requirements should be incorporated to the stakeholders who have presented those requirements. This is important because of traceability, conflict resolving, prioritisation, etc. Nuseibeh *et al.* (1996) outline the ViewPoints framework, which acknowledges the existence of ISD actors "who hold multiple views on a system and its domain" (ibid p. 267). The multiple views can be specified and managed by the use of the ViewPoint pattern (Finkelstein *et al.* 1992), which is

141 See more in Baldwin (1993) and Motschnig-Pitrik (1999).

related to the ViewPoint owner. The owner acts as the domain knowledge provider.

The strivesFor relationship between an ISD action and an ISD purpose means that an ISD action is to be conducted, is conducted, or was conducted for satisfying a certain goal. A goal may be inferred from encountered problems, specified requirements, observed opportunities, or perceived threats. From the historical viewpoint, the strivesFor relationship, together with the input and output relationships between the ISD actions and the ISD deliverables, can be used to express design rationale (Goldkuhl 1991; Ramesh *et al.* 2001). Design rationale means a "record of reasons behind the decision taken, thus providing a kind of design/project memory and a common medium of communication among different people" (Louridas *et al.* 1996, 1). Design rationale "furnishes a way of capitalizing on past experience and thereby aiding design decisions" (ibid p. 1). There are several design rationale methods (e.g. IBIS (Conklin *et al.* 1988), REMAP (Ramesh *et al.* 1992), QOC (MacLean *et al.* 1991), PDR (Carroll *et al.* 1991), which enable the modeling of and reasoning from the knowledge on produced ISD deliverables, conducted ISD actions, stated ISD goals, and reasons for them (i.e. arguments and justifications). With this knowledge it is possible to trace reasons for the made decisions and actions, which is especially beneficial in requirements engineering (e.g. Pohl *et al.* 1997; Nguyen *et al.* 2003).

8.3.6 Summary

In this section we established the first part of the ISD ontology. We defined the essential concepts and relationships with which the structural, functional and behavioral aspects of the ISD contexts can be conceived, understood, structured and presented. In building the ontology we have derived its concepts and constructs from the underlying ontologies, especially from the context ontology. Moreover, we have searched for, selected, customized and integrated concepts and constructs from the ISD literature in those cases where they fitted our views and approaches. For each ISD domain, plenty of references and comparisons to the literature were given.

Due to the large extent of the domain area, we were forced to make some limitations in our considerations. From the seven ISD contextual domains, we focused only on the most essential ones that are the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain. For the other ISD domains, the concepts and constructs can be more or less directly derived from the corresponding contextual domains (Chapter 4). We only provided an overview of the ISD intra-domain relationships. For each intra-domain relationship it is easy to formulate a definition on the bases of those given in Section 4.5. Regardless of the aforementioned limitations, this part of the ISD ontology is quite comprehensive comprising dozens of ISD concepts and constructs. To assess the ISD ontology, we will compare it with some frameworks, meta models and the like presented in the ISD literature in Section 8.5.

8.4 ISD Perspectives

Having defined the concepts and constructs within and between the ISD domains in Section 8.3, it is now possible to introduce the second main part of the ISD ontology, namely the ISD perspectives. The ISD perspectives are important to managing the complexity of ISD and, for instance, understanding how conceptions about the ISD context gradually develop when engineering an ISD method. We focus here on four ISD perspectives: the ISD systelogical perspective, the ISD infological perspective, the ISD conceptual perspective, and the ISD datalogical perspective. After defining them we consider the ISD inter-perspective relationships.

8.4.1 ISD Systelogical Perspective

Based on the definitions in Chapter 6 we state that the *ISD systelogical perspective* reveals the support that ISD provides to its utilizing system (US_{ISD}). The utilization system is composed of the IS and the US_{IS}. Implied from the definition we can say that the following questions are relevant from the ISD systelogical perspective:

- What kind of IS is it for which the ISD project is launched?
- What kind of US_{IS} is it that the IS should support with information services?
- What kinds of services should the ISD provide to the US_{IS}?
- Derived from the answers to the above questions, what are the goals at and constraints for, approaches, organizations, actions, deliverables, etc. of the ISD context?

From the characterizations of the ISD perspective given above we can now derive the meta model of the ISD systelogical perspective in Figure 81. The ISD systelogical perspective concerns three contexts: the ISD context, the IS context and the US_{IS} context. The ISD context provides ISD services to the IS context, which in turn provides IS services to the US_{IS} context. An *ISD service* means all those material or immaterial ISD deliverables that are produced in the ISD context and delivered to be exploited in the intended IS context. From the systelogical perspective the ISD context is seen as a black box, meaning that only the ISD purpose domain, in addition to the aforementioned ISD services, is recognized in the perspective. The IS context, in turn, is considered through the concepts and constructs of the IS purpose domain, the IS action domain, and the IS object domain. The IS purpose domain is needed to reveal goals and reasons for which information processing is carried out. The other IS domains help characterize the kind of the IS and the circumstances for which the ISD context should provide IS services. Note that through the concepts of the IS purpose also other IS domains, yet on a more general level, are under the consideration.

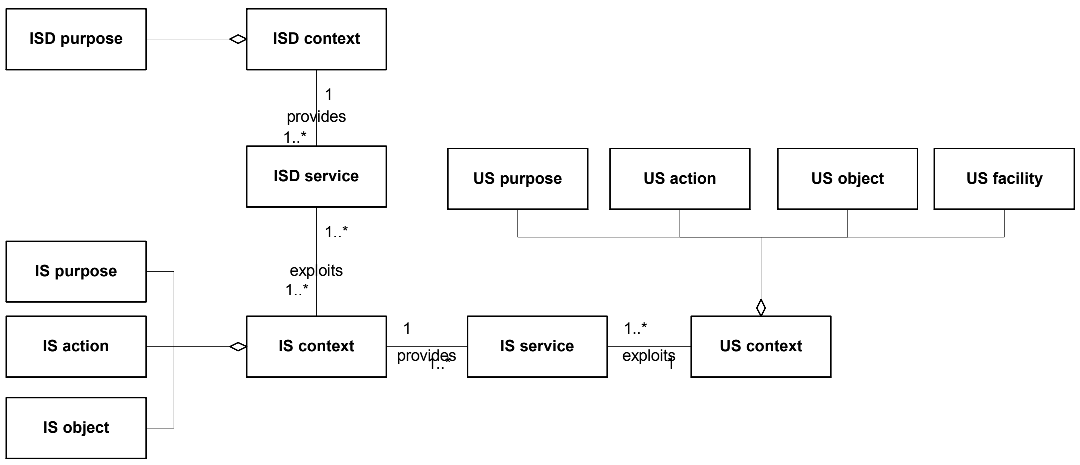


FIGURE 81 Meta model of the ISD systelogical perspective

It is the US_{IS} context, which ultimately benefits from that information processing that is hopefully improved by better ISD services provided by the ISD context. Therefore it is necessary to include the essential features of the US_{IS} context in the ISD systelogical perspective. The features relevant to the perspective are related to the US_{IS} purpose domain, the US_{IS} action domain, the US_{IS} object domain, and the US_{IS} facility domain. The last one is included to uncover a position which the new IS is to have in the US_{IS} context, that is to say, which US_{IS} actions are mainly to benefit from a new IS, or a new tool, in the US_{IS} context.

There are some implicit relationships between the domains of the three contexts that should be taken into account e.g. when engineering an ISD method. For instance, approaches and main principles selected for the ISD should suit the goals of and ways of working in those IS which are intended to exploit the ISD services. For instance, developing an information system for IS clients, who need and use expertise and knowledge of some specific area, makes it necessary to apply ISD approaches that give adequate emphasis on human beings, their needs and views (i.e. the user oriented approach), on one hand, and strongly involve IS clients into the ISD work (i.e. the participative approach, Mumford 1981), on the other hand.

8.4.2 ISD Infological Perspective

From the *ISD infological perspective* the ISD context is seen as a functional structure of information processing actions and informational objects. In this perspective, no attention is given to the features related to how the information objects are presented, neither to how they are implemented. Within this ISD perspective, the following questions are answered:

- What information is processed in the ISD context and why?
- What are the ISD actions, ISD rules, and input and output deliverables in the ISD context?

The relevant ISD domains from the ISD infological perspective are: the ISD purpose domain, the ISD action domain, and the ISD object domain. The concepts of the ISD purpose domain are used to specify conceptions about why ISD is needed, and what the goals of the ISD are. The concepts of the ISD action domain are used to express what is done in the ISD context, and the concepts of the ISD object domain pertain to ISD deliverables of the ISD context. In the following we consider more closely the concepts and constructs within the ISD infological perspective. The meta model of the perspective is presented in Figure 82. Note that all relevant sub-concepts and relationships are not depicted in the figure in order to save the space.

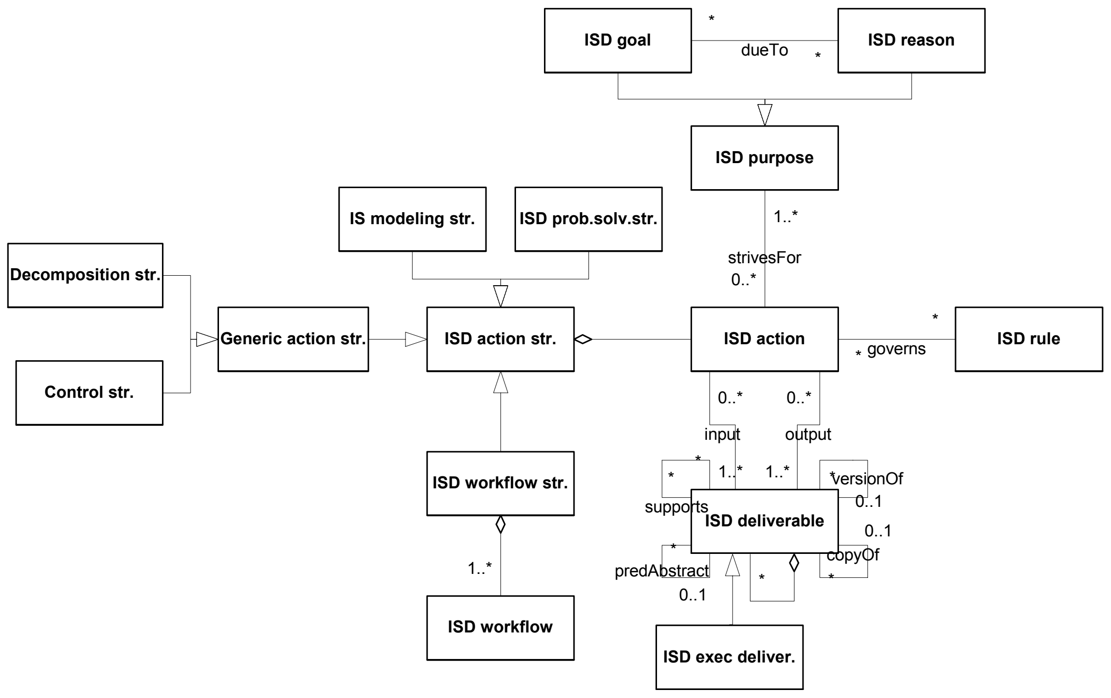


FIGURE 82 Meta model of the ISD infological perspective

The ISD reasons and the ISD goals (e.g. in problem matrices and goal/means graphs) provide answers to questions like: What are the problems, strengths, weaknesses, threats, and opportunities in the current IS and its environment? What are the requirements for a new IS, and which of the requirements are agreed on being goals for the ISD effort? The influence, refinement, dueTo, and causalTo relationships are specified to show how the ISD goals, the ISD requirements, and the ISD problems are related to one another.

Also the ISD actions are, on a general level, identified and organized according to the generic action structures (i.e. the decomposition and control structures), the ISD problem solving structure, the IS modeling structure, and the IS workflow structure. The rest of the ISD action structures (i.e. the ISD management–execution structure and the ISD phase structure) are applied later. ISD rules for ISD actions are specified on a general level.

The ISD deliverables from the ISD infological perspective only cover the ISD execution deliverables. They are further divided into categories according to the IS perspectives, resulting in specifications of ISD deliverables such as Requirements Specification (IS systelogical perspective), Function Specification in data flow diagrams (IS infological perspective), Database Schema in ER diagrams (IS conceptual perspective), Relational Schema (IS datalogical perspective), and Hardware Architecture (IS physical perspective). The partOf, versionOf, copyOf, supports and predAbstract relationships are recognized among the ISD deliverables.

8.4.3 ISD Conceptual Perspective

Applying the conceptual perspective generally means that the conceptual contents of the informational objects processed in a context are revealed. In the case of ISD context, the *ISD conceptual perspective* designates the things the ISD deliverables signify. Here we consider the ISD conceptual perspective only in relation to the ISD execution deliverables. As implied from Section 5.3, the object system of the ISD covers the IS, the OS_{IS} and the US_{IS}. Hence, the following questions are relevant:

- What are the meanings of the ISD deliverables?
- What do the ISD deliverables signify?
- What kinds of static and dynamic constraints are valid in the OS_{ISD}?

For the considerations from the conceptual perspective we have used the data model in this work. The IS data model (e.g. an ER schema) uncovers the conceptual contents of the information processed in the IS context and specifies the allowed conceptual structures. At the next higher layer, the ISD data model (e.g. the ER model) uncovers the conceptual contents of the information (i.e. ISD deliverables) processed in the ISD context. Because this information is already at the type level, the corresponding ISD data model contains meta models that are called the IS meta models. The meta models are always structural models (cf. Chapter 7). Because it is not possible here to describe all the meta models, we contend ourselves to illustrate the ISD conceptual perspective with the IS meta data model, which specifies the concepts and allowed conceptual constructs in the IS data models. Assuming that the IS data model is based on the ER model (Chen 1976) extended with concepts related to state transitions and constraints, the meta model resembles the one in Figure 83.

An *entity type* is a generic concept corresponding to the intensional specification of all those features that are shared by the entities that are regarded as instances of the entity type (Elmasri *et al.* 2000). An example of the entity type is Person. An OS relationship between two or more entities means any relevant connection, association or like between the entities. An *OS relationship type* is a generic concept corresponding to the intensional specification of all those features that are shared by the OS relationships that are conceived as instances of the OS relationship type. Each entity type that is connected by an

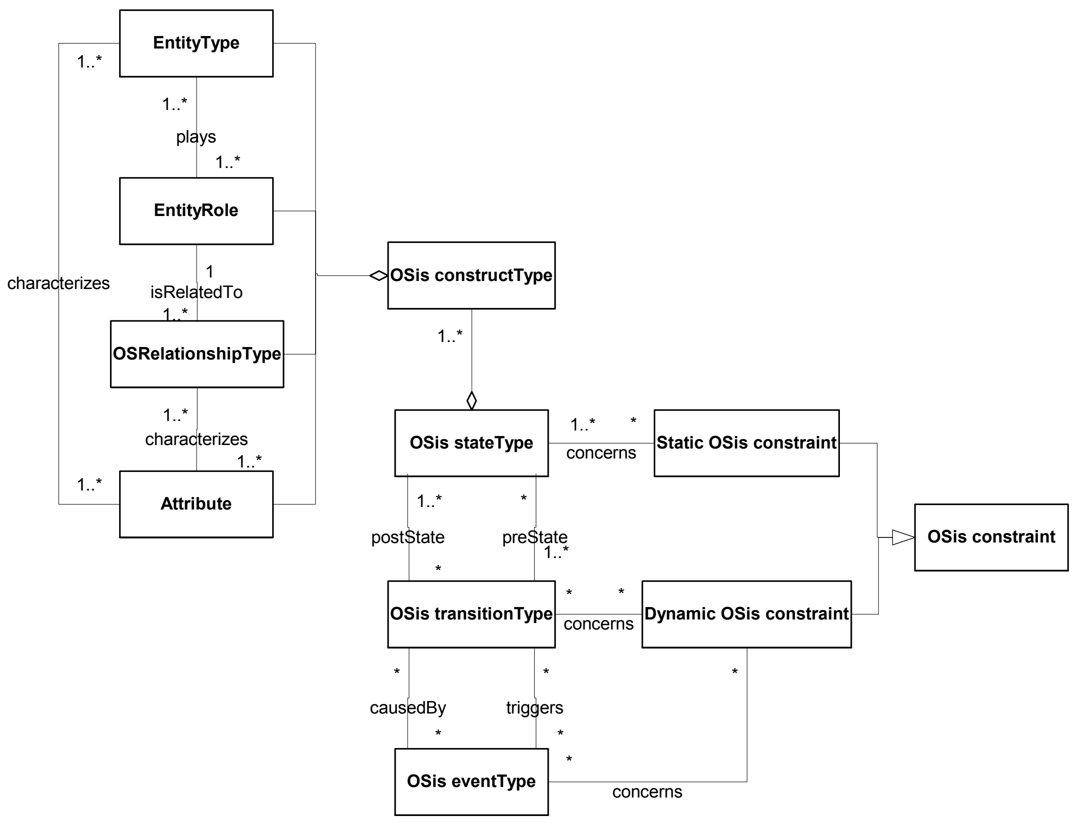


FIGURE 83 Meta model of the IS data model from the ISD conceptual perspective

OS relationship type plays a particular role in that relationship. We call this role the *entity role* to differentiate it from a role in the generic ontology (Chapter 3) and from an organizational role defined in the context ontology (Chapter 4). For example, Person may be in the entity role of Husband or Wife in the Marriage relationship type.

An *attribute* is a relevant predicate used to characterize an entity (e.g. Age) or an OS relationship (e.g. WeddingDay). A particular entity or OS relationship has one or more values for each of its attributes. A *single-valued attribute* has a single value for a particular entity or OS relationship, whereas *multi-valued attributes* may have many values. A *composite attribute* can be divided into smaller parts that still have independent meanings (e.g. Address is composed of StreetAddress, City, State, and Zip). An *atomic attribute* is not divisible. A value of a *derived attribute* can be calculated from the values of other attributes (e.g. Age from CurrentDate and BirthDate) or derived in some other way from the existing entities and/or OS relationships (e.g. NumberOfEmloyees).

An *OS$_{IS}$ construct type* in OS$_{IS}$ means here a conceptual construct composed of specific entity types related to one another through OS relationship types and characterized by attributes. For instance, Marriage [Husband: Person; Wife: Person] (WeddingDay: Value) is an example of an OS$_{IS}$ construct type. OS$_{IS}$ constructs defined in Section 6.3.3 are instances of OS$_{IS}$ construct types. An *OS$_{IS}$ state type* means a state type of the object system or its part, composed of OS$_{IS}$ construct types. An *OS$_{IS}$ transition type* is a generic concept corresponding to the specification of all those features that are shared by OS_{IS} transitions. An OS_{IS} state type may involve entity types, OS relationship types and/or attributes. An example of an *OS_{IS} transition type* is Divorce, which causes the change of the Marital status of Persons in the Marriage. The OS_{IS} transition types can be composed to establish OS_{IS} transition structures like those defined in the state transition ontology. An OS_{IS} event type means a generic concept corresponding to the specification of all those features that are shared by OS_{IS} events, which may trigger an OS_{IS} transition and which may be caused by another OS_{IS} transition.

To get a richer picture of the OS_{ISD}, it is necessary to know the rules governing the state types and the transition types in the OS_{IS}. Those rules are called the *OS_{IS} constraints.* An OS_{IS} constraint is static or dynamic. A *static OS_{IS} constraint* means a specification of allowed state or states. Static constraints may be population constraints, uniqueness constraints, referential constraints, cardinality constraints, attribute constraints, etc. A *dynamic OS_{IS} constraint* means a specification of allowed OS_{IS} transition(s) and/or allowed OS_{IS} events. A complex OS_{IS} constraint may comprise constraints of both of the types.

8.4.4 ISD Datalogical Perspective

From the *ISD datalogical perspective* the ISD context is considered through representation-specific concepts, involving, besides ISD purposes, ISD actions and ISD deliverables, on a general level ISD actors and ISD facilities as well. That means that ISD is seen as a context in which for some purposes data objects represented in some language are processed by actions of ISD actors with the support of some computer-aided tools. No reference is made to data carriers or other physical things of the ISD context. Since the number of the concepts needed to describe all the datalogical features of ISD is huge, we consider in the following only some of these essential concepts and constructs. In Section 6.3.4 we defined the IS datalogical perspective in terms of concepts and constructs concerning three parts of the IS: HIS, UI and CIS (cf. Figure 60). In the ISD context human information processing is in a much more dominating role, compared to the IS context, although ISD work is supported by computerized ISD tools. Therefore, we are here more interested in ISD as a context in which CASE environments, debuggers, and other technical things are only in a position of supporting tools. The meta model of the ISD datalogical perspective is presented from this viewpoint in Figure 84. All the details are not included in the meta model.

The ISD datalogical perspective elaborates conceptions about the ISD domains, resulted from applying the ISD infological perspective. Thus, conceptions of ISD problems, ISD requirements and ISD goals are now made more detailed and concrete. In the ISD action domain, the generic action structures, the ISD workflow structure, the ISD problem solving structure, and the IS modeling structures are refined from those considered within the ISD infological perspective. In addition, the ISD management–execution structure

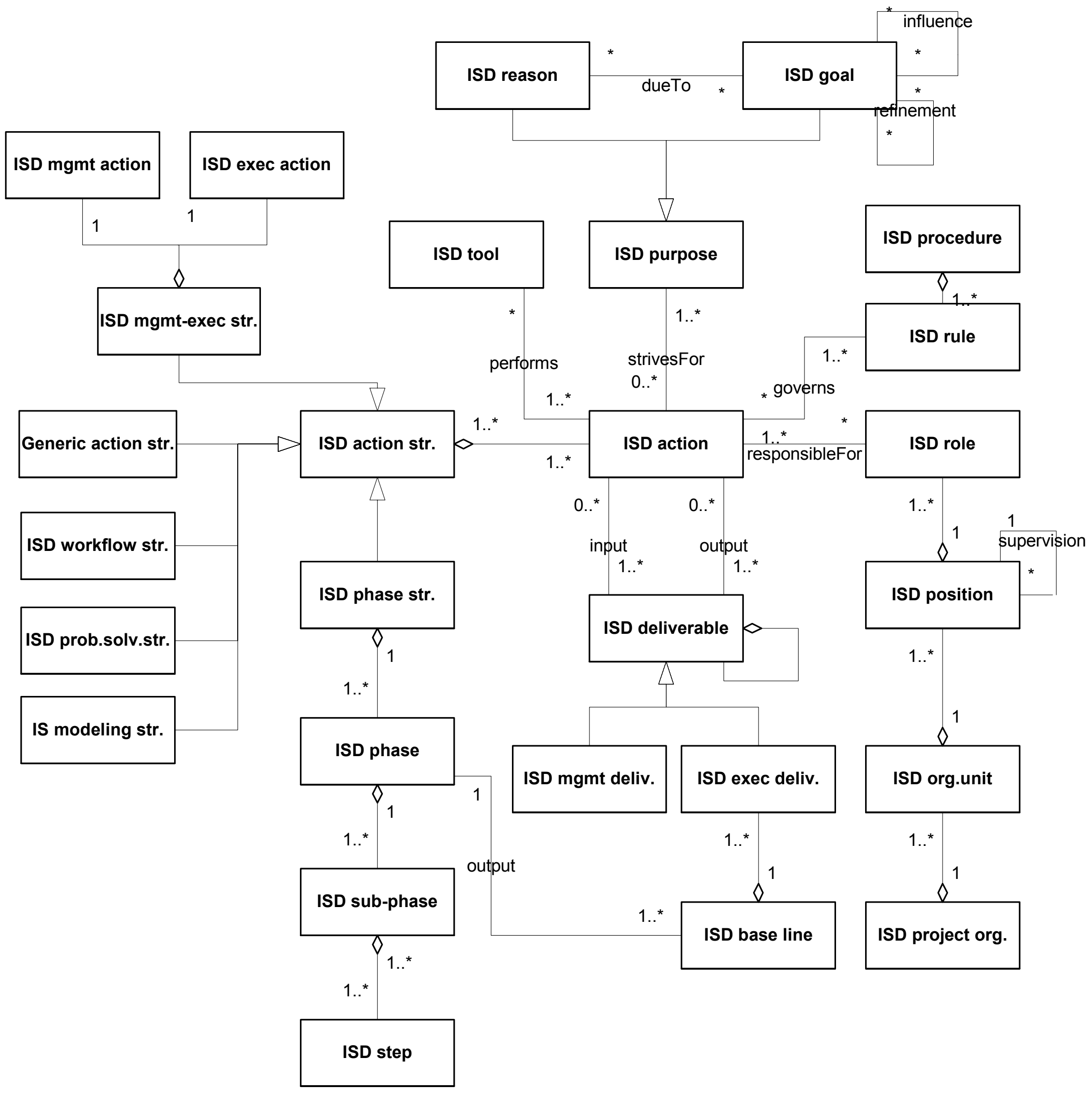


FIGURE 84 Meta model of the ISD datalogical perspective

and the ISD phase structure are taken into use. These structures are used to specify principles and ways of control and coordination of an ISD project. The ISD phase structure is composed of ISD phases, ISD sub-phases and ISD steps. The relationships between the phases and the sub-phases are grounded on the control structures, not on the temporal structures of ISD actions. For some ISD actions, general-level ISD rules are specified and aggregated to form ISD procedures.

Within the ISD object domain more refined decompositions and specializations of ISD deliverables are made. Also intra-domain relationships are detailed and refined. As a consequence of establishing the ISD management–execution structure and the ISD phase structure, the ISD management deliverables are distinguished and some of the ISD execution deliverables are collected to form baselines for ISD phases.

The ISD datalogical perspective also addresses the ISD actor domain and the ISD facility domain, yet on a general level. The ISD actions are composed through the responsibleFor relationships to constitute ISD roles. The ISD roles are further aggregated to form ISD positions, such as an IS analyzer, a database designer, and a programmer. The ISD roles with skill requirements may be parts of several ISD positions. The ISD positions are preliminarily related to one another through the supervision relationships. The ISD project organization with main organizational units is roughly established.

The basic concepts and constructs of the ISD facility domain are considered within the ISD datalogical perspective, though all physical aspects are still ignored. In Figure 84 the ISD facility domain is abstracted into the notion of an ISD tool, indicating that some of routine ISD actions can be allocated to be performed by a computerized information system (CIS). It might be possible to take a more tool-centered view exhibiting main logical components of the CIS and interaction with human information processing (HIS) through dialogs (see Section 6.3.4). We are forced to ignore this view here.

8.4.5 ISD Inter-Perspective Relationships

In this section we discuss how five ISD perspectives are related to one another. The discussion is based on the definitions given in Section 6.3.6.

Figure 85 illustrates the contents of, and the relationships between, the ISD perspectives in terms of concerned contexts and domains[142]. To distinguish between the contexts concerned, we depict them with bold line rectangles in the case there are more than one context involved by an ISD perspective. As stated above, the ISD systelogical perspective involves three kinds of contexts: the ISD context, the IS contexts, and the US_{IS} contexts. The corresponding ISD domains are shown in the figure. The ISD infological perspective concerns the ISD context only. The concepts and constructs of the ISD infological perspective are derived from the concepts and constructs established in the ISD purpose domain through the ISD systelogical perspective. The view got from the ISD infological perspective is further realized, first with the ISD datalogical perspective and then with the ISD physical perspective. The ISD physical perspective also instantiates concepts and constructs.

The ISD conceptual perspective designates, on a general level, things that ISD deliverables signify. ISD deliverables can signify things in the ISD context (e.g. time schedule for an ISD project)[143], in the IS context (e.g. a sequence diagram of invoicing), in the US_{IS} context (e.g. a business process diagram) and in the OS_{IS} context (e.g. an ER schema). About the things both structural features (i.e. states) and dynamic features (i.e. state transitions) are identified, except about the OS_{IS} about which meta-level information is only processed

142 To simplify the figure we do not present the abstractedFrom relationships between the informational objects and the ISD conceptual perspective (cf. Figure 63).

143 Note that here we consider also the OS_{IS} constructs that are signified by ISD management deliverables.

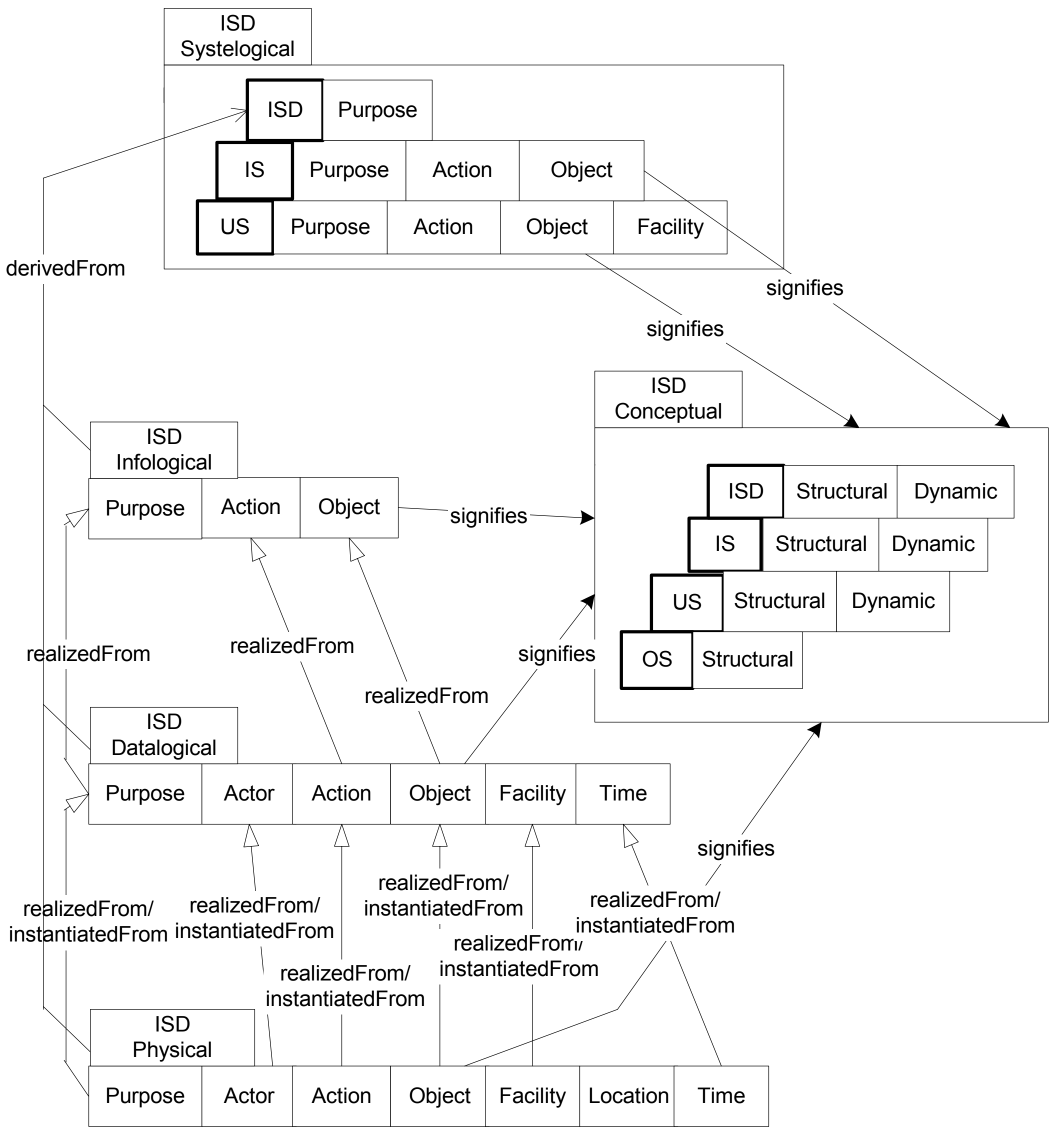


FIGURE 85 ISD inter-perspective relationships

and, as we know, meta information is always structural. ISD deliverables are recognized and conceptually elaborated within three perspectives. In each of these cases, the contents of the deliverables can be specified and analyzed through the conceptual foundation provided by the ISD conceptual perspective.

8.5 Comparative Analysis

In this section we present a comparative analysis of artifacts (i.e. frameworks, meta models or the like) that can be used as a means of understanding, structuring and presenting phenomena in information systems development. The analysis has been made for two reasons. First, we want to investigate what kinds of artifacts there are in the literature and how they compare with our ISD ontology. Second, we want to test how well the ISD ontology serves as an analytical framework. We will present the analysis in two parts. The first part (Section 8.5.2) aims to give an overview of the artifacts and their general properties. In the second part we make a deeper analysis of the concepts and constructs provided in the artifacts. This analysis reveals, among others, how comprehensive the artifacts are and what issues and domains they cover and emphasize. The analysis will be carried out for four ISD domains separately (Sections 8.5.3-8.5.6). The ISD domains are: ISD purpose domain, ISD actor domain, ISD action domain, and ISD object domain. The analysis ends with conclusions (Section 8.5.7). But before we report on the results of the analysis, we will make a short review of relevant works and select some of them for the analysis.

8.5.1 Relevant Work

In the ISD literature there is a plethora of frameworks, meta models, reference models, and ontologies that concern information system development. Here we refer to them with the common term 'artifact'. The artifacts can be categorized into two main groups: those which describe and structure ISD, and those which have been developed to describe, analyze, compare and/or engineer ISD methods. In the following we make a short review of artifacts in both of the groups and consider their relevancy to our comparative analysis.

Artifacts in the first group apply in most cases a rather general view of ISD; either they characterize ISD in terms of ISD paradigms (e.g. Hirschheim *et al.* 1989; Iivari *et al.* 1998a), of ISD approaches (e.g. Wood-Harper *et al.* 1982; Hirschheim *et al.* 1995; Iivari *et al.* 2001), or of processes (e.g. Boehm 1988; Iivari 1990a; Iivari 1990b; Sabherwal *et al.* 1995). Most artifacts in this group have too narrow a scope of and/or too general a view of ISD to be interesting for us. There are, however, same exceptions. Iivari (1990a, 1990b), for instance, presents the hierarchical spiral model, which provides an abstract explanatory model for IS/SW design process. The model contains strictly defined concepts and constructs in a large variety.

Another set of artifacts, included in the first group, consists of meta models or ontologies that have been mainly built for structuring specific phenomena of ISD. This set is very small. Ontologies such as the Frisco framework (Falkenberg *et al.* 1998) and the Bunge-Wand-Weber model (e.g. Wand 1988a; Wand *et al.* 1989; Wand *et al.* 1990a; Wand *et al.* 1990b) cover quite well the elementary phenomena in the UoD, as shown in Section 3.10, and provide some concepts for the IS layer as well. But they do not extend to the ISD layer. There are some ontologies that concern specific parts of software engineering (Kitchenham *et al.* 1999; Ruiz *et al.* 2004; Kishore *et al.* 2004). Kitchenham *et al.* (1999), for instance, suggest an ontology of software maintenance. The ontology covers maintenance activities, maintenance procedure, maintenance organization procedure, and so-called peopleware. Ruiz *et al.* (2004) build upon the work of Kitchenham *et al.* (1999) and suggest an ontology for the management of software projects. The ontology is composed of three main parts: maintenance ontology (products sub-ontology, process sub-ontology, activities sub-ontology, and agents sub-ontology), workflow ontology, and measurement ontology. These kinds of presentations are, however, too domain-specific to serve as a generic and comprehensive basis for the understanding of ISD.

Artifacts in the second group have been constructed for the analysis, comparison, and engineering of ISD methods. This group can be sub-divided in many ways. First, there is a very large array of literature providing artifacts for evaluation and comparison of methods. Artifacts are (a) idealized methods through which methods are to be assessed, (b) feature lists, (c) meta models, or (d) contingency frameworks (cf. Sol 1983, 4). In the early 1980's and ever since then, dozens of feature lists have been suggested for analysis and comparison of methods, or parts thereof (e.g. Rzevski 1983; Brodie 1983; Falkenberg *et al.* 1983; Bodart *et al.* 1983; Maddison *et al.* 1984; Ang 1993; Karam *et al.* 1993; Flynn *et al.* 1993; Kelly *et al.* 1992). More structured forms of artifacts are taxonomies of (e.g. Brandt 1983; Blum 1994), hierarchies of (e.g. Law 1988) and frameworks of (e.g. Iivari *et al.* 1983; Essink 1986; Iivari 1994; Jayaratna 1994) features. Although these artifacts are rooted upon with concepts referring to specific aspects of ISD, the concepts are not explicitly defined, nor are they properly structured (except Iivari (1994) that is based on Iivari (1989a, 1990a, 1990b). The same holds for contingency frameworks (e.g. Davis 1982; van Swede *et al.* 1993; van Slooten *et al.* 1993; Punter *et al.* 1996; Kettinger *et al.* 1997; Roberts *et al.* 1998; Lin *et al.* 1999). The use of meta models has brought necessary aid to the specification of ISD methods in a more precise and condensed fashion. ISD methods are specified in terms of notation and media, on one hand, and of conceptual contents, on the other hand. Here we are interested in the latter kinds of specifications because the conceptual contents of the ISD methods are defined with fundamental concepts of ISD contexts.

In conclusion, from a very large set of artifacts suggested for the description, analysis, comparison and engineering of the ISD methods in the ISD literature, we are here particularly interested in those which specify the conceptual contents of the ISD methods through (graphical) meta models. Applying these criteria, we select the following artifacts for the analysis: the framework and the reference model of Heym *et al.* (1992a), the process meta-model of the NATURE Approach (NATURE Team 1996; Grosz *et al.* 1997), the meta model of Saeki *et al.* (1993) and Saeki (1998), and the framework of Song *et al.* (1992) and Song (1997). In addition, we accept two more artifacts although they are not presented in a graphical form. These are the ontology of Harmsen (1997) and the hierarchical spiral model of Iivari (1990a, 1990b). Both of these artifacts are presented with preciseness that corresponds to the graphical form.

There are many other presentations (e.g. Olle *et al.* 1988a; Mi *et al.* 1996; Nuseibeh *et al.* 1996; Gupta *et al.* 2001) that are excluded from this analysis, due to their limited scope.

8.5.2 Overall Analysis

The purpose of this sub-section is to present an overall picture of the six artifacts selected. For each artifact we give a short description. In addition, we expose the purposes, theoretical bases, ISD approaches applied, representation forms, and acts of validation of the artifacts. ISD approaches are expressed in terms of ISD approaches defined in Section 8.1. Acts of validation mean efforts that have been made to show the applicability of the artifact. The results of the analysis are summarized in alphabetical order in Table 23. To ease the comparison of the artifacts to our framework, the ISD ontology of OntoFrame is presented on the first row in the table.

Harmsen (1997) proposes a framework, a language and a procedure to assemble a situational method from building blocks called method fragments. Method fragments are sub-divided into product fragments and process fragments. A product fragment describes a product of the IS engineering process. A process fragment describes the activities of that process. For defining the method fragments, an ontology, called the Methodology Data Model (MDM), and a process classification system are proposed. The ontology consists of basic concepts of ISD products. The process classification system consists of definitions of basic steps, product types and state types. To characterize the method fragments, a large set of property types, including e.g. fragment aspects and scenario aspects, are defined. In this section we mainly consider the concepts and constructs contained in the ontology. They are defined in English and, to a large extent, also with the first order predicate calculus.

The work of Harmsen (1997) has been built on an extensive analysis of existing ISD literature (ibid p. 16-17) and on the systems theoretical view of an information system. No specific ISD approach can be recognized. Some experience from the applicability of the artifact has been gained from the implementation of a prototype of the Method Base System and from its use in some empirical studies (van Slooten 1995; van Slooten *et al.* 1996). Heuristics for the selection of method fragments has been applied in the Situational Project Definition project (Klooster 1996).

Heym *et al.* (1992a) present a framework and a reference model for describing, understanding, and comparing ISD methods. The framework categorizes the aspects of ISD methods into three perspectives: application type, life cycle, and model focus. In the reference model, the methodology knowledge is decomposed and structured with five meta models. In each meta model the essential concepts and relationships of ISD and ISD methods are depicted in a graphical notation. In addition, the concepts are defined in English. No theoretical basis is mentioned. The reference model is based on "experience from description and comparison of different software engineering methods" (ibid p. 234). The reference model forms the data model of the MERET tool, which has been used in several organizations to describe, understand and compare ISD methods.

TABLE 23 Overview of the artifacts

Reference	Artifact	Purpose	Theoretical basis	ISD approach	Representation	Validation
This work	ISD ontology	For understanding the ISD domain, and for analyzing and constructing other artifacts for the ISD domain	Theories underlying the contextual approach, ISD theories, ISD methods	Contextual approach	Definitions in English, meta models in a graphical notation	Used as a basis in an comparative analysis and in the construction of the methodical skeleton of ME
Harmsen (1997)	Ontology (MDM) and a process classification system	For assembling a situational method from building blocks called method fragments	Literature on existing ISD methods	Not recognizable	Definitions in English, supported with the use of first order predicate calculus	Implemented as a prototype of the Method Base System that has been used in some empirical studies
Heym *et al.* (1992a)	Framework, reference model	For describing, understanding, and comparing ISD methods	Not mentioned	Not recognizable	Definitions in English, the reference model in a graphical notation	Used as the data model of the MERET tool which has been deployed in several organizations
Iivari (1990a, 1990b)	Hierarchical spiral model	For providing an abstract explanatory model for the IS/SW design process	Socio-cybernetics, Information economics	Transformation, learning process, and decision-oriented approaches	Definitions in English	Not mentioned

(continues)

TABLE 23 (continues)

Reference	Artifact	Purpose	Theoretical basis	ISD approach	Representation	Validation
NATURE Team (1996), Grosz *et al.* (1997)	Process meta-model	For defining way-of-working in requirements engineering	Theory of plans	Decision-oriented approach	Definitions in English, partly in a graphical notation	Widely used in succeeding projects; Prototype
Saeki *et al.* (1993), Saeki (1998)	Meta model	For representing software specification and design methods	Not mentioned	Transformation approach	Definitions in English, partly in a graphical notation	Used to develop representations of ISD methods; Prototype
Song *et al.* (1992), Song (1997)	Framework	For identifying method components (Song *et al.* (1992); For integrating software methods (Song 1997)	Abstracted from existing methods	Transformation approach	Definitions in English, partly in a graphical notation (Song 1997)	Not mentioned

Iivari (1990a, 1990b) presents the hierarchical spiral model for information system and software development, based on the sociocybernetic metamodel and information economics (Iivari 1983). The socio-cybernetic metamodel is an abstract explanatory model for the IS/SW design process. It emphasizes the design process as an iterative inquiry process that supports decisions about the IS/SW product and the IS/SW design process. Moreover, it supports the transformation view on the design process. The hierarchical spiral model has been built upon the conceptual framework for IS/SW product (Iivari *et al.* 1987; Iivari 1989a). Due to the large scope of the model, many concepts are not explicitly defined. Nothing is mentioned about the use of the model in practice, neither about its validation.

The NATURE project (NATURE Team 1996; Grosz *et al.* 1997) developed the process meta-model based on the NATURE (Novel Approaches to Theories Underlying Requirements Engineering) approach. The purpose of the approach is to address the problem of providing guidance and system support in poorly understood, largely human-driven creative processes. It applies the decision-oriented approach to requirements engineering. Also the notion of a context is essential to the approach. A context is "a meaningful association of a situation and an intention with guidance advice" (NATURE Team 1996, 527). Requirements engineering is modeled as a set of related contexts in which a decision is made on how to process product parts and in which order. The approach is partly based on the theory of plans (Wilesky 1983). The approach has been widely used in research projects. Also a prototype to support the use of the process meta-model has been implemented.

Saeki *et al.* (1993) present a meta model for representing software specification and design methods. The purpose of the meta model is to cover so-called atomic concepts that are common to all the methods. To model a particular method, this core part is to be enhanced by introducing new concepts in a method-specific part. The meta model is presented in two parts: the product part and the procedural part. The product part applies a conceptual view on the software design products. In the procedural part the software design process is seen as transformations by procedures that are connected to one another by the input/output relationships. The meta model has been used to develop formal presentations of several ISD methods. Also a prototype of the method base system has been implemented. Later, Saeki (1998) uses the meta model to demonstrate method integration.

Song *et al.* (1992) present the so-called base framework for the identification of method components that are comparable in different methods. The framework has been developed through an iterative process of abstracting from existing methods and applying the framework to model methods. It is composed of two parts: the type framework and the function framework. The type framework aggregates the components' internal characteristics. The function framework aggregates the design issues that components address. The model of design life-cycle clearly indicates the application of the transformation view. Partly based on the framework in Song *et al.* (1992), Song (1997) presents a framework for the integration of software methods. The framework contains a method composition model that distinguishes between the high-level components and the low-level components. The composition model is based on the analysis of eight structured methods and object-oriented methods. Both of these frameworks (Song *et al.* 1992; Song 1997) contain parts that are related to method components. We ignore those aspects here and return to consider them in Section 9.7.6.

We can draw the following conclusions from the above overall analysis. First, from the six artifacts only two have been built on explicated theoretical bases. The hierarchical spiral model (Iivari 1990b) has been rooted on socio-cybernetics, and the process meta-model by NATURE Team (1996) has been partly based on the theory of plans. Other artifacts have been derived mainly analyzing the existing literature of ISD methods. For some artifacts no grounds are mentioned. We have built our ISD ontology by following the contextual approach, which has been established on several underlying theories, including semiotics, pragmatics, theories of human and social action, and systems theory (cf. Chapter 4). In addition, we have reviewed a large collection of ISD literature on ISD theories and ISD methods, which has influenced our ISD ontology.

Second, from the ISD approaches defined in Section 8.1.2 the transformation approach is most commonly applied. It is intrinsic to the meta model of Saeki *et al.* (1993) and Saeki (1998) as well as to the framework of Song *et al.* (1992) and Song (1997). In addition, it is included in the hierarchical spiral model of Iivari (1990b), which also reflects the view of the learning process approach. The NATURE approach has been, in an interesting fashion, built upon the decision-oriented approach applying the notion of a context. The conception of a context is, however, very limited, as compared to the corresponding notion in OntoFrame. The contextual approach in our study is aimed to enable conceiving, structuring and representing phenomena of ISD from multiple viewpoints. This aim has been realized by including in the ISD ontology concepts and constructs that are applicable as such, or they can be specialized, if necessary, to meet views of more special approaches.

Most of the artifacts have been "validated" by using them in the intended purposes (e.g. for describing and/or integrating methods) and/or as a basis for prototypes of computer-aided method engineering systems (CAMES). The ISD ontology has been validated by using it as a basis of comparative analyses and in the construction of the methodical skeleton for method engineering.

Next, we analyze the artifacts in more detail in order to find out what concepts and relationships they provide for the four ISD domains, i.e., the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain.

8.5.3 ISD Purpose Domain

The ISD purpose domain means all those concepts and constructs through which we can conceive, understand, structure and represent problems, requirements, goals, etc. of ISD, or parts thereof. From the six analyzed artifacts only two provide concepts and relationships that belong to the ISD purpose domain. The artifacts are the ontology of Harmsen (1997) and the meta process model of the NATURE approach (NATURE Team 1996; Grosz *et al.* 1997). The concepts and relationships of these two artifacts and of our ISD ontology are presented in Table 24. From the relationships we have included in Table 24 the intra-domain relationships and those inter-domain relationships that relate some concept of the ISD purpose domain and concepts of some of the three ISD domains.

As seen in Table 24, Harmsen (1997) defines a large set of concepts for the ISD purpose domain. A goal stands for the purpose towards which a system is directed (i.e. IS goal). Also the other concepts are associated to an IS. Besides those included in our ontology, the ontology provides the notions of a benefit and a critical success factor. It specifies explicit relationships between most of the sub-concepts, while in our ISD ontology the relationships between a goal and other concepts are organized through the concept of a reason. It is noteworthy that Harmsen (1997) defines differently the relationship between a goal and a requirement; i.e. a goal imposes a requirement.

Harmsen (1997) associates the concepts of the ISD purpose domain to concepts of other domains with several relationships. 'Solution' means a final outcome of the ISD (i.e. an information system). With the base relationship he connects 'solution' to 'strength' and to 'weakness'. He also uses the term 'system', and relates it to 'problem' with the choice relationship. To reveal an informant, the MDM defines the expression relationship between 'group' and 'problem', and between 'group' and 'goal'. 'Effect' is an influence of one concept instance upon another.

In the process meta-model of the NATURE Approach (NATURE Team 1996; Grosz *et al.* 1997) the notion of an intention is a part of a decision made in the requirements engineering process. An intention expresses what a requirements engineer wants to achieve[144]. It is thus a kind of goal. An intention can be global or local. This is all that is included in the meta model from the ISD purpose domain.

Our ISD ontology provides one generic concept, namely ISD purpose, that is specialized in several ways: into goals of two kinds (hard and soft), into reasons of six kinds (requirements, problems, opportunities, threats, strengths and weaknesses), as well as into IS purposes. The IS purpose can be of any of the aforementioned kinds. Furthermore, the IS purposes are classified according to five IS perspectives. We argue that with our ISD ontology the issues of ISD contexts related to purposes, goals, objectives, intentions, desires, problems, etc.

144 In contrast to how the notion of an intention is defined in the approach, an intention is not associated to a requirements engineer in the graphical representation of the process meta-model.

TABLE 24 Summary of the concepts and relationships of the ISD purpose domain

Reference/ Concepts	**ISD ontology**	**Harmsen (1997)**	**NATURE Team (1996)**
Generic concept	ISD purpose		Intention
Sub-concepts	ISD goal ISD reason ISD requirement ISD problem Opportunity Threat Strength Weakness	Goal Requirement Problem Benefit Opportunity Threat Weakness Strength Critical success factor	Global intention Local intention
Intra-domain relationships	partOf (ISD purpose, ISD purpose) dueTo (ISD goal, ISD reason) influence (ISD goal, ISD goal) refinement(ISD goals, ISD goal) influence (ISD requirement, ISD requirement) refinement (ISD requirement, ISD requirement) causalTo (ISD problem, ISD problem)	imposition (Goal, Requirement) dependence (Goal, Critical success factor) base (Goal, Opportunity) base(Goal, Threat)	
Inter-domain relationships	motivatedBy (ISD human actor, ISD purpose) expressedBy (ISD actor, ISD purpose) strivesFor (ISD action, ISD purpose) intendedFor (ISD deliverable, ISD purpose)	base (Solution, Strength) base (Solution, Weakness) expression (Group, Problem) expression (Group, Goal) alternative (Solution, Problem) choice (System, Problem) effect (System, Benefit)	

can be addressed in a much more comprehensive and detailed fashion than in the other artifacts. In addition, the conceptual structure in the ISD ontology has been built to be inherently flexible to ease adaptations in the ontology. The ISD ontology also provides a large collection of relationships with which the concepts are related to the other concepts in the ISD purpose domain (i.e. intra-domain relationships) as well as to the concepts in three other ISD domains.

8.5.4 ISD Actor Domain

The ISD actor domain means all those concepts and constructs that are used to conceive, understand, structure and present human or social phenomena in the ISD context. Those phenomena involve persons and groups, or administrative units. From the analyzed artifacts only two provide concepts and constructs that belong to this ISD domain. They are the ontology of Harmsen (1997) and the reference model of Heym *et al.* (1992a). In addition, Iivari (1990a, 1990b) mentions users of the information produced by an IS/SW act as the usage characteristics of the IS/SW act but he does not discuss them further. The concepts and relationships of the two artifacts and of the ISD ontology are presented in Table 25.

Harmsen (1997) uses the term 'actor' in two contexts, in relation to an information system and to IS engineering. In the latter context an actor is defined to be "a function involved in an IS engineering project" (ibid p. 57). This means that the notion is much more "action-specific" than our notion, thus corresponding somewhat to our notion of an ISD role. An actor role is defined to be "the type of function an actor has with respect to manipulating and receiving product fragment instances" (ibid p. 57). Possible roles are reviewing, creating, and analyzing. The supervision relationship reflects the functional hierarchy of actors. The responsibility relationship indicates which actor is responsible for which method fragment. The execution relationship is used to specify which actor is executing which process, in which role. Correspondingly, the skill relationship reflects the capability of an actor to execute a process fragment in a certain role. The destination relationship indicates to which actor and in which role a product fragment is addressed.

Heym *et al.* (1992a) define an actor to be "any person, group of persons, or organizational unit which is involved in an activity or is responsible for a process or a milestone" (ibid p. 230). An actor may consist of other actors (e.g. an organizational unit consists of persons). An actor is related to an activity with the 'is involved in' relationship, and to a process and a milestone with the 'is responsible for' relationships.

Comparing to our ISD ontology, we can clearly see that the ontology of Harmsen (1997) only superficially addresses the ISD actor domain. Heym *et al.* (1992a) provide one generic concept and three sub-concepts. Our ontology contains nine sub-concepts, embracing human beings in different constellations, as well as in administrative units. In our view, these both sides are important to understanding ISD as a social and organizational endeavor. In addition, we specialize the persons according to the kind of their expertness and ISD roles

TABLE 25 Summary of the concepts and relationships of the ISD actor domain

Reference/ Concepts	**ISD ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**
Generic concept	ISD actor	Actor	Actor
Sub-concepts	ISD human actor ISD position ISD role ISD organizational unit ISD project organization Project team Steering committee Person Group	Actor role	Person Group Organizational unit

Intra-domain relationships	occupiedBy(ISD human actor, ISD position)
	memberOf (Person, Group)
	plays (Person, ISD role)
	supervision(ISD position, ISD position)
	controls(Steering committee, Project team)
		supervision(Actor, Actor)	consists_of

Inter-domain relationships	carriedOut(ISD action, ISD actor)
	responsibleFor(ISD role, ISD action)
	ownedBy(ISD deliverable, ISD actor)
	viewedBy(ISD deliverable, ISD actor)
		responsibility(Actor, Method fragment)
		execution(Actor, Actor role, Process fragment)
		skill(Actor, Actor role, Process fragment)
		destination(Actor, Actor role, Product fragment)
			is involved in (Person, Activity)
			is responsible for (Actor, Process)
			is responsible for (Actor, Milestone)

into six ISD specific roles. Resulting from the large number of concepts, the ISD ontology also provides a variety of inter-domain and intra-domain relationships that is much larger than in the other two artifacts.

8.5.5 ISD Action Domain

The ISD action domain contains all those concepts and constructs that are used to conceive, understand, structure and present how ISD functions. Due to its centrality to the ISD context, it is natural that all the analyzed artifacts provide large sets of concepts and relationships for it. To ease the analysis and comparison, we categorize the action-related concepts in the artifacts according to the ISD action structures (i.e. the ISD management-execution structure, the ISD workflow structure, the ISD phase structure, the ISD problem solving structure, the IS modeling structure, the control structure, and the abstraction structure) of our ISD ontology. The summary of the analysis and comparison of eight artifacts, including our ISD ontology, is presented in Table 26.

Harmsen (1997) uses the notion of a process fragment to mean any activity to be carried out within a method. A process role represents the respect in which a process fragment manipulates a product fragment. Examples of roles are ‘production’, ‘update’ and ‘usage’. There are three intra-domain relationships in the ISD domain. A contents relationship means that a process fragment consists of another process fragment. A precedence relationship reflects the fact that a process fragment precedes another process fragment. A choice relationship between the process fragments means that one process fragment conditionally precedes the other process fragment. Hence, the two last relationships establish control structures between the ISD actions. In addition, there are two inter-domain relationships. A prerequisite relationship expresses the fact that a process fragment instance requires a product fragment instance for its execution. A manipulation relationship indicates that a process fragment manipulates a product fragment in a certain role (e.g. updates). Harmsen (1997) has also constructed a process classification system. The system is based on the notion of a basic action meaning “a class of actions in IS engineering in which each action have the same effect” (ibid p. 137). According to the classification system, actions are sub-divided into six types: planning, analysis, synthesis, evaluation, implementing, and evolution. These types belong to our ISD workflow structure. For each type of actions he also provides a set of sub-actions. These types of actions are not, however, integrated into the ontology.

Heym *et al.* (1992a) define a process to “be either a phase or an activity” (ibid p. 228). A phase consists of phases, or of activities. An activity is an elementary process that cannot be divided into sub-activities. Activities are classified into six types: decision, planning and control, abstraction, checking, review activities, and form conversion. As seen from the above list, the classification is quite heterogeneous: decision, planning and control belong to the ISD management–execution structure, while abstraction, checking, review and form conversion are parts of the IS modeling structure. A process can be an iteration process or a non-iteration process. An iteration activity or phase is executed more than once in a sequence. In the framework of Heym *et al.* (1992a) the life cycle is divided into seven stages (Olle *et al.* 1988a): information system planning, analysis, design, construction design, construction, test and installation, and maintenance. It should, however, be noticed that this classification is not included in the reference model presented in meta models.

Some relationships between the processes are described with the notion of a dependency. A process dependency can hold between activities or phases. A dependency can exist also between a milestone and a phase, and between a milestone and an activity. A dependency has one of four different semantics: sequence, refinement jump, branching path, and unifying path. Hence, it is a conceptual mechanism for establishing control structures between the ISD actions (in our terminology). A process is related to deliverables through input

TABLE 26 Summary of the concepts and relationships of the ISD action domain

References/ Concepts	**ISD Ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**	**Iivari (1990b)**	**NATURE Team (1996)**	**Saeki *et al.* (1993)**	**Song *et al.* (1992)**	**Song (1997)**
Generic concept	ISD action	Process fragment	Process	Design act	Action	Procedure	Action	Design activity
Sub-concepts	ISD workflow ISD phase ISD process	Process role Basic action	Phase Activity	Main phase Subphase Design act			Step	Action Process Step
ISD manage-ment -execut-ion structure	Mgmt action: - ISD planning - ISD organiz-ing - ISD staffing - ISD directing - ISD control-ling Execution action		Decision Planning Control		Plan Execution			
ISD workflow structure	ISD worflow: - IS req's engineering - IS analysis - IS design - IS implement-ation - IS evaluation	Action type: - Planning - Analysis - Synthesis - Evaluation - Implementation - Evolution	ISD stage: - Analysis - Design - Construction design - Construction - Test and installation - Maintenance					

(continues)

TABLE 26 (continues)

References /Concepts	**ISD Ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**	**Iivari (1990b)**	**NATURE Team (1996)**	**Saeki *et al.* (1993)**	**Song *et al.* (1992)**	**Song (1997)**
ISD phase structure	ISD phase: - Inception - Elaboration - Construction - Transition		Phase	Phase structure: - Org. design - Conceptual / infological design - Datalogical/ technical design - Implement-ation				
ISD problem solving structure	Intelligence Design Choice				Decision Choice			
IS modeling structure	Conceptual-izing Representing Creating Refining Testing Transforming Translating Integrating Relating		Abstraction Checking Review Form conversion	Diagnosis/ Design Verification / Validation, Observation / Analysis, Manipulation / Refinement	Transformation		Create Modify	Create Modify Use Evaluate

(continues)

TABLE 26 (continues)

References / Concepts	**ISD Ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**	**Iivari (1990b)**	**NATURE Team (1996)**	**Saeki *et al.* (1993)**	**Song *et al.* (1992)**	**Song (1997)**
Control structures	sequence selection iteration	precedence (Process fragment, Process fragment) choice(Process fragment, Process fragment)	sequence refinement jump branching path unifying path iteration	iteration	precedence alternative	precede		
Abstraction structures	partOf isA memberOf instanceOf	contents	aggregation	component of		has	part of is a	instance of contains

Inter-domain relation-ships	input (ISD deliverable, ISD action)
	output (ISD action, ISD deliverable)
	involves(ISD action, OS$_{ISD}$ construct)
		prerequisite (Process fragment, Product fragment)
		manipulation (Process fragment, Product fragment, Process role)
			input usage (Deliverable, Process)
			output usage (Process, Deliverable)
					change (Action, Product)	input (Product part, Procedure)
						output (Process, Product part)
							input (Artifact, Action)
							output (Action, Artifact)
							affect(Artifact, Action)
							influence(Represent-ation, Action)	

and output usage relationships. A type of the usage can be essential or referential. For instance, an essential input usage is an input deliverable whose contents are primarily used or converted into an output deliverable.

The hierarchical spiral model of Iivari (1990b) expresses dynamics of a design process in three forms: evolution dynamics, main-phase dynamics, and learning dynamics. The first one concerns the relationships between the design products (cf. increments, versions, and releases). That is not considered here. The main-phase dynamics is established on the basis of the levels of modeling for an IS/SW product. Iivari (1990b) distinguishes between three main phases: the organizational design phase, the conceptual /infological design phase, and the datalogical/technical design phase. These main phases stand for our ISD phases. The learning dynamics imply that each main phase consists of successive subphases after which the IS/SW design process is replanned. The IS/SW design process is conceptualized in terms of IS/SW acts. The acts are categorized in two ways: (a) diagnosis/design and verification/validation, and (b) observation/analysis and manipulation/ refinement. These categories are considered in the ISD ontology to be parts of the IS modeling structures. Although Iivari (1990b) does not propose any ISD workflow structure in the sense of the one in the ISD ontology, he presents a categorization of diagnosis, design, and verification and validation activities on three modeling levels (ibid p. 455)

In the meta process model of the NATURE Approach (NATURE Team 1996; Grosz *et al.* 1997), an action means something which is performed to change a product of the process. A decision is a choice between sub-contexts during the requirements engineering. A decision is composed of an intention and an approach[145]. Based on the kind of actions performed in a context, the contexts are classified into executable contexts, choice contexts, and plan contexts. An executable context leads directly to the execution of an action. At the most detailed level, the execution can be seen as a set of transformations performed on the product. A choice context is used when there is more than one alternative to select as regards the direction to proceed. A plan context is a mechanism by which a context viewed as a complex issue can be decomposed into a number of sub-issues. This stands for a planning action in our terminology. The meta process model provides several relationships, which we here consider intra-domain relationships of the ISD action domain. The precedence link holds between contexts, thus concerning actions as well. Alternative contexts imply that there are alternative ways of working, selected by choice criteria. There is also one inter-domain relationship: an action 'changes' a product.

In the meta model of Saeki *et al.* (1993) the procedural part consists of an procedure concept (not defined in the article) and four relationships. The has relationship indicates "what sub-procedures belong to a procedure and also represents their hierarchy" (ibid p. 153). The precede relationship shows the

145 The composition of an intention and an approach is modeled as an "objectified relationship".

execution order of the procedures. The input and output relationships indicate which constituents in the product part are inputs to and outputs from the procedure, respectively.

In the type framework by Song *et al.* (1992) an action means one or more physical and/or mental processing steps used in design. An action may create or modify a design artifact. An action can be a part of another action, and a specialization of another action. Some artifacts are inputs to actions and outputs from actions. There are also the relationships called 'affect' and 'influence' between the artifacts and the actions, and between the representations and the actions, respectively, but they are not defined.

Song (1997) distinguishes between two domain areas: software design and design method. The former stands for design practice, while the latter means the components in a design method. Software design is presented with two concepts, a design activity and an artifact, meaning that a design activity makes, uses, or evaluates artifacts. Actions and artifact models as the components in a design method are regarded as classes of activities and artifacts, respectively. Components are sub-divided into high-level components and low-level components. On the high level, processes are "sets of steps in which designers use particular artifact models in developing software" (ibid p. 109). On the low-level, processes are decomposed into actions, either physical (such as typing) or mental (such as decision making). An action may create, modify, use or evaluate model components.

To summarize from the analysis above, Heym *et al.* (1992a) and Iivari (1990b) define most concepts for the ISD action domain among the analyzed artifacts. Heym *et al.* (1992a) provide concepts for the IS modeling structure, through the reference model, and for the ISD workflow structure through the framework. Iivari (1990b) defines concepts for the ISD phase structure and the IS modeling structure. Harmsen's (1997) process classification system contains concepts belonging to the ISD workflow structure but they are not integrated with the MDM ontology. The other artifacts provide only few single concepts, mostly for the IS modeling structure. From the intra-domain relationships the artifacts cover best, yet not completely, the control structures and the abstraction structures. From the inter-domain relationships the most commonly specified ones are the input and output relationships.

In our ISD ontology, the ISD action domain is seen as a complex construct of various ISD action structures. The action structures are categorized into three levels: the generic action structures, the contextual action structures, and the ISD-specific action structures. The first category contains the action decomposition structure, the control structures and the temporal structures. The second category comprises the ISD management-execution structure and the ISD problem solving structure. These action structures have been defined in the context ontology (Chapter 4), and they are just specialized for the ISD context. The third category is composed of those ISD action structures that have been defined particularly for the ISD contexts. These are the ISD phase structure, the ISD workflow structure and the IS modeling structure. We argue that this array of ISD action structures is comprehensive enough to address the most common action structures faced with in practical ISD and ISD methods. Further, we claim that the array of the ISD action structures at three levels is flexible enough to enable the definition of more specific structures if needed by some particular ISD approach or view.

8.5.6 ISD Object Domain

The ISD object domain contains all those concepts and constructs that refer to things to which the ISD actions are targeted. From the analyzed artifacts, only four consider the ISD object domain properly. They are: Harmsen (1997), Heym *et al.* (1992a), Iivari (1990a, 1990b), and Song *et al.* (1992). In contrast, the meta model of Saeki *et al.* (1993) and Saeki (1998) considers the ISD deliverables from the conceptual perspectives only. Song (1997) focuses on method components and the only notion in the ISD object domain he recognizes is an artifact. In the meta process model of the NATURE Approach (NATURE Team 1996; Grosz *et al.* 1997) outcomes of actions are called product parts, but they are not specialized into sub-concepts, neither associated with one another. The summary of the analysis of the four artifacts, together with the ISD ontology, is presented in Table 27.

Harmsen (1997) defines a product fragment to mean "a specification of a product delivered and/or required within a method" (ibid p. 52). He uses the result type classification based on Euromethod's Deliverable Model (Franckson 1994) to classify the result types into project domain deliverables, target domain deliverables, and delivery plans. Project domain deliverables are partitioned into plans and reports. Target domain deliverables are sub-divided into requirements statements, specifications, and operational items. The division corresponds, to some degree, to our IS perspectives. The conceptual contents, representation and abstraction levels (logical, technical) of the product fragment are not contained in the ontology, but expressed as product property types of method fragments (i.e. root, representation, and abstraction level). That is why they are not included in the table. The contents relationship between the product fragments means that one product fragment consists of another product fragment. The precedence relationship expresses ordering with respect to different product versions.

Heym *et al.* (1992a) define a deliverable to be any result from a process or an activity (e.g. document, CASE tool graphic, program code). The deliverables are classified into four disjoint types: planning deliverables, decisions, system specifications, and reports or documentation. The first two can be regarded as ISD management deliverables in our ontology, while the others are ISD execution deliverables. A deliverable may consist of other deliverables. A deliverable as a representational object is related to a conceptual construct via the covers relationship. Between the deliverables there are deliverable flows, meaning that some deliverables are used inputs to another deliverables (corresponding to our supports relationship).

TABLE 27 Summary of the concepts and relationships of the ISD object domain

References/ Concepts	**ISD ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**	**Iivari (1990a, 1990b)**	**Song *et al.* (1992)**
Generic construct	ISD deliverable	Product fragment	Deliverable	IS/SW product	Artifact
Sub-concepts	Formal Semi-formal Informal Baseline IS model IS implementation	Result type			
Management - execution dichotomy	ISD mgmt deliverable ISD exec deliverable	Project domain deliverable Target domain deliverable Delivery plan	Planning Decision System Specification Report or documentation		
IS perspectives	IS systelogical model IS infological model IS conceptual model IS datalogical model IS physical model IS inter-perspective model	Requirements statements Specifications Operational items		Organizational Conceptual/ infological Datalogical/ technical	Problem domain Problem-model domain Solution-model domain Design-document domain

(continues)

TABLE 27 (continues)

References/ Concepts	**ISD ontology**	**Harmsen (1997)**	**Heym *et al.* (1992a)**	**Iivari (1990a, 1990b)**	**Song *et al.* (1992)**
IS domains	IS purpose model IS actor model IS action model IS deliverable model IS data model IS facility model IS location model IS time model IS inter-domain model				
Abstraction relationships	partOf isA instanceOf	contents	consists of generalization		is-part_of is_a
Intra-domain relationships	supports versionOf copyOf predAbstract signifies (ISD deliverable, OS_{ISD} construct)	precedence	deliverable flow covers (Deliverable, Conceptual model object)		

In his hierarchical spiral model Iivari (1990b) refers to the ISD deliverables as the IS/SW products. The products are categorized according to the levels of modeling into products of the organizational level, the conceptual/infological level, and the datalogical/technical level. The levels correspond to our IS perspectives.

Song *et al.* (1992) define an artifact to mean "a description of some sort of entity involved in a design process" (ibid p. 46). An artifact can be a part of another artifact. The artifacts can be specialized, for instance, according to the domains, which are: problem domain, problem-model domain, solution-model domain, and design-document domain.

Interestingly, in only four artifacts among those selected for the analysis, the ISD object domain is addressed at least in a reasonable fashion. This is a surprise because the ISD deliverables are undoubtedly important to making sense of what the ISD produces in each of its parts and phases. Also in those four there are deficiencies. The ontology of Harmsen (1997) is most extensive addressing the ISD management deliverables and the execution deliverables, as well as giving a simple categorization of deliverables that is loosely based on the IS perspectives. The reference model of Heym *et al.* (1992a) and the hierarchical spiral model of Iivari (1990a, 1990b) provide specializations of ISD deliverables in a way which corresponds to our IS perspectives. The four artifacts provide most of the abstraction relationships between the ISD deliverables but only a few intra-domain relationships.

Our ISD ontology applies three major criteria in the specialization of ISD deliverables: the dichotomy of management – execution, the IS perspectives, and the ISD domains. This has yielded a comprehensive set of concepts and constructs for the ISD object domain. We have not wanted to introduce any specific deliverable concepts, such as Analysis report or Systems specification, in order to keep the ISD ontology general enough. More specific concepts can be easily specialized from those defined in the ontology. We have, however, seen it important to define intra-domain relationships, such as supports, versionOf, copyOf, and signifies, which inherently reflect ISD practice, and which therefore should be included in every ISD artifact.

8.5.7 Conclusions from the Analysis

The comparative analysis of the existing ISD artifacts was done for two reasons, first to investigate what kinds of artifacts there exist in the literature and how they compare with our ISD ontology, and second, to test how well the ISD ontology serves as an analytical framework. Conclusions in this section aim to answer to both of these questions.

We selected six artifacts (i.e. models, frameworks, and meta models) for the analysis with the purposes of (a) producing an overview of them and (b) finding out how comprehensive they are and how they are focused in terms of ISD domains. The ISD domains serve as a suitable basis for the analysis because they have been derived from the contextual approach. Most of the artifacts have been developed for the description, analysis, comparison and/or engineering of

ISD/SW methods. Hence, it was reasonable to expect that they would also reflect the essential aspects of ISD contexts. The analysis was made in two parts.

In the overall analysis we found that only two of the artifacts (i.e. Iivari 1990b; NATURE Team 1996) are established on some theoretical grounds. Most of the artifacts have been abstracted from existing ISD methods. This situation is unsatisfactory for two reasons. First, only with a sound theoretical background we can be sure that phenomena of ISD become properly conceived, understood and structured. Second, abstracting from existing methods in a way replicates properties of the methods and does not help recognize phenomena of ISD not addressed by the methods. Our ISD ontology is based on the contextual approach built on several underlying theories. Those theories help guarantee that the view applied in building the ISD ontology is broad and multifaceted enough.

Concluding from the in-depth analysis based on the four ISD domains, we can make several statements. First, all of the analyzed artifacts put an emphasis on the ISD action domain. Second, most of the artifacts address, at least to some degree, the ISD object domain. In contrast, there are just few artifacts that provide concepts and constructs for the ISD purpose domain (i.e. Harmsen 1997; NATURE Team 1996) or the ISD actor domain (i.e. Harmsen 1997; Heym *et al.* 1992a). This is surprising when taking into account that the selected artifacts have been constructed for describing, analyzing, and comparing the ISD methods in particular. How to analyze methods if some of the essential features are not addressed in an artifact? Our ISD ontology has been established on seven ISD domains enabling the perception of ISD as a context with a large set of contextual features. We have also built our ISD ontology to be structured and flexible, meaning that its adaptation and specialization are easy to accomplish.

From the six artifacts, the ontology of Harmsen (1997) appeared to be the most comprehensive. It has, however, some shortcomings in its coverage of the ISD actor domain (e.g. lack of ISD role, ISD position, ISD organizational unit) and the ISD action domain (e.g. lack of the management–execution structure, the ISD phase structure, and the IS modeling structure). The artifact can also be criticized for its incoherence and unstructuredness. The MDM ontology contains concepts and constructs that correspond to our IS ontology and partly to the ISD ontology, making no separation between these parts. Harmsen (1997) presents - as separate from the MDM ontology - the process classification system, which comprises concepts that clearly belong to the ISD ontology (cf. contents, representation and abstraction level of the product fragments). Although our ISD ontology is larger than the ontology of Harmsen (1997), it is more clearly structured, first according to the processing layers, second according to the contextual domains, and third, within each domain, according to the strictly defined generic structures. That makes it easier to understand and use.

Next in terms of comprehensiveness comes the reference model of Heym *et al.* (1992a). It lacks the concepts of the ISD purpose domain but provides a basic set of concepts and relationships in all three other ISD domains. The variety of concepts and relationships supported by it is particularly large in the ISD action domain. The meta models of Saeki *et al.* (1993) and Saeki (1998) and the frameworks of Song *et al.* (1992) and Song (1997) were found to be insufficient in all the ISD domains although they are aimed to provide a comprehensive basis for the description, analysis and comparison of ISD methods. They do not address the ISD purpose domain, nor the ISD actor domain. Also the other ISD domains are inadequately covered.

The artifacts of Iivari (1990a, 1990b) and the NATURE Team (1996) cannot be evaluated with the same measures and scales as the others, because they are different by their nature. The hierarchical spiral model of Iivari (1990b) is actually a process model, although addressing processes-in-broad. The process meta-model of the NATURE Team (1996), although presented in the form of a meta model of an ISD method, also addresses ISD process. Both of these artifacts provide a large variety of concepts and relationships for the ISD action domain, as expected. The hierarchical spiral model of Iivari (1990a, 1990b) applies diversified structures (cf. the transformation approach, learning process approach, and decision-oriented approach) to view ISD as being composed of intertwined ISD actions and ISD deliverables. In contrast, the process meta-model of the NATURE approach (NATURE Team 1996; Grosz *et al.* 1997) applies the decision-oriented approach according to which ISD actions are strongly related to the ISD purposes (cf. intentions).

To summarize the experience got from the use of the ISD ontology as an analytical framework in the comparative analysis, we can state the following. First, the ISD ontology clearly provided a comprehensive basis for understanding, structuring and comparing the existing artifacts. Theories underlying it increase our confidence that most of the essential aspects of the ISD context are included in the ISD ontology. This justifies its use as a "yardstick" in the analysis. Second, the ISD ontology appeared to be easy to apply, mostly due to its naturalness and structured form. For instance, the artifacts propose miscellaneous sets of concepts and constructs for the ISD actions domain. The seven ISD action structures at three levels defined in the ISD ontology considerably helped us in interpreting, classifying, and comparing the artifacts at the level of concepts. Although these conceptions are, to some degree, subjective we believe that we have managed to demonstrate and justify the applicability of the ISD ontology with the analysis above.

8.6 Summary and Discussions

The goal of this chapter was to present the ISD ontology. We approached this goal first considering the ISD paradigms and the ISD approaches. We applied the classification of Hirschheim *et al.* (1989) and Hirschheim *et al.* (1992a) to characterize four basic ISD paradigms. Next, we sub-divided the ISD approaches into three categories, defined the most essential ISD approaches in each category and discussed relationships between the ISD approaches. The discussion of the ISD paradigms and ISD approaches concretely showed how wide a variety among the conceptions about ISD really is. With the aim to encompass basic views of the ISD paradigms and ISD approaches, we formulated a generic but comprehensive definition of ISD, according to which ISD is seen as a context with aspects from seven contextual domains.

Rooted on the definition of ISD we engineered the ISD ontology. The ISD ontology is composed of two main parts: the ISD domains and the ISD perspectives. For the first main part, we defined the concepts and relationships in four ISD domains and supported the presentations with meta models. In addition, we brought out plenty of references to the literature where similar or differing conceptions are suggested about the discussed issues. For the second main part we defined four ISD perspectives (systelogical, infological, conceptual, and datalogical). The inter-perspective relationships were also specified. The ISD perspectives are important to managing the conceptual complexity of ISD and to understanding how conceptions about the ISD context develop stepwise when engineering an ISD method.

Finally, we made a comparative analysis of six artifacts (i.e. models, meta models, frameworks) selected from the literature for two reasons. First, we wanted to investigate which kinds of artifacts there exist in the literature and how they compare with our ISD ontology in terms of comprehensiveness and coverage. Second, our intention was to test how well the ISD ontology suited as an analytical framework. The analysis was made in two parts. In conclusion, most of the artifacts turned out to be totally lacking of theoretical basis and to be insufficient in coverage of essential aspects of ISD. The ISD ontology appeared to be the only artifact built firmly on sound theoretical bases. Only some of the artifacts provide any concepts for the ISD purpose domain and the ISD actor domain. While the ISD action domain is, at lest to some degree, addressed in the artifacts, there are large gaps in the coverage of the ISD object domain. The ISD ontology was shown to be the most comprehensive, covering four ISD domains with a large number of concepts and relationships, organized into flexible and easy-to-adapt structures.

9 ISD METHOD ONTOLOGY

The ISD method is a highly complex and multi-faceted notion of which there are multiple views and conceptions in the literature. It is regarded as a discipline, an approach, a body of skill, a procedure, a way of accomplishing something, etc. The ISD method is also said to be a collection or system of various ingredients, such as concepts, models, techniques, rules, procedures, tools, beliefs, and values. The purpose of this chapter is to clarify and elaborate conceptions of the ISD method. This will be done specifying the ISD method ontology. The ISD method ontology aims to provide concepts and constructs for conceiving, understanding, structuring and representing contextual aspects of ISD methods. The ontology is composed of several parts. One of those parts is the conceptual content of the ISD method, which provides the "glasses" through which we can conceive phenomena in ISD. That part, known as the ISD ontology, was presented in Chapter 8. The other parts view the ISD method as a representational and physical thing, consisting of different components and having certain basic assumptions, intentions and historical backgrounds. The method ontology has been derived from the underlying contextual ontologies (see Figure 86).

This chapter is organized as follows. We start with considering why the ISD methods are actually needed and used in practice, by reviewing empirical studies that report on benefits from the use of the ISD methods. Second, we discuss the difference between the terms 'methodology' and 'method'. Third, we delineate the concept of an ISD method as a 'carrier' of ISD knowledge and specify basic classifications of the ISD methods. Fourth, we define seven fundamental views through which the ISD method can be perceived, and present an integrative definition of the ISD method that highlights the essential features of the method from all these viewpoints. Fifth, we establish the ISD ontology composed of parts corresponding to the seven views. Sixth, we apply the ISD method ontology to consider, from a broader perspective, a range of methodical artifacts. We also discuss criteria for acknowledging a methodical artifact to be an ISD method. Seventh, we make a comparative analysis of

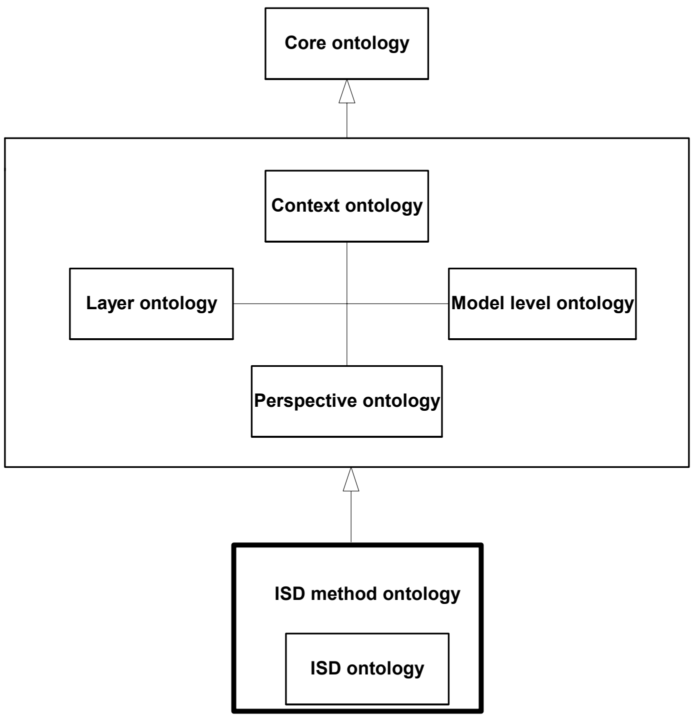


FIGURE 86 Basis and structure of the ISD method ontology

frameworks and categorizations of ISD methods proposed in the literature. Eighth, we consider the notion of a method component in more detail. We define the notion, present a classification scheme and specify a contextual interface of a method component. We also discuss the integration of method components, illustrate the discussion with examples, and make a comparative analysis of conceptions of a method component in the literature. The chapter ends with a summary.

9.1 Why to Use an ISD Method?

The first software systems were scientific applications in which algorithms and technical problems were the most critical issues. Later "administrative" information systems, more oriented towards basic operational functions of organizations, were built. At that time the only conceivable design task was programming and specifying computer room operations (Somogyi *et al.* 1987). To accomplish these tasks systems developers often followed a variety of systematic practices. New practices were invented as needed, and those, which seemed to work in previous projects, were subsequently mobilized again. They formed the developer's 'rules-of-thumb' and, in a sense, his/her 'method' (Episkopou 1987). They were passed on to other developers, often by word of mouth. These practices were typically not codified and sometimes not even written down (Hirschheim *et al.* 1995).

This pre-method area ended roughly in the mid-1960s (Hirschheim *et al.* 1995), when a need for more comprehensive and concise work instructions became urgent. This was boosted by several trends: information systems became more complicated, a number and variety of ISD endeavors increased, and requirements on the productivity of projects and the quality of outcomes became more demanding (Fitzgerald 1997). This led first to the emergence of life-cycle methods and structured approaches with process-based description models, procedural techniques and strict phase structures. Later, new approaches and methods were developed to address more comprehensively the whole ISD life cycle, to cover a larger variety of application areas, to acknowledge the importance of human and social aspects to ISD, and to cope with more advanced technologies (Hirschheim *et al.* 1995; Avison *et al.* 1995a; Truex *et al.* 2000; Avison *et al.* 2003).

As a result from the proliferation of different applications and expansion of ICT technology into novel areas, thousands, or even tens of thousands of ISD methods have been developed during the last four decades. But are they really needed? Why to use these ISD methods anyway? What are the real benefits from using them in practical ISD? To these questions we try to shed some light by making a short review of empirical research on experience of method use in practice. In this section we focus only on the reported benefits. We are fully aware that there are also a lot of evidence about negative impacts of the method use in ISD work, either due to defects in the ISD methods or from reasons that are more related to human and organizational issues. We return to reported problems in and negative impacts of the ISD methods in Chapter 10.

Based on the review of empirical studies on the use of ISD methods, we sub-divide reported benefits into five groups. Group (a) facilitates the acquisition, accumulation, use and dissemination of ISD knowledge, group (b) helps the management of ISD projects, group (c) reduces the need of money, labour and time in the ISD process, group (d) improves the quality of ISD deliverables, and group (e) provides a political support to ISD. Next, we discuss the benefits in this order.

Schönström *et al.* (2003) conclude from two case studies that ISD methods play an important role as knowledge enabler in large software projects. These methods stimulate individual knowledge development and facilitate the sharing of individual knowledge and its transformation to organizational knowledge. The methods create a common platform for communication and understanding by defining a communicative framework consisting of a common terminology, workflows and best practices. Also Rahim *et al.* (1998), Middleton (1999) and Hardy *et al.* (1995) noticed better communication and increased user involvement in the ISD as a consequence of the method use. The developers benefit from the methods in different ways. Fitzgerald (1996a) concludes that the relationship between the developer experience and the method use resembles a U-shaped curve, which shows that the more necessary and explicit support the method provides for novices, and the more experienced the developers become, the more 'invisible' their method use turns out to be. Hidding (1997) call this the process of 'internalization' through which the method use becomes subconscious. The ISD knowledge becomes re-considered when a new method is implemented in an organization. According to Backlund *et al.* (2003) the public knowledge contained in a commercial method, such as RUP (Rational Unified Process (Kruchten 2000)), is made organization-specific through the customization process. By this process the generic knowledge, conveyed by the method, is assimilated with the existing organizational knowledge.

The use of ISD methods has been noticed to help the planning and control of ISD projects. Fitzgerald (1998a), for instance, found in his field study that methods facilitate project control and increase visibility into the development process. Chatzoglou (1997) reports that the method use resulted in fewer iterations in requirements capture and analysis. The study of Hidding (1997) indicated that the methods were promoted by management in order to attain more sophisticated or better project control. And the larger the projects, the more important it becomes to control them.

The method use has also been found to reduce the need for money, labor and time in the ISD process. Concluding from a case study, Jones *et al.* (1988) report on significant reduction in estimated project budgets, below-budget project completions, and significant productivity improvements in maintenance. Chatzoglou (1997) reports that on an average not using an ISD method results in the involvement of more people, a need for more time and effort and higher costs compared to when a method is used. The survey (Rahim *et al.* 1998) conducted within Bruneian organizations indicates improved productivity resulting from the method use. Also Hardy *et al.* (1995) found that in some cases methods improved productivity and enabled better timescale estimates.

The method use may also improve the quality of ISD deliverables. Chatzoglou (1997) found out that managers were more confident about the quality of requirements gathered when a method was used. Also Rahim *et al.* (1998) report on better fulfillment of user requirements. In addition they found that the adoption of a method produced quality documents. The survey of Hardy *et al.* (1995) indicates that the method use means fewer errors in design, specifications match the requirements and the system matches the specification. There are also some studies on formal methods indicating positive effects on deliverables. According to Pfleeger *et al.* (1997) formal methods, combined with other techniques, yielded highly reliable code. Snook *et al.* (2001) report on a higher level of reliability, reduction in post-delivery failures, higher efficiency of the product code, and benefits in aiding traceability between the specification and the code.

While the aforementioned findings are related to so-called rational roles behind the use of methods, there is also a set of political roles that ISD methods can play in ISD (Keen 1981; Newman *et al.* 1990; Chang *et al.* 2002; Fitzgerald *et al.* 2002). Chang *et al.* (2002) sorted out 192 examples of political games from 56 cases and categorized them into 41 kinds of games. Middleton (1999) concludes that the developers generally found a method a source of political protection should things go wrong. Using a method they could show that they had tried to adhere to the approved procedures for ISD (cf. 'social defence' (Wastell 1996)). The developers also felt it increased their professionalism (cf. Fitzgerald *et al.* 2002). According to Nandhakumar *et al.* (1999) methods can be valuable in ISD projects as a necessary fiction to present an image of control or to provide a symbolic status.

In addition to the findings in the empirical studies reviewed above, there are a lot of issues in which the method use is believed to contribute to the ISD contexts. The methods are said, for instance, to allow skill specialization and division of labor (Fitzgerald 1998a), to further the standardization of the development process (Avison *et al.* 1995a; Pijl *et al.* 1997; Roberts *et al.* 1998), and to help in the selection of more suitable techniques (Fitzgerald 1998a).

In conclusion, numerous methods have been developed for ISD during the past few decades. Despite various problems in methods and their usage, ISD methods have appeared to be beneficial in many ways. They are considered to be artifacts that convey the best practices on ISD for helping achieve better outcomes through a more efficient, effective, and manageable ISD process. But what is, actually, this artifact? What kind of knowledge does the ISD method help us accumulate, convey and share? What types of methods are there? What kinds of parts is the ISD method composed of? What is it that is required from these component parts to make them suitable for integration into a proper method. These questions will be addressed in the next sections where we will define the concepts and constructs of the ISD ontology.

9.2 Methodology vs. Method

In the ISD literature the terms 'methodology' and 'method' are used in various meanings. Sometimes they are seen to be separate, sometimes they are used interchangeably.

Originally, 'methodology' is a Greek term meaning the study of methods. Webster´s Dictionary (Webster 1989) defines the concept as "the study of the principles underlying the organization of the various sciences and the conduct of scientific inquiry". In accordance with this, Oliga (1988) defines a methodology "as a method of methods that examines systematically and logically the aptness of all research tools, varying from basic assumptions to special research techniques" (ibid p. 90). Stamper (1988) also reserves the term 'methodology' for a comparative and critical study of methods in general. Checkland (1981) states that "a methodology will lack the precision of a technique but will be a firmer guide to action than a philosophy". Heym *et al.* (1992a, 215) use the term 'methodology' to refer to a class of methods, e.g. the description or representation of different methods. Brinkkemper (1996) argues that "a methodology is the systematic description, explanation and evaluation of all aspects of methodical information systems development" (ibid p. 276).

On the other hand, 'methodology' is used to refer to a more concrete prescription, implying that the term is regarded as a synonym for 'method' (e.g. Oliga 1988; Jayaratna 1994; Avison *et al.* 1995a; Hirschheim *et al.* 1995; Russo *et al.* 1996; Hidding 1997; Iivari *et al.* 1998a). For instance, Hidding (1997) states that a methodology "represents a body of skills and knowledge that becomes an organization's standards, based on a common language among practitioners" (ibid p. 105). Iivari *et al.* (1998a) define the methodology as "an organized collection of concepts, methods, beliefs, values and normative principles supported by material resources" (ibid p. 165). There are also those who propose that 'method' is more comprehensive, and includes 'methodology' (Davis 1982; Hackathorn *et al.* 1988).

Some differences in the literature can be explained by different naming conventions in Europe and North America. In Europe 'method' is preferred to prescribe a more systematic way of conducting ISD, and 'methodology' is reserved for the use with the original meaning. In North America, 'methodology' is commonly used as Europeans use the term 'method' (cf. Iivari *et al.* 2001, 207).

In this study, we prefer the European naming convention. Thus, 'method' means a prescription for some specific actions in ISD, and 'methodology' is reserved for a comparative and critical study of methods in general.

9.3 ISD Method as a 'Carrier of ISD Knowledge'

The ISD method is commonly considered to contain collective knowledge and experience that are made 'visible' to enable their exploitation and advancement in succeeding ISD projects (Tolvanen 1998; Fitzgerald *et al.* 2002; Schönström *et al.* 2003; Backlund *et al.* 2003). We also start our work of specifying the notion of the ISD method by considering it as a 'carrier of ISD knowledge'.

To enable the understanding of, and the comparison between, various conceptions of the ISD method and to make our view clear, we construct a general framework that helps us highlight the essential aspects of the method. The framework is presented in the form of a cube that depicts the body of ISD knowledge needed in a particular ISD context (Figure 87). The framework is composed of three dimensions: the contents of ISD knowledge, the representation of ISD knowledge, and the type of the ISD method. Next, we discuss these dimensions in more detail.

The contents of ISD knowledge can be subdivided in many ways. Extending the categorizations by Freeman (1987) and Iivari *et al.* (2001, 206) we state that the body of ISD knowledge is composed of four components: knowledge of ISD process, knowledge of application domain, knowledge of IC

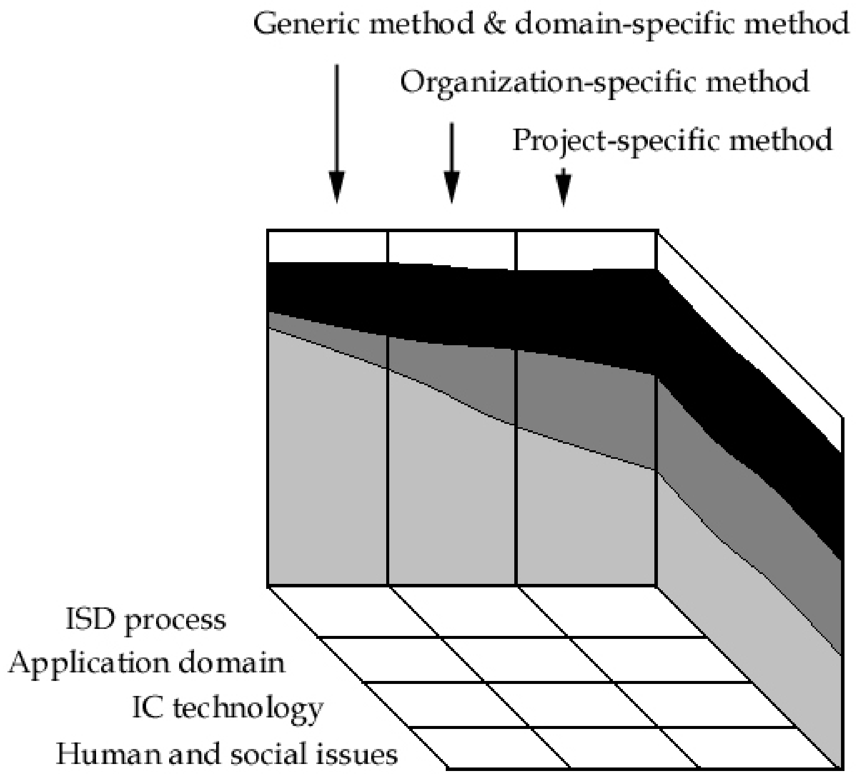


FIGURE 87 Framework for classifying ISD methods

technology, and knowledge of human and social issues[146]. The *knowledge of ISD process* means all the knowledge that concerns how to accomplish an ISD work. Examples of the questions it aims to answer are: What are the approaches and main principles to be applied in the ISD context? What are the ISD actions to be carried out and by whom? What are the deliverables that the ISD actions should produce? How ISD actions should be decomposed, related and managed? This body of knowledge also contains information about how to improve the ISD process. The *knowledge of application domain* means all the knowledge that concerns the information system to be designed, its utilization system and its object system. Each application domain has specificities of its own that are necessary to know for accomplishing the ISD. The *knowledge of IC technology* means all the knowledge that concerns the search, acquirement, installation, and deployment of hardware and software for the IS, as well as for the ISD. The *knowledge of human and social issues,* also known as 'humanware' and 'orgware', means all the knowledge that concerns human characteristics and behavior as well as social and organizational aspects that should be taken

[146] Iivari *et al.* (2004, 319) propose five knowledge areas in the information systems body of knowledge: technical knowledge, application domain knowledge, organizational knowledge, IS application knowledge, and ISD process knowledge. These are included in our components of ISD knowledge.

into account in building the IS and in organizing the ISD work. This body of knowledge is a counterbalance to the knowledge of IC technology. On a general level, this means the knowledge related to humans and organizations in general, and on an instance level, it is related to particular persons, teams and organizations.

The second dimension of our framework is the representation of the ISD knowledge. Next, we first discuss, based on the works of Schön (1983), Nonaka *et al.* (1995), Mathiassen et al. (1988), and Mathiassen (1998), to what extent it is possible at all to present the ISD knowledge. After that we distinguish between three different presentation forms of the ISD knowledge.

Schön (1983) differentiates between two perspectives of looking at professional practice: technical rationality and reflection-in-action. From the technical rationality viewpoint, professional practice is seen as instrumental problem solving. In this view, situations can be scientifically categorized, and professional practice applies scientifically–based theories and techniques. In contrast, from the perspective of reflection-in-action, situations of professional practice are unique, complex, uncertain and even conflicting. Nonaka *et al.* (1995) recognize two types of knowledge: explicit knowledge and tacit knowledge. Explicit knowledge can be articulated in a natural or formal language, which enables the transfer of the knowledge through documents. Tacit knowledge has to do with personal knowledge that is embedded in personal experience. It is not easy to explicate, not to speak of formalizing.

Based on the categorizations of Schön (1983) and Nonaka *et al.* (1995) we state that the ISD knowledge can take different forms. Part of it remains local, individual, and even tacit. Other parts of the ISD knowledge can be explicated and made publicly available. The ISD method is a means to capture and convey that knowledge. The ISD method provides explicit knowledge from technical rationality viewpoint in the form of principles, procedures, guidelines, etc. But because knowledge and action are intrinsically related (Mathiassen *et al.* 1988), the method should not restrict intuitive and ad hoc -like actions necessary in the ISD practice but rather encourage the use of tacit knowledge and creative thinking. Ciborra (1999) calls this intuitive part of ISD "work improvisation".

There are varying conceptions on a degree to which the ISD method as a "knowledge base" can convey and provide knowledge to ISD developers (Wastell 1996; Wordsworth 1999; Truex *et al.* 2000). Most optimistic think that all the ISD knowledge can be thought to reside outside the ISD practice – in books and in learned institutions. This view follows the scientific reductionism that is, to some degree, the underlying paradigm behind many ISD methods (Baskerville *et al.* 1992). The belief to the pervasiveness of the method is sometimes so firm that it is believed it is mainly the method that does the work (Wastell 1996). This, of course, is not the case.

But what is the relation between the knowledge, conveyed by the ISD method, and the knowledge used and accumulated in a particular ISD context? Fitzgerald *et al.* (2002, 13) calls the ISD practice using a method the method-in-action. Derived from the distinction made by Argyris *et al.* (1974) between an 'espoused theory' and a 'theory-in-use', we can say that knowledge within the method is never used exactly as originally intended. Different developers do not interpret and apply the same method in the same way. And the same developer applies the same method in different ways in different ISD contexts (Fitzgerald *et al.* 2002). This deviation of method-in-action from the method is made visible in two ways in Figure 87. That part of the ISD knowledge within the method that is totally ignored in a particular ISD context is represented with the white area in the upper part of the figure. The part of the ISD knowledge within the method that is customized for a particular ISD context and during it is represented with the grey area in the figure. Both of these areas are always present in an actual ISD context. Their sizes just vary. If these areas become very large, it is a reason to question whether the method is applied at all in the context. The other two areas in Figure 87 are: the ISD knowledge conveyed by the ISD method and used more or less as such in a particular ISD context (presented with the black area), and the ISD knowledge totally residing outside the ISD method (presented with the lowest area). The latter stands for all that ISD knowledge that is accumulated and applied in an ISD effort in an intuitive and improvising fashion (cf. Orlikowski 1996; Ciborra 1999).

The ISD knowledge can be presented in several forms in the method. We distinguish between three forms: non-structured, structured and formal. A non-structured form means that views, conceptions, principles, guidelines, rules, etc. are expressed in a natural language. A structured presentation of ISD knowledge contains also diagrammatic descriptions of ISD deliverables, ISD processes, and organizations. Part of these descriptions can be given in the form of meta models.

The notion of a formal presentation, or a formal method, is a more tricky one. According to the Oxford Advanced Learner's Dictionary of Current English, the term 'formal' means "in accordance with rules, customs, and conventions". Fitzgerald *et al.* (2002) use the term 'formalized methods' to mean "formally documented in-house and commercially available methods (e.g. SSADM, SSM)" (ibid p. 13). Wordsworth (1999) defines a formal method as "a process for developing software that exploits the power of mathematical notation and mathematical proofs" (ibid p. 1027). Pfleeger *et al.* (1997) state that a "formal technique involves the use of mathematically precise specification and design notations, and is based on refinement and proof of correctness at each stage in the life cycle" (ibid p. 34). Finally, Mathiassen *et al.* (1986, 146) recognize formality in the methods in two forms: formal models and formalized behavior. In the former case, the method provides description tools and related techniques that presume the use of formal expressions. For example, the B method (Wordsworth 1996) uses the Z notation. In the latter case, at least part of a way of working is also formalized (e.g. the Vienna Development Method (VDM) (Jones 1986)). We share the view of Mathiassen *et al.* (1986).

In our view, all three forms are needed to present the ISD knowledge in the methods. To give an overview of ISD, a non-structured form is the most suitable. To provide more exact knowledge on e.g. concepts and notations, structured forms including meta models, can be used. Formalization can also be of practical value (Pfleeger *et al.* 1997; Wordsworth 1999; Snook *et al.* 2001). It helps avoid ambiguity and perform inferences and verifications. A formalized presentation is also a prerequisite for embedding ISD knowledge into a computer-aided tool (i.e. CASE tool), which can support not only making models but also enacting ISD processes (Rolland *et al.* 1999; Koskinen 2000).

Next, we move to discuss the third dimension of our framework, which concerns generality vs. specificity of the ISD knowledge (see Figure 87). We distinguish between four types of ISD methods depending on how generic or specific ISD knowledge they convey. The method types are: generic ISD methods, domain-specific ISD methods, organization-specific ISD methods, and project-specific ISD methods. *Generic ISD methods* provide general support, such as general approaches, principles, models and guidelines, to conduct an ISD effort in a wide range of ISD contexts. These kinds of methods are called ”off-the-shelf” methods (e.g. SSADM, IE, and RUP), which must always be configured (Karlsson *et al.* 2001). *Domain-specific ISD methods* provide more domain-specific support to conduct an ISD effort in a specific application domain. The application domain may concern a geographic information system, a web information system, a mobile information systems or the like. *Organization-specific ISD methods* provide customized support to conduct an ISD effort in a specific organization. The properties of the organization-specific ISD method have to match the culture, conventions and infrastructure of the organization. *Project-specific ISD methods* provide configured and instantiated support to conduct an ISD effort in a specific project in an instantiated fashion. The properties of the project-specific ISD method have to match the needs and profile of the specific project. The division of the methods into different types is denoted with vertical lines in Figure 87. To have a more compact drawing we have grouped the ISD methods into three groups: generic methods & domain-specific ISD methods, organization-specific ISD methods, and project-specific ISD methods. As seen in the figure, proportions of different kinds of ISD knowledge vary depending on the type of the ISD method.

Let us consider in more detail how the method types differ from one another, based on Figure 88. The arrows in the figure denote the direction of configuration, customization, and instantiation of one method to another. In moving from the generic ISD methods, through the domain-specific and organization-specific ISD methods, to the project specific ISD methods the ISD knowledge, conveyed by the methods, becomes more complete, detailed and concrete. The generic ISD method provides ‘universal’ prescriptions/ descriptions of an ISD process, as well as of the environment in which an ISD process should be accomplished. It also outlines ISD roles and ISD positions in terms of responsibilities, authorities and skill requirements. Some general classifications of and suggestions for IC technology may also be incorporated to the later stages of an ISD process. In the domain-specific method the application domain, especially the part concerning an IS, is described in much more detail. As a consequence, also the knowledge of an ISD process related to the specific features of the application domain is elaborated respectively. This means that the ISD conceptual perspective (cf. Section 8.4.3) is more explicitly defined, as compared to the generic ISD methods. To customize the generic ISD method into an organization-specific ISD method, descriptions of organizational structures and ways of working, "inherent" to the particular organization, are included in the method. Also the ISD knowledge of IC technology is customized to match the technical infrastructure existing in the organization. Policies and principles concerning e.g. user participation and approval procedures are taken into account in the method. This all means that the ISD datalogical perspective is applied in the organization-specific ISD method (cf. Section 8.4.4). Finally, the project-specific ISD method prescribes/describes, in the most detailed form, contextual structures and behavior of the project, which is to accomplish a particular ISD effort.

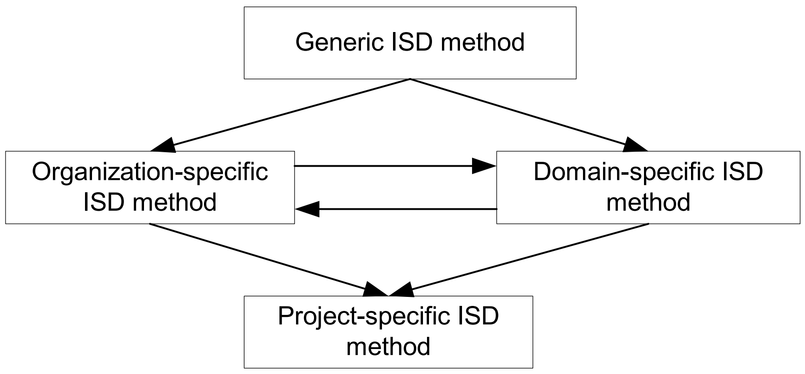

Generic ISD method
Organization-specific ISD method
Domain-specific ISD method
Project-specific ISD method

FIGURE 88 Relationships between the types of the ISD methods

Next, we compare our categorization of the ISD methods with those few categorizations suggested in the ISD literature. Vlasblom *et al.* (1995, 600) introduce the terms 'open' and 'closed' methods, but with only generic characterizations. The open methods provide more freedom on the general level, whereas the closed methods completely restrain choices. Between the two extremes, Vlasblom *et al.* (1995) identify 'half-closed' methods, in which a number of choices have already been made. This situation resembles the one that is recognized in problem-solving field, where a difference is made between weak methods and strong methods (Howard *et al.* 1999, 178). The weak methods are general. They can accommodate a large variety of problems, but give no assurance that the solution thus derived will be 'optimal'. The strong methods are only useful for certain specialized problems, but the solutions are more likely to near 'optimum'. Baskerville (1996) suggests the use of the term 'contingency methods' to mean the organization-specific ISD methods, as they are situation specific for certain types of bounded organizational settings. Harmsen (1997) uses the term 'situational method' to mean a method tailored and tuned to a particular situation. A situation means "the combination of circumstances at a given moment, possibly in a given organization" (ibid p. 25), embracing an organization, a project, and an application domain.

Though not associating to the notion of a method, Backlund *et al.* (2003) identify three levels of ISD knowledge: industry level (public knowledge), organizational level, and project level. The topmost level corresponds, to give an example, to a commercial method like RUP (Rational Unified Method (Kruchten 2000)). Also Fitzgerald *et al.* (2003) identify the industry level, the organizational level, and the project level in discussing components (e.g. standards, prescriptions, conventions, etc.) affecting the software development process.

We argue that it is highly important to differentiate between various types of ISD methods. The type of the method affects, for instance, the process and technology of engineering the method, the way of presenting and manifesting the method, and the view of how to best use the method. We have above first categorized the ISD methods according to the knowledge they convey. Another categorization has been made on the basis of the form used to represent the ISD knowledge in the method. Our second argument in this section is that though the ISD method may contain massive amount of ISD knowledge, it can, however, cover only a small portion of that knowledge that is accumulated and used in an actual ISD context.

9.4 Definition of the ISD Method

In the ISD literature, the notion of a method has been given quite different meanings[147]. To illustrate the variety of meanings we have collected exemplars of the definitions into two tables. Table 28 contains parts of the definitions characterizing the nature of the ISD method. The ISD method is seen as an approach (e.g. Brinkkemper 1996; Truex *et al.* 2000; Russo *et al.* 1996), a body of skills and knowledge (Hidding 1997), a way of accomplishing something (Kruchten 2000), a description of a technique (Hirschheim *et al.* 1995), a procedure (Kitchenham *et al.* 1999), and a system (Jones *et al.* 1988). Table 29 comprises parts of the definitions, which describe the ISD method from the structural viewpoint. The ISD method is seen as a collection (e.g. Nuseibeh *et al.* 1996; Tolvanen 1998; Avison *et al.* 1995a), a mixed bag (Rumbaugh 1995), and a system (Veryard 1987; Krogstie 1995) of various ingredients.

Method is an essential subject in the ISD research in general, and the most important concept in our study. Therefore, we take seriously the need of defining the notion as thoroughly as possible. We proceed as follows. We first define seven main views from which the method must be considered. Based on these views, we construct a general-level definition of the method. Taking the views and the definition as a departure point, we then establish the detailed ISD method ontology in the next section.

147 The term 'method' originates from a Greek term 'méthodos', meaning a systematic course. In Webster's Encyclopedic Unabridged Dictionary (1989) the method is defined as "a way of doing something, especially in accordance with a definite plan".

TABLE 28 Nature of the ISD method as seen in the ISD literature

Reference	**The method is…**
Benjamin *et al.* (1994)	″...a discipline or practice for accomplishing some set of tasks″ (ibid p. 169).
Brinkkemper (1996)	“an approach to perform a system development project” (ibid. p. 275). “the systematic description, explanation and evaluation of all aspects of methodical ISD” (ibid p. 276).
Green *et al.* (2000)	″an approach to system planning, analysis, design, and construction″ (ibid. p. 73).
Heym *et al.* (1992a)	″..an approach to information systems development″ (ibid p. 215).
Hidding (1997)	“a body of skills and knowledge” (ibid p. 105).
Hirschheim *et al.* (1995)	″...a description of a technique″ (ibid p. 11).
Jayaratna (1994)	“… an explicit way of structuring one’s thinking and actions” (ibid p. 242).
Jones *et al.* (1988)	″a system which provides the information required to build information systems″ (ibid p. 264).
Kitchenham *et al.* (1999)	″...a procedure defining steps and heuristics to permit the accomplishment of one or more activities″ (ibid p. 376).
Kruchten (2000)	“way of accomplishing something” (ibid p. 276).
Russo *et al.* (1996)	″... an approach to conducting at least one complete phase […] of computer information system development″ (ibid p. 387).
Truex *et al.* (2000)	“… an approach to information systems development “ (ibid p. 54).

Method is a very multifaceted concept. As concluded in Section 9.3, it is a kind of knowledge base or an organizational memory that is accumulated and shared across organizational boundaries and time dimension (Schönström *et al.* 2003; Backlund *et al.* 2003). It should also carry justifications for its creation and modifications (Kaipala 1997; Rossi *et al.* 2004). It is a linguistic artifact written, edited, read and interpreted by ISD actors in different ISD roles, in different contexts (Jayaratna 1994; Hidding 1997). In addition, it can be considered an aggregate of various constituents which may have different appearances.

To cope with the most essential facets, we define and apply seven views of the notion of an ISD method. To distinguish these views from the other views defined in this work, we call them the *methodical views*. The methodical views are (Figure 89): (a) The historical view: What kind of historical background does the method have? (b) The application view: Where and how is the method to be applied? (c) The contents view: To what kinds of phenomena do the knowledge within the method refer? (d) The presentation view: In what ways is the knowledge within the method presented? (e) The physical view: How is the ISD method materialized? (f) The structural view: What is the structure of the method?

The methodical views roughly correspond to the layers of the semantic ladder (Stamper 1973; Stamper 1996). The historical view and the application

TABLE 29 Notion of the ISD method seen from the structural viewpoint in the ISD literature

Reference	**The method ..**
Avison *et al.* (1995a)	"is a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement a new information system" (ibid p. 10).
Iivari *et al.* (1998a)	is "an organized collection of concepts, methods, beliefs, values and normative principles supported by material resources" (ibid p. 165).
Karlsson *et al.* (2001)	"consists of three interrelated parts: a process, some sort of notation and a set of concepts used to describe the problem domain and the method itself." (ibid p. 2).
Krogstie (1995)	"is a system of rules, techniques, and tools to aid development and/or maintenance of application systems." (ibid p. 13).
Nuseibeh *et al.* (1996)	"is a collection of procedures and heuristics for…" (ibid p. 269).
Rumbaugh (1995)	"is a mixed bag of guidelines and rules, including the following components: a set of fundamental modeling concepts, a set of views and notations for presenting the underlying modeling information, a step-by-step process for constructing models and implementations of them, a collection of hints and rules of thumb for performing development." (ibid p. 10).
Tolvanen (1998)	"is a collection of techniques and a set of rules which state by whom, in what order, and in what way the techniques are used, to achieve or maintain some objectives." (ibid p. 33).
Veryard (1987)	"is a system of tasks and techniques, supported by automated tools and/or directed experience, for carrying out some or all of the following IS activities…" (ibid p. 469).

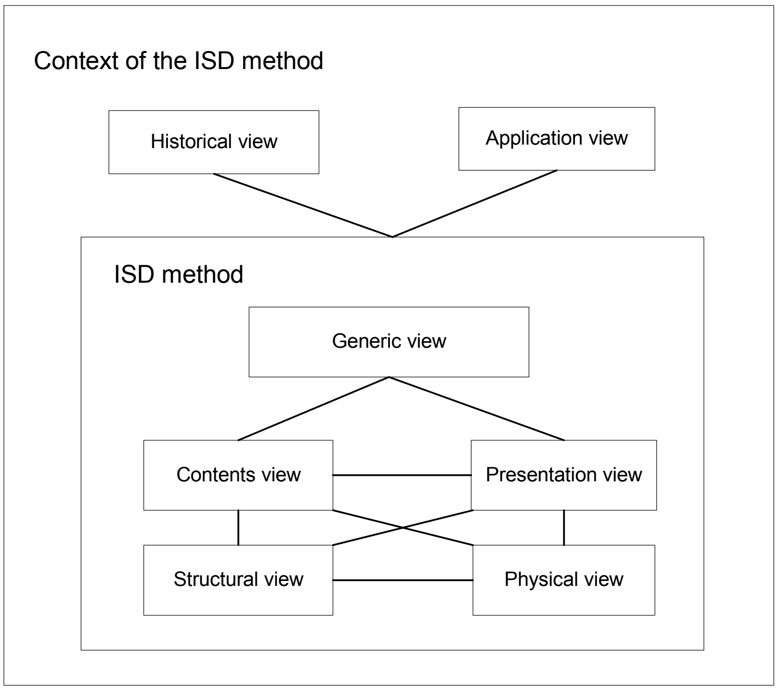


FIGURE 89 Methodical views

view "contextualize" the method into the past contexts in which the method has been engineered and applied, as well as into the future contexts, in which the method is to be engineered and applied. By these views only, the real nature and meaning of the method can be understood. These views reflect aspects of social world and pragmatics (cf. Section 4.3). The generic view provides an overview and general understanding of the method. The contents view reveals the object system (OS_{ISD}), that is to say, the semantics of the method (cf. the layer of semantics). This view was applied in Chapter 8 to establish the ISD ontology. The presentation view, corresponding to the syntactic layer, is used to consider the method as a set of expressions presented in some language. The physical view considers the method as a materialized thing. Finally, from the structural view the method is seen as a modular structure composed of various descriptive and prescriptive parts.

Based on the methodical views we can now present an integrated definition of the ISD method for the use of this work:

> The *ISD method* is an artifact anchored on certain historical, intentional and functional backgrounds and aimed to be applied and deployed as a prescription in the intended kinds of ISD contexts, in order to make organizational and technical changes in IS's possible or more productive. The ISD method, presented and materialized in certain forms, contains four kinds of knowledge[148] bringing out how ISD actors carry out ISD actions to produce ISD deliverables, by means of ISD facilities, in an organizational and spatiotemporal context, in order to satisfy ISD goals set by ISD stakeholders. The ISD method is composed of descriptive and prescriptive parts with a large variety.

In the next section we apply these methodical views, except the contents view, to establish the ISD method ontology. As said above, the contents view was already applied in Chapter 8 in engineering the ISD ontology.

9.5 Definition of the ISD Method Ontology

Generally speaking, an ontology is "consensual knowledge represented in a generic and formal way to be used and shared across applications..." (Corcho *et al.* 2003, 44). An ontology is also seen as an artifact engineered with the purpose of expressing the intended meaning of a shared vocabulary (cf. Uschold *et al.* 1996). Recalling that there is a wide variety of conceptions about the ISD method in the literature, we need a systematic and detailed way to define the nature, contents and structure of the notion. We do this through the ISD method ontology.

---

148 Recall the division of ISD knowledge into four components: knowledge of ISD process, knowledge of application domain, knowledge of IC technology, and knowledge of human and social issues (see Section 9.3).

The *ISD method ontology* is composed of concepts and constructs, with which contextual aspects of the ISD methods can be conceived, understood, structured, and represented. Due to the complexity of the ISD method ontology, we decompose it into seven parts based on the seven methodical views defined in Section 9.4. An overall picture of the ISD method ontology is presented in Figure 90. Next, we define the views and the concepts comprised by the views.

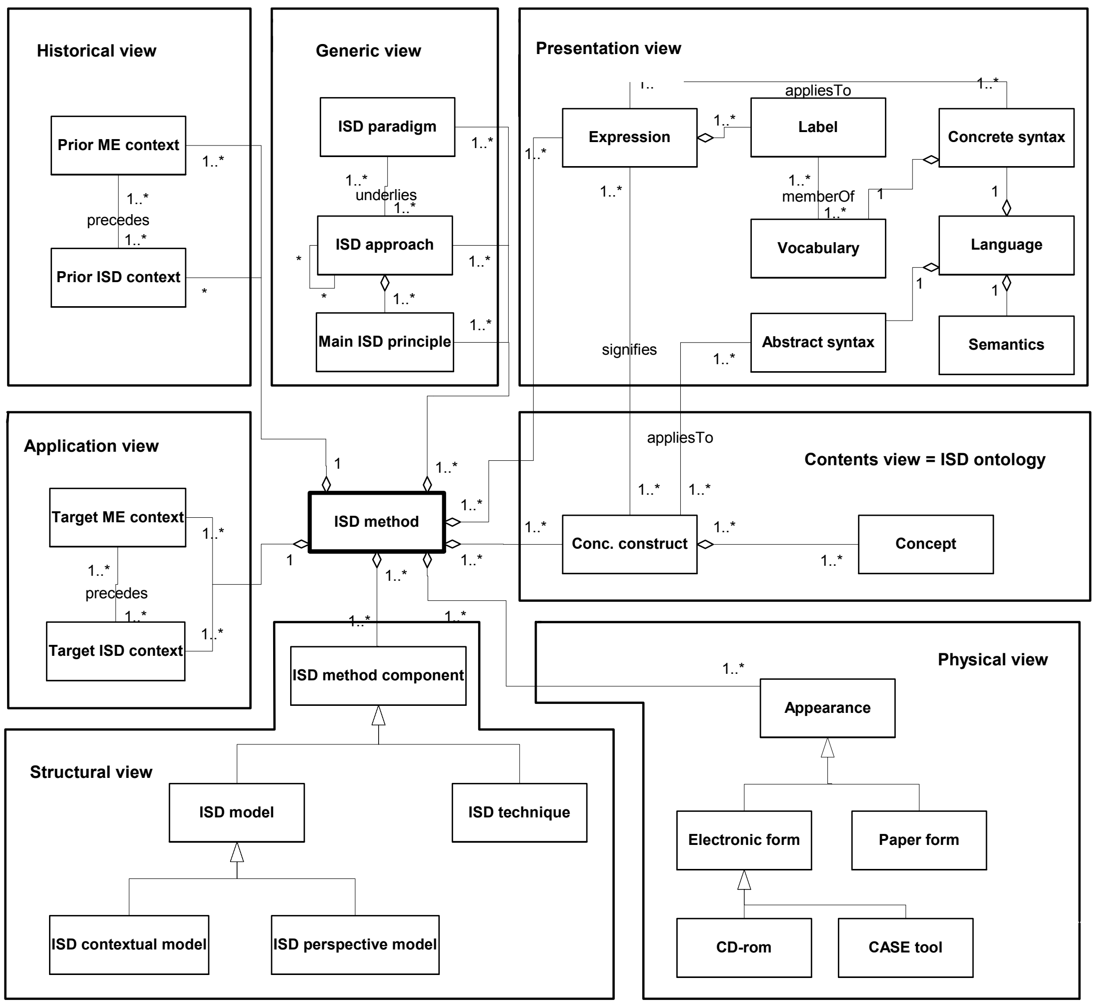


FIGURE 90 An overall picture of the ISD method ontology

A. Historical View

The *historical view* enlightens the backgrounds of and experience from the engineering and use of the ISD method. It involves both prior ME contexts and prior ISD contexts. *Prior ME contexts* mean those contexts that have contributed to the creation and engineering of the ISD method. *Prior ISD contexts* mean those contexts in which the ISD method has been deployed. Resulting from the former, the ISD method has to include knowledge of the intentions, approaches and principles by which the ISD method has been engineered, of ME actors who have been responsible for the engineering, and of ME actions by which the

ISD method has been engineered, etc. This knowledge is known as the method engineering rationale. It establishes a systematic and organized trace of method evolution (cf. method construction rationale in Rossi *et al.* 2004). Resulting from the latter, the method has to include knowledge about e.g. applications developed, ISD actors involved, and successes/failures encountered in prior ISD contexts in which some version of the ISD method has been deployed. This knowledge is known as method use rationale (Rossi *et al.* 2004) or "specification development status history and rationale" (Nuseibeh *et al.* 1996). It is a kind of "experience base" which can be used to help making and justifying the decisions on whether to use and how to use the ISD method. Both of these bodies of knowledge from history are important to considering the relevance and applicability of the method to the ISD context at hand. A detailed description of an ME context can be established with concepts and constructs given in the ME ontology in Chapter 10. The corresponding concepts and constructs for the description of an ISD context are contained in the ISD ontology in Chapter 8.

B. Application View

The *application view* outlines where and how the ISD method can be applied. Consequently, the ISD method contains descriptions of those ISD contexts, called *target ISD contexts,* for which the ISD method is intended, as well as descriptions of those ME contexts, called *target ME contexts,* in which the ISD method is to be customized and instantiated for the use of a particular organization or project. A target ISD context means an application area, that is to say, some part of the real world that is seen to be problematic and worthy of investigation (Checkland 1981). In the ISD method the arguments for the applicability to certain kinds of ISD contexts should be justified with appropriate evidence. Evidence can be based on logical argumentation derived from the perceived match between the ISD method and the suggested application areas, or on empirical experience got from the prior usages (cf. method rationale in Rossi *et al.* 2004). For the target ME contexts, it is necessary to include in the method outlines of factors to be considered and guidelines to be followed in the customization.

C. Generic View

The *generic view* provides the general understanding of the nature of the ISD method. It highlights: (a) the philosophic assumptions and values, on which the ISD method has been built (cf. ISD paradigms), (b) the ISD approaches to be applied in the target ISD contexts, and (c) the main ISD principles to be followed in the target ISD contexts.

The *ISD paradigm* means here a set of fundamental paradigmatic and philosophical assumptions underlying the ISD method. The paradigmatic assumptions are divided into four categories (see Section 8.1.1): (1) assumptions about the nature of the IS and the ISD (ontology), (2) assumptions about what human knowledge is and how it can be acquired in the IS and the ISD (epistemology), (3) research methodology, and (4) ethics (Iivari 1991; Iivari *et al.* 1998a).

The *ISD approach* means a generic way of conceiving certain aspects of ISD and/or a generic way of working in ISD (cf. Section 8.1.2). Among a large variety of ISD approaches we distinguish between the approaches that are based on (a) different schools of thought, (b) different views of ISD, and (c) different focuses on the contextual ISD domains. The ISD approaches are highly interrelated.

The *main ISD principle* expresses essential aspects of a way in which the ISD context is to be structured, accomplished, and/or managed. Examples of the main ISD principles are: iterative design, end-user participation, problem-orientation, and contingency-based application of the ISD method. Hidding (1997, 107) call this "the first principle" and defines it as a "way of thinking" that is embedded in a particular method.

The ISD paradigms, the ISD approaches and the main ISD principles are inter-related to one another. The relationships include the "inheritance" relationship (cf. Iivari *et al.* 2001, 188). Several ISD approaches may be needed to confirm a particular paradigmatic assumption. An ISD approach in turn can be based on one or more paradigms. An ISD approach can lead to the application of one or more main ISD principles, and a particular principle may be followed to implement one or more ISD approaches.

D. Contents View

The *contents view* reveals the conceptual contents of the ISD method. According to this view, the ISD method is composed of concepts and conceptual constructs referring to the ISD context, as well as to some parts of the ME context(s). The former correspond to the prior and target ISD contexts. The conceptual contents of the ISD context have been established in the form of the ISD ontology in Chapter 8. The latter mean the prior and target ME contexts. The conceptual contents of the ME context will be presented in the form of the ME ontology in Chapter 10.

E. Presentation View

From the *presentation view* the ISD method is seen as a set of expressions presented in some language(s). Expressions signify conceptual constructs constituting the contents of the ISD method. A language is defined by an abstract syntax, a concrete syntax (or notation) and semantics. An abstract syntax states the allowed conceptual constructs composed of concepts (ter Hofstede *et al.* 1998). A concrete syntax gives the notational elements, including labels, of a language and rules for connecting them with one another and with the concepts. Semantics specifies the meaning of notational elements.

F. Physical View

The *physical view* reveals the appearance(s) of the ISD method, that is to say, the media on which the ISD method is made visible or "functioning". The ISD method may appear in a paper form (e.g. text books, manuals, pro forma documents), or in an electronic form (e.g. CD-rom, Word Wide Web). It can be in a form of a lecture with e.g. Power Point slices and implemented with CASE tools. CASE tools support the creation and editing of IS models and their implementation (e.g. Rational Rose, ParadigmPlus, Objecteering), ISD process management (e.g. Ernst & Young's Navigator), James Martin & Co.'s Architecture and SHL/Transform), and ISD process enactment (Koskinen 2000).

G. Structural View

From the *structural view* the ISD method is seen as a modular structure of various parts. This structure embraces e.g. paradigmatic assumptions, ISD approaches, ISD principles, background and application knowledge, concepts, notations, ISD models, ISD techniques, ISD rules, and ISD guidelines (see Figure 90). Some of these parts are considered method components. An *ISD method component* is a well-defined part of the ISD method that can be integrated to other ISD components to form a coherent and consistent ISD method. In Figure 90 we recognize two kinds of ISD method components, ISD models and ISD techniques. In Section 9.7 we discuss more about the notion of a method component and identify several kinds of method components.

An *ISD model* is a model that describes/prescribes structural and/or behavioral features of the ISD context(s). An *ISD technique* is a technique, which guides the accomplishment of specific actions in the ISD context(s). The technique can be presented as a set of precisely described procedures that help the achievement of certain outcomes if executed correctly (cf. Kettinger *et al.* 1997, 58; Iivari *et al.* 2001, 186). ISD techniques may also be presented in heuristics, in guidelines or as rules of thumb. An ISD technique can involve one or more ISD models, and an ISD model can be involved by one or more techniques.

ISD models are further specialized into ISD contextual models and ISD perspective models (cf. Section 7.2). *ISD contextual models* refer to ISD models that are classified into eight categories according to the contextual domains they address. The categories are (cf. Figure 72 in Section 7.4): ISD purpose models, ISD actor models, ISD action models, ISD deliverable models, ISD data models, ISD facility models, ISD location models, ISD time models and ISD inter-domain models (ISD ID models). The models have the following functions (see Figure 91):

- The *ISD purpose model* prescribes/describes problems in, requirements for, and/or goals of, the intended[149] ISD context, or some part(s) thereof.

[149] The intended ISD context refers to a prior ISD context or a target ISD context.

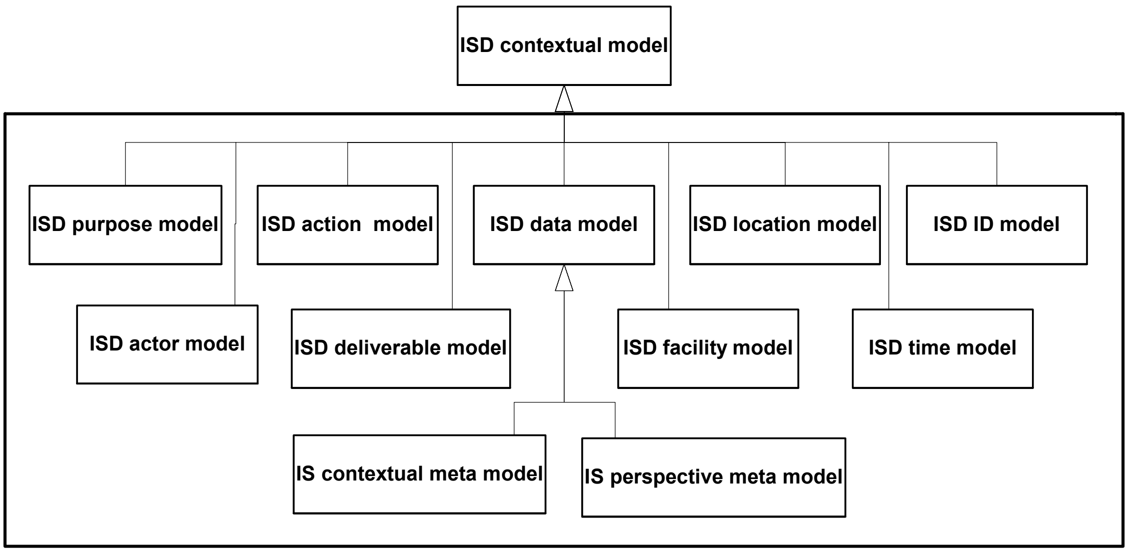


FIGURE 91 ISD contextual models

- The *ISD actor model* prescribes/describes ISD roles, ISD positions, ISD organization units, persons and/or groups participating one way or another in the intended ISD context.
- The *ISD action model* prescribes/describes ISD actions and their relationships in the intended ISD context. Each action model brings out various ISD action structures (e.g. generic action structures, IS modeling structures, ISD phase structures, ISD management–execution structures, and ISD workflow structures) as detailed in Section 8.3.3.
- The *ISD deliverable model* prescribes/describes the structure and presentation of ISD deliverables and how they are related in the intended ISD context.
- The *ISD data model* prescribes/describes the conceptual contents of the ISD deliverables in the intended ISD context.
- The *ISD facility model* prescribes/describes resources and tools available and used in the intended ISD context. The resources include e.g. manpower and money, and the tools cover manual, computer-aided and automated tools.
- The *ISD location model* prescribes/describes the nature, structure and features of locations, whether physical or logical, involved in the intended ISD context.
- The *ISD time model* prescribes/describes the time system used in the intended ISD context.
- The *ISD ID model* prescribes/describes inter-domain features of the intended ISD context. Questions relating to the inter-domain features include: Who is responsible for certain ISD actions? What are the inputs to and outputs from certain ISD actions? What tools are to be used in producing certain ISD deliverables?

The *ISD perspective models* refer to ISD models that can be classified into six categories according to which ISD perspective they address. The categories are (Figure 92): ISD systelogical models, ISD infological models, ISD conceptual models, ISD datalogical models, ISD physical models, and ISD inter-perspective (shortly ISD IP) models. These ISD perspective models have the following functions:

- The *ISD systelogical model* describes/prescribes the support the intended ISD should provide for its utilizing system (US_{ISD}), as well as the assumptions on the target IS's and their US_{IS}.
- The *ISD infological model* describes/prescribes the purposes, actions and deliverables of the intended ISD context.
- The *ISD conceptual model* describes/prescribes the conceptual contents of the deliverables of the intended ISD context.
- The *ISD datalogical model* describes/prescribes the purposes, actions, deliverables, actors, and tools of the intended ISD context, last two on a general level.
- The ISD physical model describes/prescribes, besides the features mentioned above, yet on a more concrete level, also spatial and temporal features of the intended ISD context, instantiated into a particular ISD context.
- The *ISD IP model* describes/prescribes features of the intended ISD context from more than one ISD perspective.

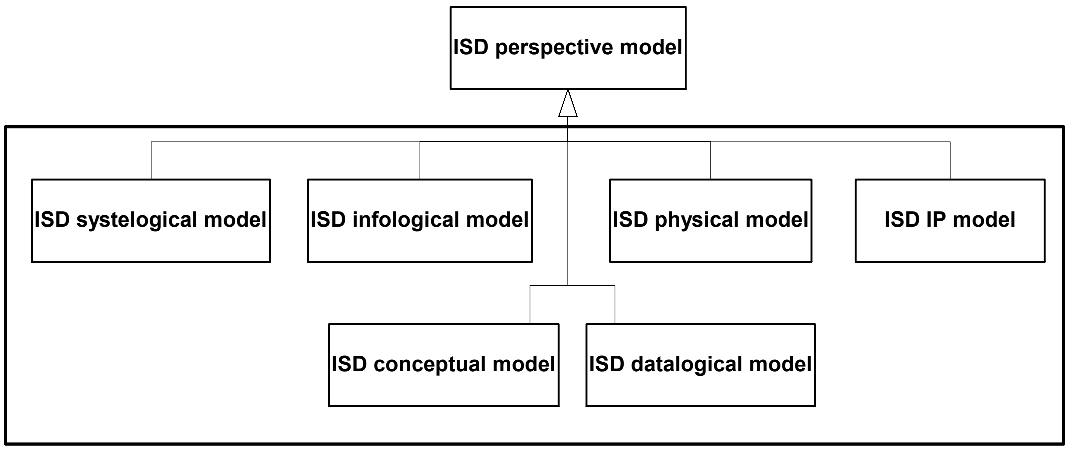


FIGURE 92 ISD perspective models

Some of the ISD data models concern the contents of the IS data. These models are called the IS meta data models[150]. The ISD data models can be specialized into the IS contextual meta models and the IS perspective meta models (Figure

[150] Kumar *et al.* (1992, 264) calls this part of the method "the representation frame" in contrast to "the process frame" which concerns e.g. the definition of and sequence of development workflows, actions, and tasks and the definition of ISD actors and their roles in the ISD process.

91). The IS contextual meta models, in turn, can be classified into eight meta models according to the IS contextual domains. The categories are (Figure 93): IS meta purpose models, IS meta actor models, IS meta action models, IS meta deliverable models, IS meta data models, IS meta facility models, IS meta location models, and IS meta time models. Further, we can recognize IS meta inter-domain (ID) models. The IS meta models provide concepts (i.e. the abstract syntax) and/or notations (i.e. the concrete syntax) of the language(s) in which the corresponding IS models are, or are to be, expressed[151].

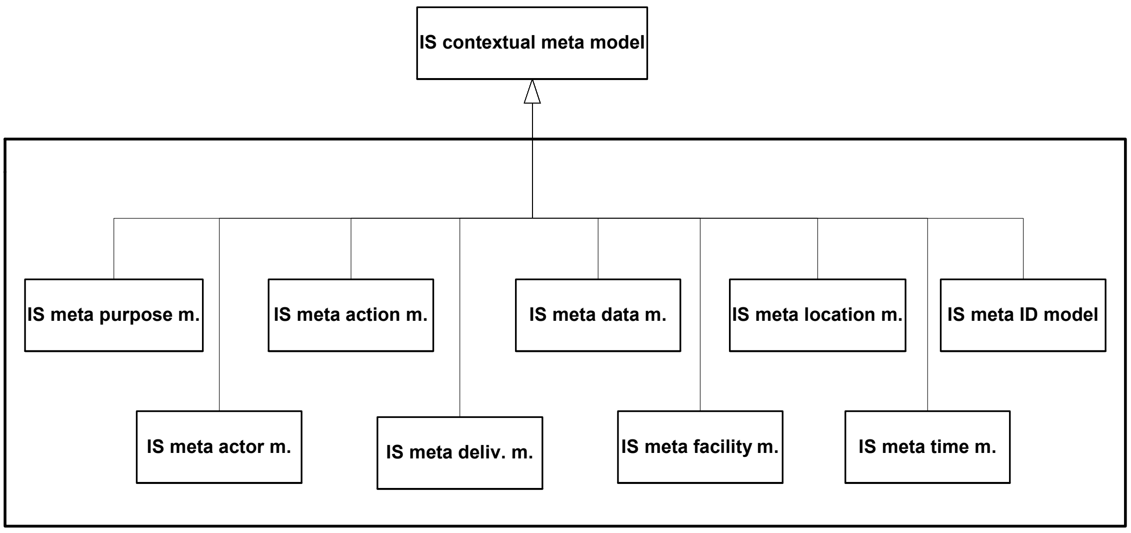


FIGURE 93 IS contextual meta models

The ISD method ontology presented above has been derived from, and structured according to, the underlying ontologies. This becomes obvious in looking at the overall picture of the ontology (Figure 90). The division into the presentation view, the contents view, and the physical view is a specialization of the semiotic ontology. The internal structure of the presentation view results from the language ontology. The historical view and the application view, defined in terms of the prior and target ISD and ME contexts, is an application of the context ontology. The conceptions of ISD contexts and ME contexts can be further elaborated by the concepts and constructs provided in the ISD ontology and the ME ontology. The structural view decomposes the ISD method into parts, which are recognized in the model level ontology.

151 Sections 4.4.-4.6 and Section 6.3 provide concepts and constructs for the IS models in the form of meta models.

9.6 Methodical Support

Up till now we have considered methodical support from the viewpoint of ISD methods only. Actually, ISD can be methodically supported with many kinds of descriptions/ prescriptions on various levels of detail and concreteness. In this section we will first present a range of artifacts that methodically support ISD, compare them with one another, and consider one artifact type, an ISD methodical skeleton in particular. Second, we will discuss whether there are explicit criteria for deciding if an artifact can be regarded as an ISD method or not. The considerations and discussions in this section are based on the ISD method ontology.

9.6.1 Range of Artifacts

Knowledge about ISD process, application domain, IC technology, and human, social and organizational issues are accumulated, presented and shared in various forms and details. One of the most low-level and fuzziest fashion to present and share guide-lines for ISD is to serve them with a set of tips or hints. These "best practices" are commonly written by consultants in commercial magazines (Makmuri 1998). Alternatively, ISD knowledge can be packaged into the form of ISD approaches and generic principles (see Bracchi *et al.* 1984; Vessey *et al.* 1994; Iivari *et al.* 2001). We have sub-divided ISD approaches into three categories (i.e. Categories A, B and C) in Section 8.1.2, based on the scale of their scopes. Common to all of them is that the support that they provide remains on a very general level in order to satisfy practical needs of ISD contexts. The ISD literature also proposes a large array of ISD techniques (e.g. normalization technique (Codd 1972), transformation technique (Batini *et al.* 1992), and conceptual modeling technique (Gemino *et al.* 2002)) and IS models (e.g. ER model (Chen 1976), role activity model (Kueng *et al.* 1996), data class / process matrix (IBM 1984) and responsibility matrix (van Slooten *et al.* 1993)) to guide the accomplishment of certain ISD actions and to present their outcomes. ISD techniques are described in terms of generic principles, such as in Codd (1972), or with detailed steps, such as in Batini *et al.* (1992).

The aforementioned artifacts provide ISD knowledge that is either too vague (e.g. hints and tips), on too a general level (e.g. ISD approaches and generic principles), or too narrow-scoped (e.g. single IS models and ISD techniques) to satisfy needs for the guidance in ISD contexts. In the following we extend the range of artifacts to embrace those that have a more comprehensive scope. They are: an ISD methodical framework and an ISD methodical skeleton.

An *ISD methodical framework* consists of meta models. In a simple form, an ISD methodical framework is composed of IS meta models that are used to semi-formally specify IS models. The IS meta models provide the concepts and constructs used in an ISD effort but only in those parts which concern phenomena in the IS domains. In a more comprehensive form, an ISD methodical framework also includes ISD meta models describing generally ISD processes.

An ISD methodical framework is descriptive and contains no normative ingredients. An ISD methodical skeleton provides all that is contained in an ISD methodical framework and in addition prescriptive constructs for a skeleton-like structure of the ISD context. This structure integrates and gives normative meanings for the IS meta models, as well as instantiates and specializes the ISD meta models. To put it more precisely, an *ISD methodical skeleton* is a normative prescription for the ISD context, structuring and guiding the ISD process on a general level. An ISD methodical skeleton is not a complete ISD method. In contrast, an ISD method can be engineered elaborating and refining an ISD methodical skeleton.

The notion of a methodical skeleton is important in our work, because we aim to develop a methodical skeleton for ME (see Chapter 11). Therefore, we elaborate the nature and contents of this artifact, in relation to ISD in this section, with the ISD method ontology. Figure 94 presents an ISD methodical skeleton and its intended use in ISD, as well as the overall structure of the ISD method ontology. Included in the ISD method ontology there are the methodical views, the ISD ontology, and the IS ontology. The ISD ontology and the IS ontology are composed of the domains and the perspectives. The arrows denote how the ontologies are deployed to engineer an ISD methodical skeleton. We can see that the structure of an ISD methodical skeleton is adapted, in quite a straightforward way, from the ISD method ontology. The main parts of an ISD methodical skeleton are methodical views, ISD models, ISD meta models, and IS meta models. The ISD models are specialized and instantiated from the corresponding ISD meta models. The ISD models, the ISD meta models and the IS meta models in the skeleton address only some of the ISD domains and the IS domains, on a general level, and from the limited viewpoints.

In Section 9.3 we distinguished between four types of ISD methods: a generic ISD method, a domain-specific ISD method, an organization-specific ISD method, and a project-specific ISD method. To compare an ISD methodical framework and an ISD methodical skeleton with one another and with the types of ISD methods, we next position them in the space established by the model levels and the ISD perspectives (see Figure 95). We use ellipses to depict the scopes covered by the artifacts. An ISD methodical framework provides meta models, mostly for the ISD systelogical perspective, the ISD infological perspective, and/or the ISD conceptual perspective (cf. the IS meta models). An ISD methodical skeleton covers the same ISD perspectives as above but extends in part to the type model level as well. A generic ISD method embraces all what has been said above and extends more clearly to concepts, models, and guidelines for the ISD datalogical perspective. A domain-specific ISD method differentiates from a generic ISD method only in the level of specialization into which concepts in IS meta models are extended. Therefore, we do not present

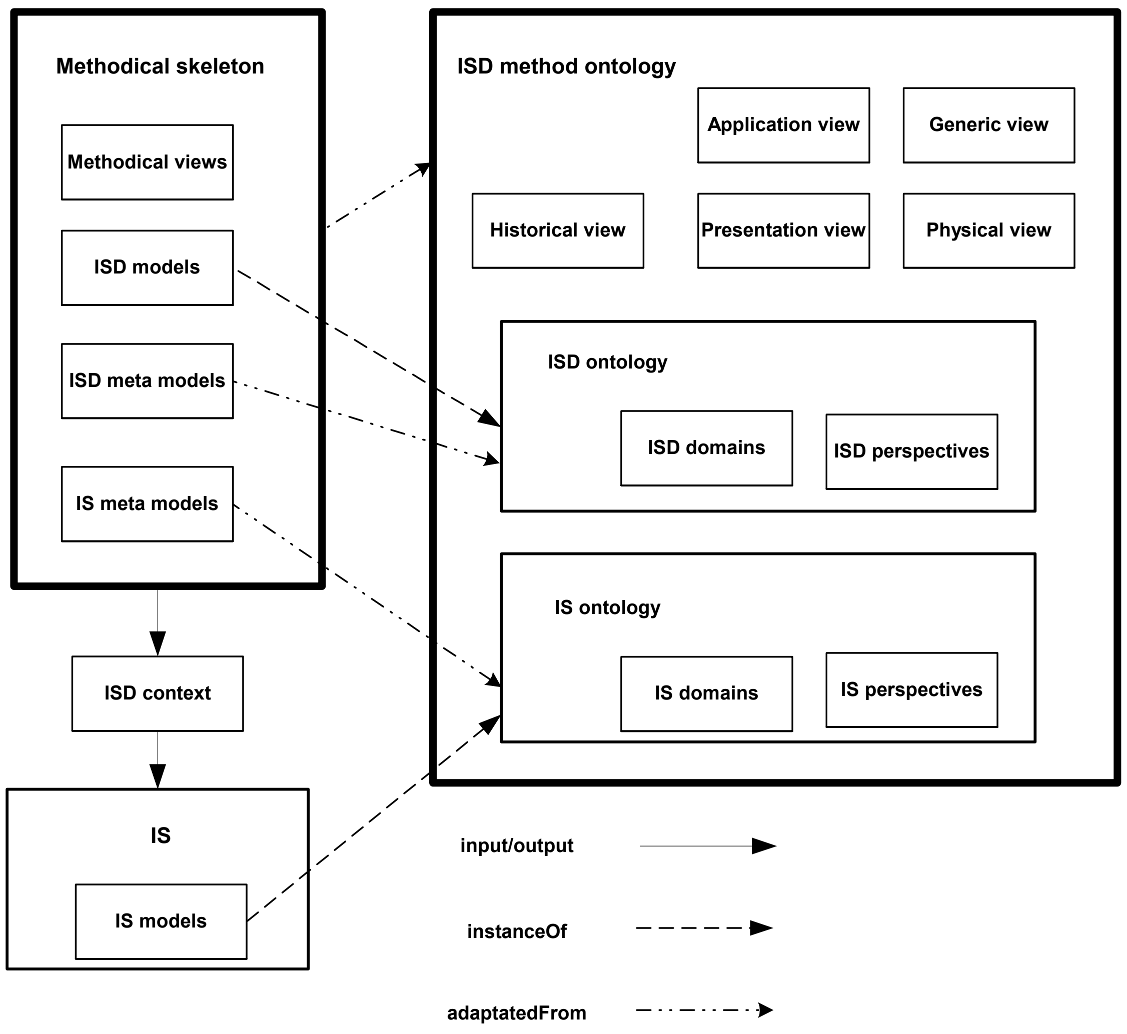


FIGURE 94 Basis and contents of an ISD methodical skeleton

separately the scope of this type of ISD method in Figure 95. An organization-specific ISD method provides general issues also from the ISD physical perspective and, in some cases, present ISD knowledge also on the instance model level. A project-specific ISD method provides ISD knowledge with the most extensive scope, covering all five ISD perspectives and all three model levels.

There is still one type of artifact that can be used to support ISD. We call it a methodical tool kit (cf. Benyon *et al.* 1987). A *methodical tool kit,* or a method base, is a collection of more or less unrelated methodical parts, which do not, as such, constitute any coherent and concrete method. In contrast, the idea is to select suitable parts from a methodical tool kit and integrate (or assembly) them to engineer an ISD method (e.g. Kronlöf 1993; Song 1997; Harmsen 1997; Saeki 1998; Wieringa *et al.* 1998). These parts may be on various levels of specificity and detail. Most commonly they are IS meta models and generic ISD models. Due to the variety of these parts, a methodical tool kit is not included in the comparison in Figure 95.

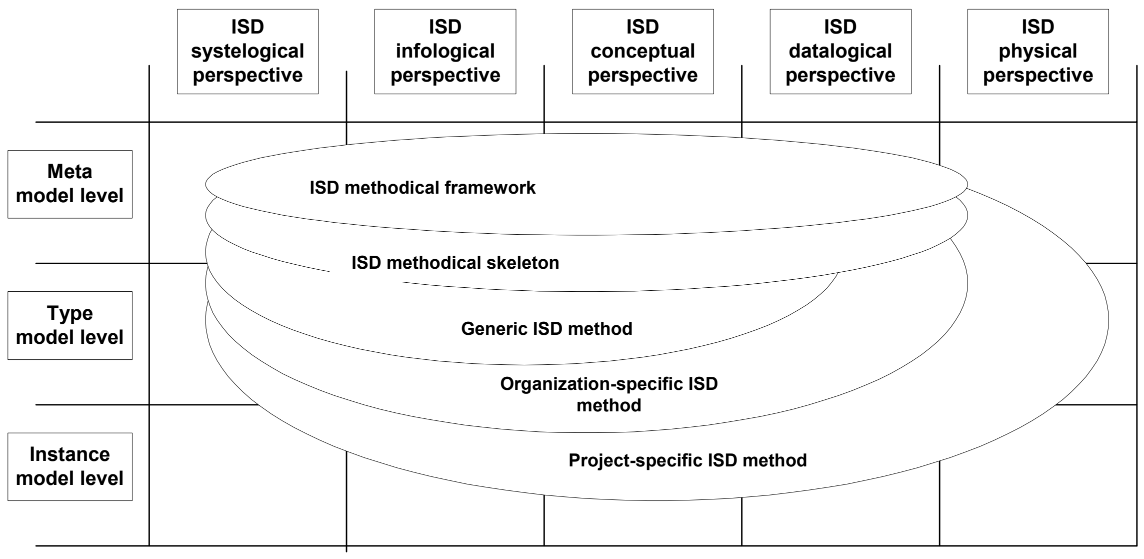

ISD systelogical perspective
ISD infological perspective
ISD conceptual perspective
ISD datalogical perspective
ISD physical perspective
Meta model level
Type model level
Instance model level
ISD methodical framework
ISD methodical skeleton
Generic ISD method
Organization-specific ISD method
Project-specific ISD method

FIGURE 95 Scopes of the artifacts supporting ISD

Next, we compare our range of methodical artifacts to ISD artifacts presented in the ISD literature. In the ISD field the two best known generic methodical artifacts are the Unified Process (Jacobson *et al.* 1999) and the OPEN framework (Graham *et al.* 1997). In the Unified Process (Jacobson *et al.* 1999) the notion of a process refers "to a concept that works as a template that can be reused by creating instances of it" (ibid p. 24). It is compared to a class form, which can be used to create instances. Hence, the Unified Process is to be instantiated to make a project-specific process, called a process instance or a project. The key concepts in the Unified Process are an artifact, a model, a worker, and a workflow. OPEN is "a framework for third-generation OO software development methods" (Graham *et al.* 1997, 4). A software engineering process (SEP) is "a time-sequenced set of activities and provides a tested and well-defined approach to the development of OO software systems" (ibid p. 6). According to Henderson-Sellers and Mellor (1999c 40) OPEN is a "methodological framework" or a "process metamodel" which can be instantiated to have specific methodological processes, such as SOMA (Graham 1995) and RUP (Kruchten 2000). A process has three dimensions: methodology, people and organizational influences, and technology. Generalizing the SEP's, the underlying architecture, known as the software engineering process architecture (SEPA), may be distinguished. SEPA constitutes the core of the OPEN framework. The key concepts in the OPEN framework are an activity, a task, a technique, and a deliverable.

In the Unified Process (Jacobson *et al.* 1999) and the OPEN framework (Graham *et al.* 1997) ISD actions and ISD deliverables are specified on a rather detailed level. The Unified Process introduces the ISD workflow structure and the ISD phase structure. The OPEN framework defines activities and for each activity a set of ISD tasks. In both of the artifacts ISD deliverables are specified and illustrated with examples. ISD actions and ISD deliverables are associated to the corresponding IS meta models (i.e. UML in Jacobson *et al.* (1999) and

OML in Graham *et al.* (1997)). In the Unified Process ISD, roles, known as workers, are introduced and associated to ISD actions. In the OPEN framework a large set of ISD techniques is given.

Based on what has been said above, we can state that both the United Process (Jacobson *et al.* 1999) and the OPEN framework (Graham *et al.* 1997) are methodical skeletons, as understood in our work. They don't aim to provide an ISD method, but something more "abstract" from which an ISD method can be instantiated. Instantiation in these cases embraces more than the application of the instantiation principle (cf. Section 3.9.2.1). For some aspects of ISD contexts, the artifacts specify quite detailed and concrete prescriptions, e.g. steps for the accomplishment of tasks. The Unified Process also recognizes a large set of ISD roles, meaning that it partly applies the ISD datalogical perspective.

In addition to the Unified Process and the OPEN framework, there are other presentations that are comparable to the artifacts in our range. Hruby (2000b) presents a 'methodological framework', which regards the 'software development artifacts' as the most essential constructs. The artifacts are viewed as conceptual, not as representational entities (ibid p. 23). In engineering an ISD method, artifacts are first selected. Artifacts (artifact types) have two kinds of "methods"[152] that guide how to create, interrelate and check the artifacts (instances). For instance, the "methods" specify preconditions requiring that certain artifact (instance) must exist before another artifact (instance) can be created. Thus, preconditions indirectly impact on an order in which the ISD actions creating the artifacts should be performed. For instance, to create a class life cycle, the artifact 'class' must be first created (ibid p. 29). When selecting and including artifact (types) to the body of an ISD method, the "methods" attached to the artifacts also indicate which kinds of ISD actions there should be in an ISD method. Hruby (2000b) calls his framework a "product-focused" framework, compared to the OPEN framework (Graham *et al.* 1997), which he calls the "process-focused" framework. Concluded from the above, the methodological framework of Hruby (2000b) is not an ISD method, not even an ISD methodical skeleton. It corresponds to our notion of a methodical framework, containing, as suggested by Hruby (2000b), ISD meta deliverable models from which basic ISD action structures (i.e. ISD models) can be derived when instantiating the framework for a particular ISD effort.

Vlasblom *et al.* (1995, 601) propose a three-level description of an ISD method. The levels are: the generic level, the model level, and the specific level. The generic level is composed of building blocks for various elements of an ISD method (i.e. products, activities, development strategies, techniques, tools etc). The model level contains development models for specific application domains. The lowest level corresponds to project-specific development methods, called project scenarios. The generic level corresponds to our notion of a methodical tool kit. Vlasblom *et al.* (1995) present a seven-point protocol for establishing a development method by selecting specific building blocks.

[152] A method here corresponds to the notion of a method in the object-oriented paradigm. To separate it from our term, we present it here in quotation marks.

9.6.2 Requirements for an ISD Method

As illustrated in Section 9.4, the ISD literature reflects quite divergent conceptions about the contents and structure of an ISD method. But what should be required from an artifact in order to acknowledge it as an ISD method? Is a set of IS meta models, for instance, enough, or should an ISD method also offer procedures, rules and guidelines? What about the necessity of support for project management? Is a set of separate techniques enough for an ISD method, or should there be a kind of frame to integrate them into a whole? How mandatory part in an ISD method is a computer-aided tool supporting the method use? To these questions we find only a few answers in the literature. One of the most challenging lists of requirements is given by Graham *et al.* (1997) who state that an ISD method should provide "at least a full life cycle process, a comprehensive set of concepts and models, a full set of techniques, a fully delineated set of deliverables, a modeling language, a set of metrics, quality assurance, standards, reuse advice, and guidelines for project management" (ibid p. 2). One may ask whether there is any ISD artifact that fulfills these requirements. In our view, although no clear-cut limits can be specified, there are, however, some requirements that an ISD method should fulfill. In the following, we bring out our conception based on the ISD method ontology.

In Figure 90 we presented the overall structure of the ISD method ontology reflecting seven methodical views. Following this structure we bring out the following requirements for an ISD method. First, any artifact to be acknowledged as an ISD method should contain the part that provides knowledge of how it has been engineered and with what experience it has been used earlier (cf. the historical view). Without this knowledge, there is a risk that unwritten intentions, approaches and principles considered important in the engineering of the artifact appear to be unsuitable to the situation at hand. Another risk is that problems and failures experienced in prior efforts will reoccur if they are not learned. Hence, it is required that every ISD method must contain method engineering rationale and method use rationale (cf. Rossi *et al.* 2004). Second, it is necessary that an ISD method provides a description of where and how it can be applied (cf. the application view). This implies that the target ISD contexts as well as the target ME contexts should be outlined. Only in this way it is possible to figure out whether an artifact is initially even intended for the situation at hand. Third, every artifact to be regarded as an ISD method must provide at least two kinds of languages for the presentation of ISD deliverables (cf. the presentation view). One of these languages is a natural language that helps ISD stakeholders communicate with one another through ISD deliverables. Other language(s) should be semi-formal (e.g. a graphical language), or formal, to enable IS developers present and reason from complex structures and behavior of an IS in a concise and strict fashion. Fourth, an artifact should carry some information about its underlying paradigmatic assumptions (cf. the generic view), in order to avoid problems in customizing and/or configuring it into a situation, which is based on different values and beliefs.

Also the scope an artifact covers has an influence upon our conception about whether an artifact is an ISD method or not. In what follows, we consider the coverage of an artifact in terms of (a) ISD domains, (b) ISD perspectives, (c) model levels, and (d) ISD management vs. ISD execution.

ISD methods differ from one another in the emphasis they put on ISD deliverables, on one hand, and ISD actions, on the other hand (Wijers 1991; Vlasblom *et al.* 1995, 597). We argue that to be acknowledged as an ISD method, an artifact must address both ISD deliverables and ISD actions. For instance, UML in Booch *et al.* (1999) is not an ISD method but a language providing the syntax and part of the semantics to present the ISD deliverables. Likewise, an artifact only providing procedures and rules for ISD actions and no language(s) to present results with it not an ISD method. ISD purposes, either related to ISD actions and ISD deliverables or characterizing an ISD context as a whole, should also be addressed in an ISD method. In contrast, it is not necessitated that an artifact should describe/prescribe how to compose ISD actions into ISD roles and ISD positions.

Russo *et al.* (1996, 387) define an ISD method to be "a systematic approach conducting at least one complete phase" of the ISD. In our view, the support to only one phase or workflow is not enough to regard an artifact as an ISD method. An artifact may be an analysis method or a design method, but not an ISD method. We require that an artifact covers an essential portion of ISD workflows or ISD phases to be seen as an ISD method. That portion is definitely more than one phase or one workflow. To put this more generally, we state that an ISD method must provide support for ISD within three ISD perspectives: the ISD systelogical perspective, the ISD infological perspective and the ISD conceptual perspective.

An artifact merely consisting of work instructions and examples of descriptions about outcomes is lacking essential components, namely the coherent and consistent concepts with which an ISD context as well as an IS context can be conceived, understood, structured and presented. Hence, it is not enough that an ISD method provides descriptions / prescriptions on the type model level, and possibly at the instance model level. It is necessary that an ISD method also provide descriptions on the meta model level, and preferably on the meta meta model level.

Project management is mentioned in only few definitions of the ISD method (e.g. Avison *et al.* (1995a, 418); Graham *et al.* (1997, 2)). One reason for this is that many ISD methods avoid presenting rules and guidelines for management of an ISD effort, and instead rely on the support of some existing generic methodological framework of project management. In our view, this is quite possible. However, exploiting two different methodical sources, perhaps established with divergent assumptions and concepts, may result in problems in the integration of execution actions and management actions.

In conclusion, we argue that to be regarded as an ISD method, an artifact has to provide knowledge about its history, domains and ways of applying it, as well as about fundamental assumptions underlying it. It must offer informal and semi-formal languages for communication, presentation and reasoning. Its conceptual foundation should cover at least three ISD domains (i.e. the ISD purpose domain, the ISD actor domain, and the ISD actions domain), three ISD perspectives (i.e. the ISD systelogical perspective, the ISD infological perspective, and the ISD conceptual perspective) and two model levels (i.e. the meta model level and the type model level).

9.7 Comparative Analysis of ISD Artifacts

In this section we present a comparative analysis of those ISD artifacts in the literature that are comparable with our ISD method ontology in terms of aims and comprehensiveness. Our purpose here is to discover what kinds of principles the artifacts use to structure the ISD methods, what main parts the ISD methods are seen to contain, and what kinds of atomic elements the ISD methods are seen to be composed of.

There is a huge amount of literature on ISD methods. Surprisingly, there are only a few presentations that are named as method ontologies or methodology ontologies (e.g. Chandrasekaran *et al.* 1998; Lin *et al.* 1999; Fensel *et al.* 2003). These appear, however, to be very far, in terms of aims and domains, from what we are interested in here. Therefore we have excluded them from this analysis. We recognized two groups of artifacts to be relevant for our analysis. The first group comprises frameworks for comparing and evaluating ISD methods. We are interested in these frameworks because it is reasonable to expect that the frameworks established for the comparison of the ISD methods are comprehensive. To this first group we have selected the socio-cybernetic framework for "the feature analysis of the ISD methods" by Iivari *et al.* (1983), the framework "for comparing methods" by Avison *et al.* (1995a) and the "cataloguing" framework for software development methods by Karam *et al.* (1993). All these frameworks are quite comprehensive. The second group is composed of frameworks which aim at categorizing method knowledge. To this group we have selected the "anatomy" of the method by Lyytinen (1986), the reference model for information systems development by Heym *et al.* (1992a), the "architecture" of the method by Vlasblom *et al.* (1995), and the shell model of method knowledge by Tolvanen (1998).

There are, of course, many other artifacts (e.g. Wijers 1991; Saeki *et al.* 1993; Vasconcelos *et al.* 1998; Gupta *et al.* 2001; Tun *et al.* 2003) that provide structural descriptions of methods. They are not, however, comprehensive, and so we have to ignore them here. Next, we will give short descriptions of the selected artifacts and then compare them with our ISD method ontology. The summary of the results from our analysis is presented in Table 30.

TABLE 30 Summary of the comparative review of the literature on the ISD methods

Reference	**Main structuring principles**	**Main parts**	**Elements / features**
Iivari *et al.* (1983)	Based on a sociocybernetic interpretation of ISD	- Information system - ISD decision making and learning dynamics - ISD actions	Concept structures, languages, substantive assumptions. Life cycle dynamics, main-phase dynamics (phase structure), learning dynamics. Decomposition and control structures of ISD actions, a list of specific ISD activities, features of ISD actions categorized by three "perspectives" (P, I/O, C/O).
Avison *et al.* (1995a)	No specific principle is mentioned		Philosophy (paradigm, objectives, domain, target), models, techniques and tools, scope, outputs, practice (e.g. background, user base, participants), product.
Karam *et al.* (1993)	No specific principle is mentioned	- Technical properties - Managerial properties - Usage properties	Life cycle model, philosophy, work products, notations, procedures, guidelines, criteria and measures, verification, method specialization. Software development organization, ease of integration. Application area, information system, software tools, instructions, user base.
Lyytinen (1986)	No specific principle is mentioned	- Paradigm component - Normative component - Resource component	Ontology, epistemology, rationality, metarules, theories, and exemplars; Organizing prescriptions (choice directives, sequencing directives, change norms, performance prescriptions), and working norms. People, money, tools, computer systems, physical locations, etc.
Heym *et al.* (1992a)	Distinguishes between how to accomplish ISD work and how to apply the method	- Methodology object - Guideline object	Method, technique, deliverable model objects, representation components, perspectives/views, dependencies, deliverable flows, deliverable usages, process models objects. Experience objects, integrity objects.
Vlasblom *et al.* (1995)	'Architecture paradigm'		Development strategy, products, activities, disciplines, techniques (incl. representation practices and working practices), tools.
Tolvanen (1998)	The shell model in which each level complements the next lower level		Conceptual structure, notation, process, participation and roles, development objectives and decisions, values and assumptions.

Iivari *et al.* (1983) divide the features of ISD methods into three parts: those concerning the information system[153], those related to ISD dynamics, and those related to ISD actions. The first category contains semantic and syntactic features (i.e. conceptual structures of IS models and notations used to present the models), as well as implicit and explicit substantive assumptions andrecommendations concerning the application area of the method. The second category comprises features related to decision-making (i.e. life-cycles and main phases) and learning dynamics of ISD assumed by the method. The third category features ISD actions from the viewpoints of decomposition and control structures and addresses some specific ISD actions (e.g. group analysis, goal analysis, situation analysis) and ISD perspectives. Iivari *et al.* (1983) distinguish between three ISD perspectives, namely P (pragmatics), I/O (input/output) and C/O (construction/operation) perspectives.

Avison *et al.* (1995a) propose a framework for comparing ISD methods, based on the earlier works of Wood-Harper *et al.* (1982) and Fitzgerald *et al.* (1985). The framework contains seven basic elements: philosophy, models, techniques and tools, scope, outputs, practice, and product. The philosophy element contains a paradigm, objectives, a domain and a target (applicability). The domain means situations that the method addresses. The target is seen as the applicability of the method to e.g. particular types of problems, environments, or organizations. The scope means the stages the method covers, and the outputs stand for the deliverables from those stages. The practice element is composed of the method background, the prior and existing users of the method, the participant roles in the method and the skill levels required for them, the experiences from the method use, and the degrees to which the method has been customized in prior projects. The last element of the framework, referred to as the product, means the forms in which the method appears and, if commercial, what purchasers actually get for their money.

Karam *et al.* (1993) present a cataloguing framework for software development methods. The framework lists 21 properties grouped into three categories: the technical properties, the managerial properties, and the usage properties. The technical properties concern e.g. a life cycle model (e.g. ISD approach, phase structure), the governing philosophy (e.g. structural), work products and notations, procedures, guidelines, criteria and measures, verification, degree of formality, and method specialization. The managerial properties are related to the software development organization, and ease of integration. The usage properties address the type of the application area, the size of the information system, the availability of software tools, the ease of instructions, as well as the extent and variety of the method's user base.

Lyytinen (1986) presents an "anatomy" of the method that is composed of three main components: the paradigm component, the normative component, and the resource component. The paradigm component suggests something about the purpose, environments and contents of change and the basic ways of carrying out the change. It contains six sub-components: ontology,

153 Iivari *et al.* (1983) use the term 'data system'.

epistemology, rationality, metarules, theories, and exemplars. The normative component comprises organizing prescriptions and working norms. The working norms can be choice directives, sequencing directives, change norms consisting of grammar of a developer's language and procedural directives, and performance prescriptions. The resource component helps identify resources (people, money, tools, computer systems, physical locations, physical communication media etc.) and keep track of their consumption.

Heym *et al.* (1992a) present a methodology[154] reference model for information systems development as some kind of semantic data model. According to this model, the methodology (knowledge) is composed of two kinds of objects: methodology objects and guideline objects. The methodology objects are methods and techniques (as the components of the methodologies), deliverable model objects, representation components, perspectives/views, dependencies, deliverable flows, deliverable usages, and process model objects. Because the method cannot be taken off the shelf and used as such in a project, it is necessary to preserve the development knowledge of organizations as guidelines. The guideline objects are either experience objects or integrity objects. Through the guidelines that can be connected to any methodology objects, the systems developer can store or search for experiences about certain activities or deliverables, collected in other projects and by other people. The guideline objects are presented in rules, notices, conditions, and/or conclusions.

Vlasblom *et al.* (1995) propose a three-level description of a method. The highest level, the generic level, is composed of building blocks for the various elements of the method. The building blocks form a kind of architecture of the method. The building blocks are ISD activities, ISD products, a development strategy (i.e. philosophy relating to the manner in which the activities and the products are arranged in time), techniques (representation and working practices), tools, and disciplines (particular areas of expertise). The disciplines mean those who play a role in the ISD activities and those who will use the products of the ISD.

Tolvanen (1998, 35-37) presents a categorization of the types of method knowledge in the form of a shell model in which each type of knowledge complements the others and all are required to yield a "complete" method. The core of the shell stands for the concepts and constructs used in modeling techniques. The next level contains the notations that are used to present models. Processes on the next upper level are based on the concepts and they prescribe how models are created and manipulated. Three outer levels stand for participation and roles, development objectives and decisions, and values and assumptions, respectively.

Next, we compare the artifacts to one another and to our ISD method ontology, first on a general level and then individually for each methodical view (see Table 30). Principles used to structure the parts of the method vary greatly. Iivari *et al.* (1983) make the main division between features of the IS and

154 Note that a methodology here means the description or representation of different methods (Heym *et al.* 1992a, 215).

the ISD, and Karam *et al.* (1993) distinguish between the ISD execution, the ISD management, and the usage. Heym *et al.* (1992a) categorize the knowledge objects into two parts depending on whether the knowledge is related to the methodology or its use (cf. experience objects). Tolvanen (1998) divides the method knowledge on the basis of how close to the "core knowledge" of ISD they are. In our ISD method ontology, the main structure is based on the methodical views that guide to recognize generic features, history, application, conceptual contents, languages, physical appearance and main components of the method. We argue that this way of categorization results in more sound, orthogonal and distinguishable structures of parts than in the reviewed artifacts. This argument is supported by the fact that the methodical views, underlying the ISD method ontology, are largely based on the theoretical framework of the semantic ladder (Stamper 1973; Stamper 1996).

The historical view brings out features of and experience from the prior ME contexts and the prior ISD contexts. No analyzed artifact addresses the prior ME contexts. The prior ISD contexts are considered in Iivari *et al.* (1983) ('empirical support to e.g. languages'), Avison *et al.* (1995a) ('practice'), Karam *et al.* (1993) ('user base'), and Heym *et al.* (1992a) ('experience objects'). The application view concerns the target ISD contexts and the target ME contexts. The target ISD context, in the sense of application area, is considered in Avison *et al.* (1995a) ('domain'), Karam *et al.* (1993) ('application area'), and Heym *et al.* (1992a) ('application type perspective'). But nothing is said about the target ME contexts in any artifact.

The generic view addresses the paradigms, ISD approaches, and main ISD principles underlying the method. Paradigms are recognized in Avison *et al.* (1995a), Lyytinen (1986) and Tolvanen (1998). ISD approaches are included in the frameworks by Iivari *et al.* (1983), Karam *et al.* (1993) and Vlasblom *et al.* (1995). The representation view is addressed in Iivari *et al.* (1983) and Vlasblom *et al.* (1995) on a general level, and in Karam *et al.* (1993), Lyytinen (1986), Heym *et al.* (1992a) and Tolvanen (1998) on a more detailed level. The method as a physical entity is recognized in Avison *et al.* (1995a) (cf. 'product') and partly in Karam *et al.* (1993) (cf. 'software tools, instructions') and Vlasblom *et al.* (1995) (cf. 'tools').

The ISD models and ISD techniques are included in all the analyzed artifacts, except in Lyytinen (1986) and Tolvanen (1998). The ISD perspectives are only addressed in Iivari *et al.* (1983), but on a very general level only.The partition into the ISD contextual models is most clearly visible in the shell model of Tolvanen (1998) (cf. levels of process, participation and roles, and objectives). In the other artifacts, only the ISD deliverables (products) and the ISD processes are typically distinguished. Some of the artifacts (Iivari *et al.* 1983; Vlasblom *et al.* 1995; Lyytinen 1986) consider relationships between the ISD actions. There are also some artifacts that recognize, although in a not-so-explicit way, issues in the ISD purpose domain (Avison *et al.* 1995a; Iivari *et al.* 1983) and in the ISD actor domain (Iivari *et al.* 1983; Avison *et al.* 1995a;

Lyytinen 1986). The framework of Iivari *et al.* (1983) addresses the IS meta models indirectly through the consideration of concepts and languages.

In conclusion, an ISD method ontology has to be comprehensive and well-structured to suit the evaluation, comparison, and engineering of ISD methods. We have strived for a reasonable level of comprehensiveness by applying the well-grounded and well-defined methodical views. The comparative analysis of the existing artifacts shows that our ontology is more comprehensive than any of them. Comprehensiveness in this case does not mean the number of the concepts but the degree to which the artifact covers the features that are significant to distinguishing the meanings of things in the ISD and the ISD method. Our way of structuring the parts and features of the ISD method also makes the ISD method ontology more explicit and easier to apply.

9.8 ISD Method Component

In this section we continue the discussion about the method components. The ISD models and the ISD techniques are method components, as we found in Section 9.5, but actually there are many parts in the ISD method that can be considered to be method components. Here, we first define the notion of an ISD method component and present some classifications. Second, we define the granularity levels of the components, and consider the interface of the method component. Third, we give examples of the method components and discuss their integration on a general level. Fourth, we make a comparative analysis of conceptions of the method components presented in the literature. The section ends with a summary.

9.8.1 Definition of the ISD Method Component

Reuse is an essential objective towards which software engineering has strived for several decades. The most effective means to achieve this objective has been considered to be the construction of compatible components or modules that are general enough for reuse. Originally, these compatible components were code components. This reuse view is best reflected by the definition by Kruchten (2000): "A component is a nontrivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces" (ibid p. 274)

In recent years the component-based paradigm has been extended to cover the construction and reuse of design components as well. Examples of the design components are domain models (e.g. Arango *et al.* 1991; Arango 1994), design patterns (Gamma *et al.* 1995) and application frameworks (Fayad *et al.* 1997; Carey *et al.* 2002). In the domain models the ontological descriptions specific to certain domains, such as insurance, transportation, banking etc., can be used as the fundamentals on which the applications can be designed and implemented. The design patterns provide structured descriptions of proved solutions to commonly appearing problems. The application frameworks provide platforms for integrating reusable components. The framework itself is a large component that can be extended and configured, resulting in a functioning application in a given problem domain.

The component-based paradigm has also been proposed as a means of constructing ISD methods. A reusable part of the method is called a method component (e.g. Kumar *et al.* 1992[155]; Song *et al.* 1992; Gupta *et al.* 2001; Zhang *et al.* 2001), a method fragment (e.g. Harmsen 1997; Nuseibeh *et al.* 1996)), a building block (Vlasblom *et al.* 1995), and a task package (Hidding *et al.* 1993). To be faithful, also literarily, to the component-based paradigm, we prefer the term 'ISD method component' and define it as follows. An *ISD method component* means a well-defined part of the ISD method that can be integrated to other ISD method components to form a meaningful, coherent and consistent ISD method. The ISD method component is reusable if it is specifically developed for reuse (cf. Zhang *et al.* 2001, 117).

9.8.2 Classifications of Method Components

Based on four contextual ontologies (i.e. the context ontology, the layer ontology, the model level ontology, and the perspective ontology) and the principle of decomposition we establish a five-dimensional scheme by which the method components can be classified (see Figure 96). We argue that a component can be placed in any position along these five (nearly orthogonal) dimensions. Next, we define the dimensions and illustrate the use of them with examples.

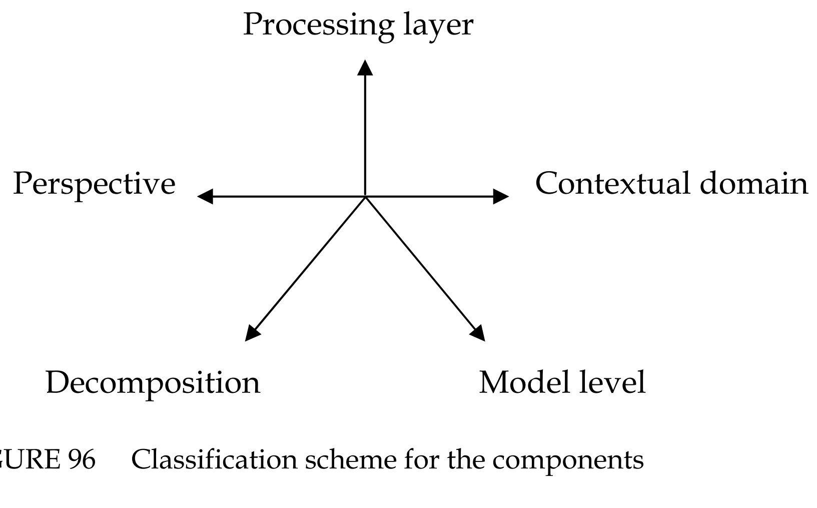


FIGURE 96 Classification scheme for the components

Contextual domains
The classification is here based on those contextual concepts that the component contains. We distinguish between purpose components, actor components,

[155] Strictly speaking Kumar *et al.* (1992, 262) use the term 'methodology component'.

action components, and so on. For instance, a certain kind of organizational structure (e.g. matrix-like structure), a step-by-step procedure for a specific ISD action (e.g. normalization procedure), and a genre-based classification of ISD deliverables are method components belonging to the ISD actor domain, the ISD action domain, and the ISD object domain, respectively.

Processing layer
The classification of method components is here based on the processing layer, on which the component is to be reused. Components fabricated and reused in software engineering (e.g. code components, domain models, and application frameworks) reside at the ISD layer. They are referred to as IS components. Method components (e.g. data flow diagram, normalization technique) are engineered and reused at the ME layer. Further, at the RW (research work) layer components are reused to engineer methods for method engineering. Some of the components are general-purpose in a sense that they can be reused at several processing layers. For instance, a component specifying the goal / means structure can be reused as the basis of the IS goal meta model, the ISD goal meta model, and the ME goal meta model. Typical examples of general-purpose components are abstraction structures defined in Section 3.9.

Model level
The components are here considered models that are classified according to which model level they belong to. A code component, for instance, is on the type model level, and so is the normalization technique as well. The ER model specifying the allowed concepts and constructs for ER schemas is on the meta level. Some components may contain parts on more than one level. A data flow diagramming technique, for instance, is a component that comprises a part that prescribes ISD actions and ISD deliverables (i.e. the type model level), and another part that specifies the concepts and constructs allowed in the data flow diagram (i.e. the meta model level).

Perspective
The classification is here based on the perspective, through which method components view the context(s) concerned. Consequently, we can have systelogical, infological, conceptual, datalogical and physical components. The meaning and contents of the component depend on what layer the component is situated. At the ISD layer, for instance, a technique of conceptual modeling presented in a data flow diagram is an IS infological component, because it is described through concepts of ISD actions and ISD deliverables. Conceptual components, like the domain models (Arango *et al.* 1991; Arango 1994), provide ontological constructs for structuring the contents of the ISD deliverables. A method integration technique presented in a data flow diagram, in turn, is an example of the infological component at the ME layer.

Decomposition
Components can be situated on various granularity levels. At one extreme of this dimension is a method as a whole. At the opposite extreme we could see an individual concept. A detailed discussion about the proper granularity levels of method components at the ISD layer is given in the next section.

9.8.3 Granularity Levels of Method Components

Any part of the method at any level of detail could be, in principle, considered a method component. In practice, however, this is not the case. According to the definition of the method component given above, it is required that it should be integratable to another method component. In addition, it is required that a totality constructed by integration constitutes something consistent, coherent, and meaningful to ISD contexts. To further the fulfillment of these requirements, we define three granularity levels for the method components. The levels are: the level of contextual ISD components, the level of domain-based ISD components, and the level of construct components. In what follows, we define these with examples.

A *contextual ISD method component* is a method component that contains descriptions/prescriptions of features of the ISD within several contextual domains. An example of a contextual ISD method component is the use case technique (Jacobson *et al.* 1999) that is aimed at guiding IS developers, in a step-by-step manner, in their specifying services the IS clients expect from the IS. The specification of the component requires the use of concepts of at least four ISD domains, i.e. the purpose domain, the actor domain, the action domain, and the object domain. At the extreme case, a contextual method component is an ISD method itself.

A *domain-based ISD method component* is a method component that contains descriptions/prescriptions of features of the ISD within one or at most two contextual domains. We can distinguish between several domain-based ISD method components. *Ontological components* are method components which provide concepts and constructs for conceptual modeling. Meta models (e.g. the meta data model of the ER model (Chen 1976)) are typical examples of this kind of components. *Notational components* are method components which provide sets of symbols (without any predefined semantics). For instance, the graphical notation of the ER model is a notational component.

While ontological components and notational components merely concern the conceptual contents and representation of ISD deliverables, respectively, *action-based components* mainly concern ISD actions. These are also known as process fragments (Harmsen 1997). An example of an action-based component is that part of the ER technique (Batini *et al.* 1992), which provides stepwise instructions for e.g. identifying entity types, relationship types, and attributes. The ER technique prescribes the predefined order (not necessarily a temporal order) for actions and input/output relationships between ISD actions and ISD deliverables. Another example of an action-based component is a transformation technique by which an ER schema can be transformed into a relational schema. This is called a 'transformational method' in the terminology of Prakash (1997, 1999).

Furthermore, we can distinguish between actor-based components and tool-based components. *Actor-based components* are method components which contain concepts and constructs to specify, for instance, an organisational structure. *Tool-based components* are method components which offer concepts and constructs to describe elements and architecture of a computerized information system. These components correspond to technical fragments in Harmsen (1997).

The third granularity level is the level of construct components. A *construct component* is a method component which cannot be decomposed into smaller parts without loosing some of its meaningfulness and integratability. In many studies (e.g. Harmsen 1997) individual concepts are regarded as atomic parts. In our opinion, individual concepts cannot be real method components for the same reason as no individual row of code can be regarded as a component.

The integratability of method components depends on the kinds of interfaces the components have. In the next section we discuss the notion of an interface in general and define the notion of a contextual interface.

9.8.4 Interface of the Method Component

In software engineering a reusable component must have a well-defined interface[156]. An interface shows the services the component provides for the other components and the services it demands from the other components. In the object-oriented paradigm the services are specified through operation signatures with parameters. Converting this directly into the contextual viewpoint would mean that an interface is specified by concepts of the action domain (cf. operation call) and the object domain (cf. parameters). When the component-based paradigm has been applied at the ME layer, this kind of conception has survived (cf. Ralyte *et al.* 2003). Consequently, it is common to think that two method components can be integrated if one component receives a specific piece of data from the other component and conducts the next action for the data in the pre-defined order (cf. Kinnunen *et al.* 1996). This view is illustrated in Figure 97 below. The method components A and B can be integrated, if the outgoing interface of the component A and the incoming interface of the component B match. The interfaces in this case consist of two parts: action part (unbroken line) and object part (dotted line). Although this kind of conception is adequate at the ISD layer and for technical artifacts, in

[156] D'Souza and Wills (1999), for instance, state that a code component is " a coherent package of software implementation that (a) can be independently developed and delivered, (b) has explicit and well-defined interfaces for the services it provides, (c) has explicit and well-specified interfaces for services it expects from others, and (d) can be composed with other components, perhaps customizing some of their properties, without modifying the component themselves" (ibid p. 387).

Component A
Component B

FIGURE 97 Simple interfaces of two method components

method engineering the method components are much more multifaceted requiring a more elaborated notion of an interface.

According to the contextual approach there are seven domains. We argue that each of them may be of importance to revealing the real nature and meaning of the interface of the method component. Therefore, we define: a *contextual interface* of a method component is a white-box like description of those contextual relationships through which a method component can be integrated into other method components. The contextual relationships are inter-domain relationships and/or intra-domain relationships (see Chapters 4 and 8). Figure 98 below illustrates the contextual interface of a method component. The component C has an interface that is composed of seven 'threads'. Each of them specifies an important contextual relationship by which concepts of the component should be connected to concepts of another method component.

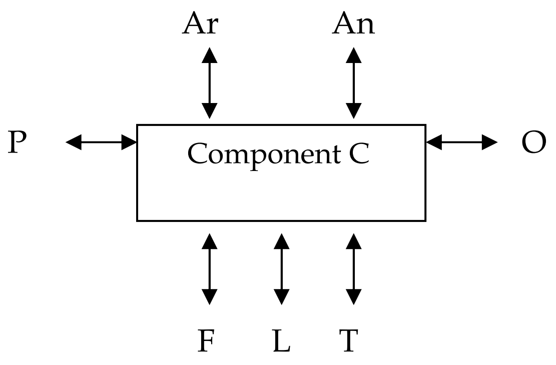


FIGURE 98 Contextual interface of the method component (P = purpose, Ar = Actor, An = Action, O = Object, F = Facility, L = Location, T = Time)

To aid the comparison of method components and the identification of the most suitable component for the integration at hand, it is possible to define attributes for components. There are two approaches for this. In the first approach (Castano *et al.* 1993; Ralyte *et al.* 2001) measures are defined to enable the measurement of the similarity or closeness of the concepts and constructs in the method components. For example, Ralyte *et al.* (2001) define semantic and structural measures for elements of the product models (i.e. IS meta models), and semantic affinity of intentions for the process models (i.e. ISD action models). We argue that similar measures can be defined for all the contextual parts of the interface of the method components.

Another approach is to define contingency factors or properties for method components (van Slooten *et al.* 1993; Vlasblom *et al.* 1995; Harmsen 1997). Harmsen (1997), for instance, defines a large set of property types of method fragments. Property types are categorized into fragment aspects and scenario aspects. Fragment aspects are partitioned into several groups: e.g. general properties (e.g. granularity level, goal, focus, maturity level), property types of process fragments (process type, capability maturity level), and property types of product fragments (e.g. level of detail, temporal state, abstraction level, representation). Scenario aspects are subdivided into general IS modeling aspects, aspects related to the user of the IS, aspects related to engineering strategies, and aspects related to IS engineering management. Examples of the scenario aspects are modeling aspect, modeling scope, project goal, approach orientation, validation type, degree of participation, degree of user responsibility, and iteration type.

The contextual view on the interface can help the specification and use of properties of method components in many ways. As seen from the short abstract of Harmsen (1997), a set of relevant properties of a method component can be very large. Some of the properties concern the ISD context as a whole (e.g. project goal, approach orientation) while others address characteristics of specifics in one or two contextual domains (e.g. process type, goal, level of detail, representation). Structuring the contingencies and properties of method components according to the contextual view of the interface helps in their definition and use in method integration. We will illustrate this with examples in the next section.

9.8.5 Examples of ISD Method Components

In this section we present examples of ISD method components. Our aim is to illustrate the notions of method component and component interface and to show the importance of the contextual approach to the integration of method components. We consider three examples, two of which are modeling techniques and one that is a description model. The modelling techniques are the use case technique and the sequence diagramming technique (Jacobson *et al.* 1999). The description model is the goal model (cf. Lee *et al.* 2001)[157]. These ISD method components are selected because they are commonly known and suitable to integration. Next, we first model the ISD method components and define their concepts. After that, we discuss the nature and properties of the ISD method components according to our classifications. Finally, we consider the issues to be faced when trying to integrate the ISD method components.

The *use case technique* is "a systematic and intuitive way to capture the functional requirements with particular focus on the value added to each individual user or to each external system" (Jacobson *et al.* 1999, 131). Because the technique contains the specification of the description model and prescriptions for how to make an instance of the model, the model of the technique consists of ISD models at two levels (see Figure 99). The upper ISD model prescribes the context in which the IS model is to be produced. It is

157 We apply the most basic concepts of the model by Lee *et al.* (2001) and call them the goal model.

presented in a data flow diagram extended with ISD actors. The other ISD models, called the IS meta data models, describe the concepts and constructs with which the use case model is created and presented in UML. In Figure 99 the IS meta data model on the left side specifies the conceptual contents of a use case diagram. The IS meta data model on the right side specifies the conceptual contents of structured descriptions of use cases.

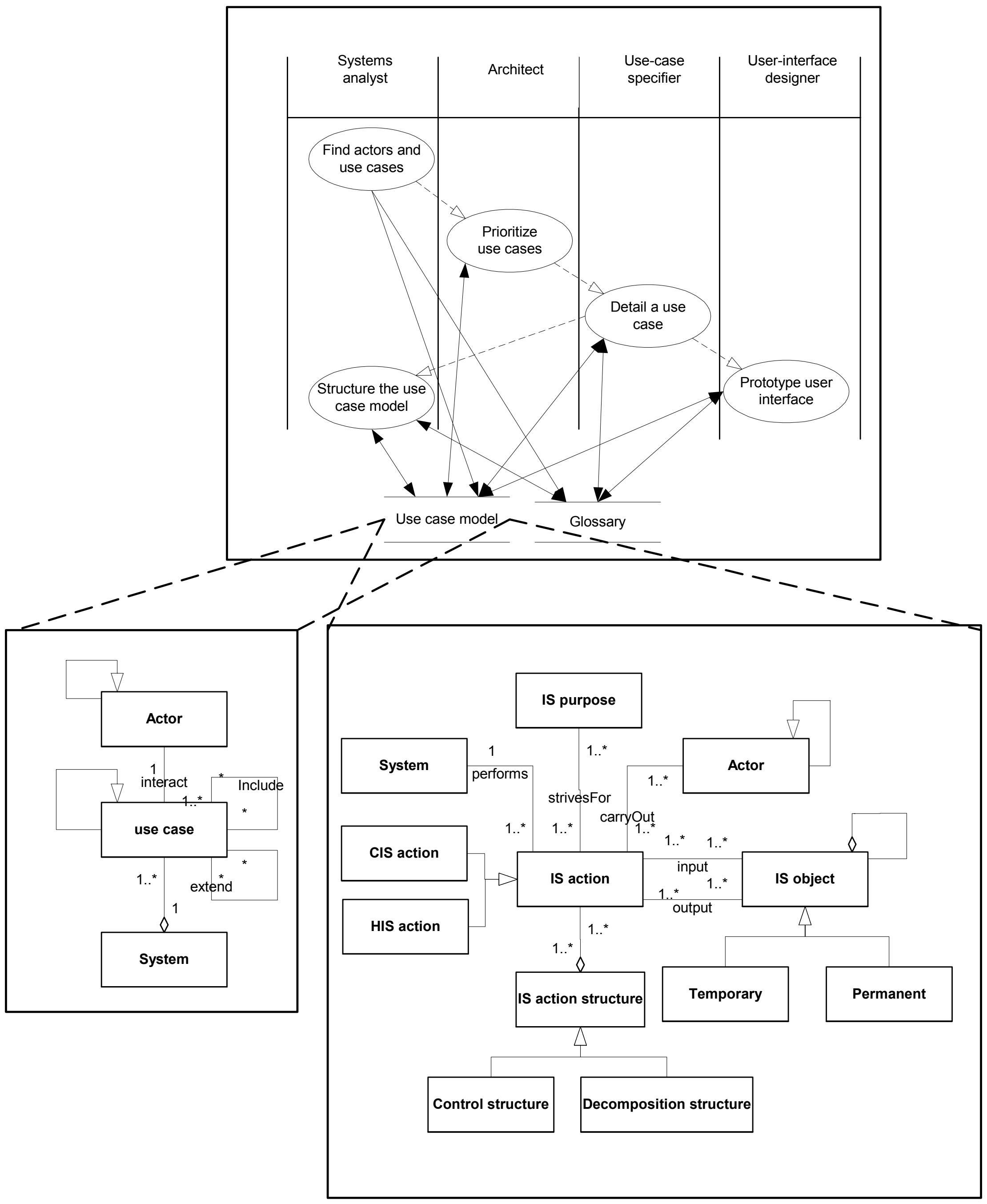


FIGURE 99 ISD models of the use case technique

Requirements capture with the use case technique produces two kinds of ISD deliverables: the glossary and the use case model. A *glossary* defines common terms used to describe a system. A *use-case model* is a "model of a system containing actors and use cases and their relationships"(Jacobson *et al.* 1999, 133). The model is composed of two parts: a use case diagram and use case descriptions. Next, we define the concepts and constructs used in the use case diagram.

An *actor* is "a type of user (of the system) or an external system, device etc. associated to the system. Actors represent parties outside the system that collaborate with the system" (Jacobson *et al.* 1999, 134). An actor plays one role for each use case with which it collaborates. Actors may have generalization relationships with one another indicating that a child actor can play the same role(s) as the parent actor. A *use case* specifies a sequence of actions (i.e. IS actions), including alternatives to the sequence that the system can perform when interacting with actors of the system. The *system* may include one or more use cases. A use case may be related to other use cases by generalization, include and extend relationships. The *generalization relationship* between two use cases means that the child use case inherits the behavior and features of the parent use case and may add new features. The *include relationship* signifies that the base use case contains the behavior of the addition use case. The *extend relationship* implies that the extension use case extends the behavior described in the base use case under certain conditions.

For each use case depicted in the use case diagram, a description in structured English is given. This description reveals, in more detail, goals (i.e. IS purpose) for which the system functions, as well as information services (i.e. IS objects) the system processes and provides for actors. Some of the IS objects may be temporary while the others are permanent. The description may also distinguish between the actions that are performed by the system (i.e. the CIS actions) and the actions that are carried out by human actors (i.e. the HIS actions).

The use case modeling is decomposed into five activities (Jacobson *et al.* 1999): find actors and use cases, prioritize use cases, detail a use case, structure the use case model, and prototype user-interface. Activities and deliverables as well as control and information flows between them are illustrated in the upper part of Figure 99. In the figure the ISD actors (workers in Jacobson *et al.* (1999)) responsible for the activities are also presented.

A *sequence diagram* describes interaction between the system and its actors, as well as interaction between the parts of the system. An (human) actor interacts with the system by manipulating and/or reading interface objects. Interaction between the parts of the system occurs through sending and receiving messages. A sequence diagram emphasizes logical or temporal ordering of messages (Booch *et al.* 1999). Graphically, a sequence diagram is like a table: it shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis. Next, we define the most essential concepts of the sequence diagram (Figure 100).

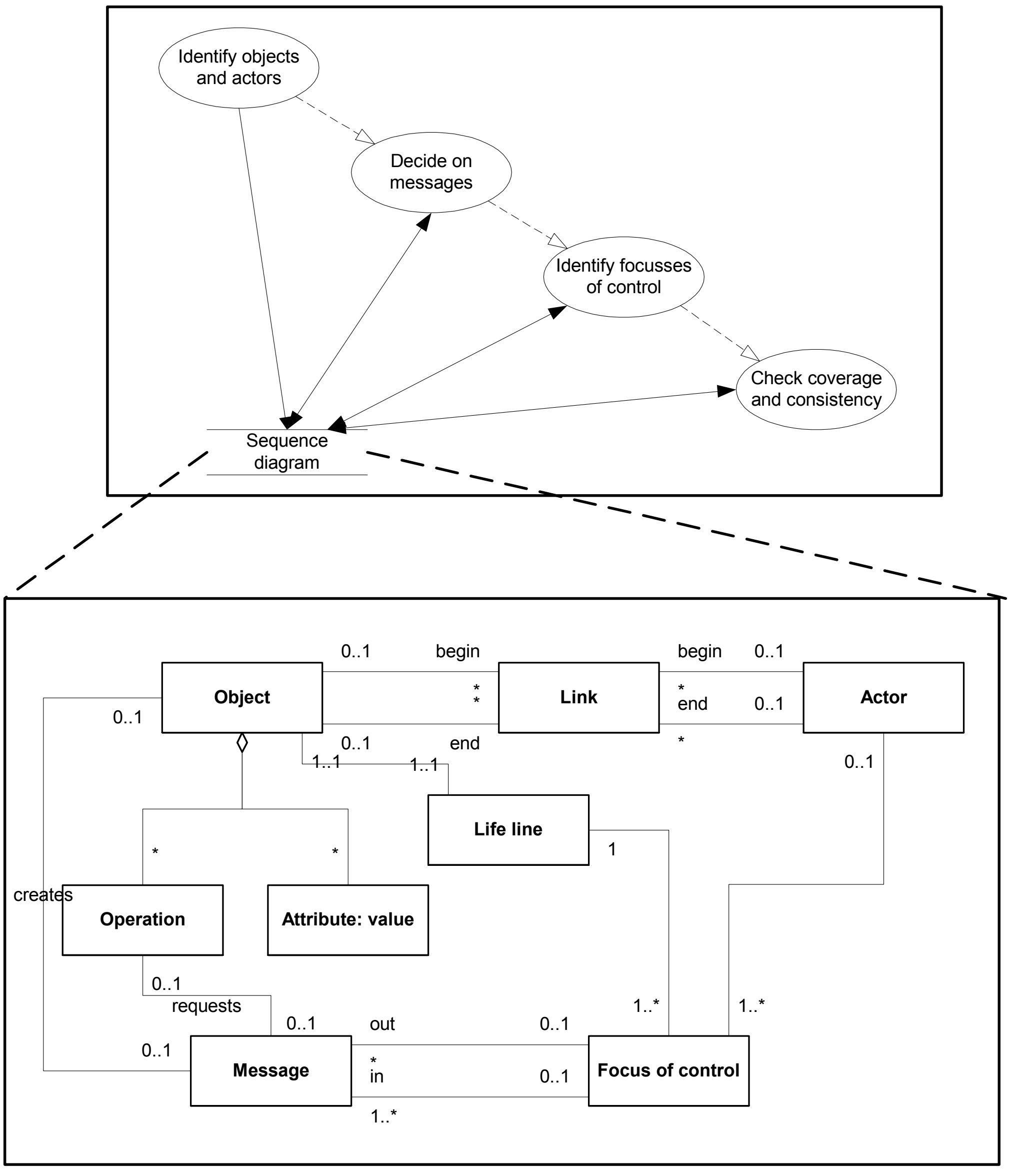


FIGURE 100 ISD models of the sequence diagramming technique

The parts of the system are called objects. An *object*[158] is "an entity with a well-defined boundary and identity that encapsulates state and behavior" (Booch *et al.* 1999, 464). A state is a composition of values of the attributes of the object. An *attribute* is "a named property [..] that describes a range of values that instances of the property may hold" (Booch *et al.* 1999, 458). Behavior results from the execution of operations of the object. An *operation* is "the implementation of a service that can be requested from an object of the class in order to affect behavior" (Booch *et al.* 1999, 464). Compared to our terminology, an object is a manifestation of both IS object(s) (cf. attribute) and IS action(s) (cf. operation). A *message* is "a specification of an interaction between objects that

158 Note that 'object' in the object-oriented paradigm is totally different from 'object' in our terminology.

conveys information with the expectation that activity will ensue" (Booch *et al.* 1999, 463). A message carries information in the form of action(s) requested and data transmitted from the sender to the receiver. If the message is stereotyped as 'create', the receiving object is created. That means the beginning of the *life line* of the object. The life line ends with the object receiving the message stereotyped as 'destroy'. The *focus of control* shows the period of time in the life line during which the object is performing the action. The focus of control begins when the object receives the message and ends when it sends the return message. There can be several focuses of control within the life line of the object.

The sequence diagramming proceeds, on a general level, with the following steps (see the upper part in Figure 100). First, identify and present the most essential objects and actors. Then, consider what kinds of interaction there exist between the actors and the objects, as well as between the objects. Recognize messages and their sending orders. The names of the messages reveal actions (signatures of operations) and parameters. For each object, identify focuses of control within the life line. Check the coverage and consistency of the diagram.

The *goal model* is a description model for conceiving, structuring, classifying and representing goals and relationships between them (cf. Lee *et al.* 2001) (Figure 101). A *goal* is a required or desired state of affairs (Koubarakis *et al.* 2000, 144)[159]. The goals are classified into rigid goals and soft goals, as well as into functional goals and non-functional goals (Lee *et al.* 2001, 124-125). A *rigid goal* expresses "a minimum requirement for a target system, which is required to be satisfied utterly". A *soft goal* describes "a desirable property for a target system, and can be satisfied to a degree". A *functional goal* "can be achieved by performing a sequence of operations". A *non-functional goal* "is defined as constraints to qualify its related functional goal". The goals are interrelated through the refinement relationships. *Refinement relationship* establish a goal hierarchy, meaning that a goal can be achieved when the goals below it in the hierarchy are reached.

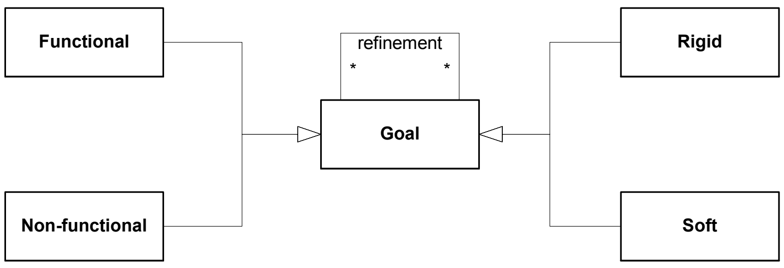


FIGURE 101 Meta model of the goal model

Of the three ISD method components described above, the first two are ISD techniques and the last one is a construct-based component. The ISD techniques

159 A goal is defined as in Koubarakis *et al.* (2000), because Lee *et al.* (2001) do not provide any definition for this generic notion.

are multi-level components because they contain concepts for prescribing ISD actions and ISD deliverables (i.e. the type model level), as well as specify the concepts and constructs (cf. the IS meta data model) allowed in the use case model and in the sequence diagram, respectively. The goal model is a general-purpose construct that can be associated to the method body at any layer. It is also a purpose-based component. The use case technique, as described above, is an ISD datalogical component, because it recognizes, although only at a general level, actors responsible for ISD actions. The sequence diagramming technique is described as an ISD infological component. The goal model can be applied with any ISD perspective.

Now let us assume that there is a need to integrate the three ISD method components in the following way: (a) sequence diagrams are used to define more precisely use cases, and (b) goal models are used to present, more explicitly, goals toward which each use case in the use case diagram aims. How would the process of integration proceed and how do the interfaces of the components affect the integration considerations? To answer the questions we first discuss the integration of two ISD techniques and then consider the integration of the goal model into this method body.

To integrate the ISD method components one should first examine how the purposes of the ISD deliverables involved in the ISD method components match. The use case technique aims to produce general descriptions of actions that the system performs when interacting with actors. The sequence diagrams, in turn, can be used to make detailed descriptions of the internal behavior of the system. Hence, at least from the viewpoint of purposes the integration is reasonable and justified. Second, it is important to consider how the deliverables of the ISD method components match in terms of their presentations. The use cases are presented in easy-to-read diagrams supported by descriptions in structured English. These presentation forms enable the understanding of the deliverables also for the non-IT-experts. The sequence diagrams are presented in a semi-formal form. They are constructed and used by IS analysts and IS designers. The variety of presentation forms used in the ISD method components and their match with the skills and profiles of intended ISD actors strengthen the view that the components are suitable for integration.

Third, it should be investigated how the contents of the deliverables can be related. This investigation is conducted by considering the corresponding IS meta data models. In Figures 99 and 100 we can find counterparts in several IS domains: (a) an actor in the use case model corresponds to an actor in the sequence diagram; (b) a system in the use case technique is decomposed into objects in the sequence diagram; (c) an IS action, or more specifically a CIS action, in the use case technique is realized by actions requested by messages in the sequence diagrams; (d) a CIS action is composed of operations of one or more object; (e) an HIS action in the use case technique can be seen as a pro-action or reaction of an (human) actor in the sequence diagram; (f) an IS object in the use case technique is embedded into attributes of objects (cf. permanent

IS objects) or transmitted by messages (cf. temporary IS objects). Thus, we can conclude that the two ISD method components are highly related to one another also through concepts of several IS contextual domains.

Fourth, it is important to consider how the ISD actions can be integrated. There are many ways to structure and associate the actions of constructing use case models and sequence diagrams. Most commonly the processes are performed partly in parallel. For each use case it is considered whether it is useful to make one or more sequence diagrams. Especially in situations where textual descriptions are written about complex use cases, it is beneficial to sketch in parallel sequence diagrams to find out an order in which events and actions occur in the use case. The identification of objects and actors for sequence diagrams is based on the use case descriptions (see Figure 100). Checking the coverage and consistency of sequence diagrams is carried out by comparing them to the corresponding use case description(s). Working with the sequence diagrams increases the understanding of the textual descriptions of the use cases and may, in turn, cause changes in them.

Fifth, one should consider how ISD actors, with their responsibilities, in the method components should be related. The ISD actors are clearly defined in the use case technique (see Figure 99). In contrast, the sequence diagramming technique does not provide exact specifications of ISD actors. We can, however, assume that those ISD actors are IS analysts and IS designers. Integration of the techniques for the part of ISD actors can now be done, either (a) by including the responsibility of making sequence diagrams in the role of the use-case specifier, or (b) maintaining the roles of IS analyst and IS designer and including the responsibilities of the use case specifier into the role of IS analyst.

The goal model can be easily integrated into the use case technique by enhancing the meta model of the use case diagram (see Figure 99) with the concepts and constructs in the meta model of the goal model (see Figure 101). With relating the concept of a goal, and its sub-concepts and relationship, to the concepts of a use case, the purposes of the use cases are made more explicit. It is possible to further refine the use case diagram by defining one more specialization of the goals, yielding actor-specific goals and system-specific goals. Actor-specific goals are objectives of an actor, and system-specific goals are requirements on services that the system provides. Relating the actor-specific goals to the actors enables to explicitly specify the goals of the actors of the system.

The result of the integration of the three ISD method components is presented in Figure 102. The original boundaries of the meta data models are presented with bold lines. Integration has been done via shared concepts (i.e. an actor) or by defining contextual relationships between the concepts of two meta models. The bold lines represent the interfaces of the method components. The more relationships cross the boundary, the more complicated the way is in which the interface is utilized in the integration. The use case technique and the sequence diagramming technique are integrated through four relationships: partOf (Object, System), partOf (CIS action, Operation), isA (Message,

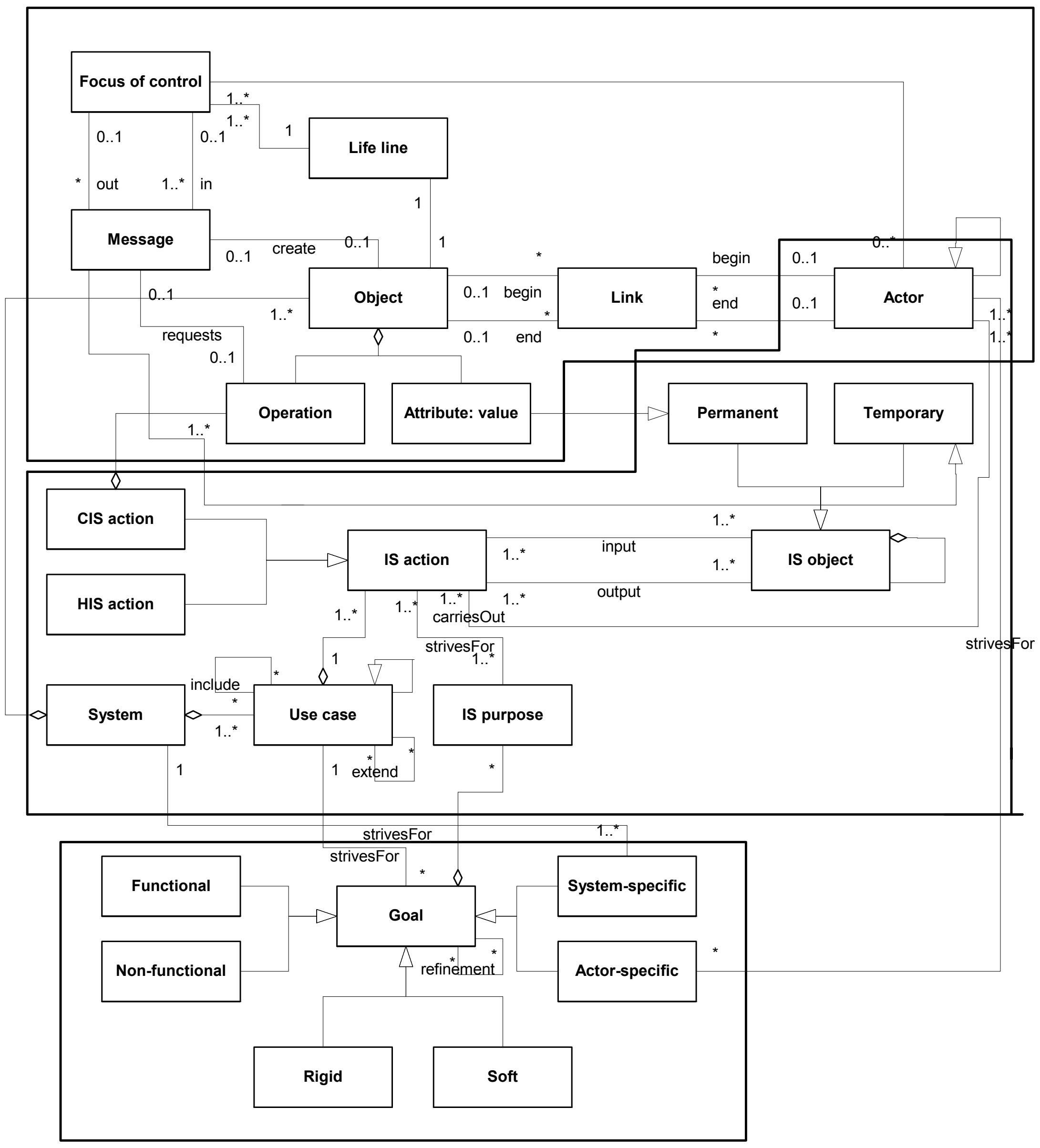


FIGURE 102 Meta data model of the integrated method components

Temporary), and isA (Attribute:value, Permanent). The goal model and the use case technique are integrated through four relationships: strivesFor (System, System-specific), strivesFor (Use case,Goal), strivesFor (Actor, Actor-specific), and partOf (IS purpose, Goal). Hence, the inter-related concepts represent five different contextual domains.

In conclusion of the above consideration of ISD method components and integration we can state the following: The ISD method components can appear in various types, sizes and forms. Small components, like the goal model, can be specified through a simple interface. But the integration of larger ISD method components must be based on the explicit specification and consideration of contextual interfaces. For instance, the ISD techniques considered above extend to two model levels (i.e. type model level and meta model level) and to eight contextual domains (i.e. ISD actor domain, ISD action domain, ISD object domain, IS purpose domain, IS action domain, IS actor domain, IS object domain, and IS facility domain). The common approach in the ISD/ME literature (e.g. Vlasblom *et al.* 1995; Harmsen 1997) to attaching attributes as some kinds of "contextual properties" to the method components does not provide specifications that would be detailed and structured enough to be matched with one another in the integration process. Our way of specifying contextual interfaces provides a natural and well-defined means to structure and realize those "attributes" and thus furthers the right interpretation of the nature, contents and use of the method components.

9.8.6 Comparative Analysis of Conceptions of ISD Method Component

A view of the method as an assembly of components is not a new one. More than ten years ago Kumar and Welke (1992) presented an idea of "combining and structuring selected methodology components (i.e. representation frames and process frames) into an integrated or 'seamless' methodology" (ibid p. 264). Since then the so-called integration approach to method engineering has gained a large popularity in the ME literature (e.g. Kronlöf 1993; van Slooten *et al.* 1993; Vlasblom *et al.* 1995; Ramackers 1994; Ryan *et al.* 1996; Kinnunen *et al.* 1996; Nuseibeh *et al.* 1996; Song 1997; Harmsen 1997; Saeki 1998; Wieringa et al. 1998; Paige 1999; van Hillegersberg *et al.* 1999; Leppänen 2000; Zhang *et al.* 2001; Karlsson *et al.* 2001). The purpose of this section is to shortly describe and compare conceptions of an ISD method component presented in the ME literature. We do this in two parts. First, we analyze presentations using our classification scheme with five dimensions. The purpose of this part is to find out what kinds of ISD method components have been recognized. For this analysis we have selected presentations which apply multiple views on the method component. The presentations are: Harmsen (1997), Zhang *et al.* (2001), Song *et al.* (1992), Gupta *et al.* (2001) and Song (1997). The results from this part are presented in Table 31. Second, we briefly discuss presentations that, though not contributing to the variety of ISD method components, suggest interesting ideas that complete the picture of ISD method components and their integration. This discussion concerns the works of Hidding *et al.* (1993), Nuseibeh *et al.* (1996), and Vlasblom *et al.* (1995).

Harmsen (1997) defines a method fragment to be "a description of an IS engineering method, or any coherent part thereof" (ibid p. 26). A method fragment is coherent if and only if it can be represented by a connected sub-graph of products or processes. He distinguishes between conceptual fragments and technical fragments. A conceptual fragment is "a non-executable fragment, which is described as complete as possible, without taking into account the actor or actor type that will possibly use it" (ibid p. 51). A technical fragment is "a fragment implemented as an IS engineering tool or part thereof"(ibid p. 51). Compared to our scheme, the technical fragments are built on the specifications of the physical view, while the conceptual fragments follow the IS infological, IS conceptual or IS datalogical perspective. Furthermore, Harmsen (1997) distinguishes between process fragments and product fragments. A process

TABLE 31 Summary of the comparative analysis of the notions of method component

Classification scheme	Harmsen (1997)	Zhang *et al.* (2001)	Song *et al.* (1992)	Gupta *et al.* (2001)	Song (1997)
Method component	Method fragment	Method component	Method component	Method component	Method component
Domains: - purpose, actor, action, object, facility, location, time, inter-domain	*Perspective:* - product fragment - process fragment	- data	- action - artifact - concept - representation	- objective (product type, process type) - approach - work procedure	- artifact model - process
Perspectives: - systelogical - infological - conceptual - datalogical - physical - inter-perspective	*Abstraction:* - conceptual fragment - technical fragment	*Levels of abstraction:* - analysis - design - implementation	- problem-domain, - problem-model domain, - solution-model domain, - design-document domain	*Views of method architecture*: - generic view - method view - construction view	
Processing layers: - ISD - ME - RW	- ME layer		- ME layer	- RW layer	- ME layer
Decomposition: - contextual - domain-based - construct-based	*Granularity layer*: - method - stage - model - diagram - concept	*Granularity levels:* - project/method - graph/method/ technique - component unit		- complex component - simple component	*Level:* - high-level component - low-level component
Model level: - type model - meta model - meta meta model		*Information type levels:* - IRD level - IRD definition level - IRD schema level			

fragment is "a description of an activity to be carried out within a method" (ibid p. 52). A product fragment is "a specification of a product delivered and/or required within a method" (ibid p. 52). This division is based on the dimension of contextual domains in our classification scheme. Harmsen (1997) defines five granularity layers[160]: (a) method, (b) stage, (c) model, (d) diagram, and (e) concept. The first layer corresponds to the whole method. A stage "addresses an abstraction level", which stands for an ISD workflow in our ontology. Harmsen (1997) applies the set of stages defined in Olle *et al.* (1988a) (e.g. information systems planning, business analysis, system design, construction design). A model corresponds to a perspective in Olle *et al.* (1988a) (i.e. data oriented, process oriented, behavior oriented), which is loosely comparable to the IS domains in our terminology. A diagram (e.g. class diagram) corresponds to the representation of an aspect of the abstraction level. A concept addresses a concept or an association of the method fragment on the diagram layer, or manipulation defined on it.

Zhang *et al.* (2001) define a design component to mean "any reusable design artifact" (ibid p. 117), and present a classification framework with three dimensions: levels of abstraction, information type levels, and granularity levels. The levels of abstraction correspond to stages in the development process, i.e. analysis, design and implementation. The information type levels are derived from the data levels of the Information Resource Dictionary System (IRDS) framework (ISO 1990). This dimension corresponds to the model levels in our scheme. The granularity levels contain a component unit, a graph, and a project. A component unit is composed of the non-property primary data types of GOPRR (Kelly *et al.* 1996). A graph is a collection of objects, relationships, roles and properties, commonly specifying a technique. A project forms a complete design product, or a plan to produce it. On the model level this means a system development project. On the meta model level, a project establishes a method. Due to its focus on the data at each information type level, the framework of Zhang *et al.* (2001) addresses only the structural features of the method, not the whole method.

Song *et al.* (1992) present the so-called base framework for the identification of method components that are comparable in different methods. The framework is composed of the type framework and the function framework. None of the frameworks is based on explicitly defined dimensions. The type framework presents the method-component type hierarchy, which comprises four top-level types: concept, artifact, representation, and action. A concept corresponds to our generic view on the ISD method. It is "an idea that influences the design of a method" (ibid p. 46) (e.g. problems of software design and application, principles for coping with these problems, guidelines (principles) for designing software and coping with these problems). An artifact means a description of some sort of entity involved in a design process (i.e. programs, diagrams, etc.). A representation stands for a means for describing or specifying design artifacts. An action refers to one or more physical and/or

160 This categorization is also used in Brinkkemper *et al.* (1999) and Karlsson *et al.* (2001).

mental processing steps used in design. Artifacts and actions correspond to ISD deliverables and ISD actions, respectively. Concepts and representations can be interpreted as applying the systelogical and datalogical perspectives, respectively. The function framework contains all those design issues that components address. The framework is based on a life-cycle model, which describes the transformation from the problem-domain to the problem-model domain, and further to the solution-model domain and the design-document domain. These domains correspond to the perspectives in our scheme.

Gupta *et al.* (2001) propose a representation system for a method requirements specification (MRS) as an analogy to a software requirements specification at the ISD layer. A MRS describes what a method meeting the MRS has to offer. It is implementation-independent and based on an abstract meta model. The approach applied uses the three-view architecture of methods (see Prakash 1997; Prakash 1999). The views are: the generic view of a method, the method view, and the "construction view"[161]. These views loosely correspond to our perspectives, but it should be noted that they are applied at the RW (research work) layer!

The generic view is used to produce MRS's. It is divided into two parts, the static part and the dynamic part. According to the static view a method is composed of method blocks. A method block is a pair formed of objective and approach. An objective tells what the block tries to achieve. An approach describes the technique that can be used to achieve the objective. An objective is presented with a pair formed of product type and process type. The dynamic part of the method is composed of generic work procedures containing work elements. Each work element is an objectified relationship between possibility and selection. Possibility identifies the set of procedure elements that can be performed in the work procedure. Selection is a particular choice of the procedure element selected in the work procedure. The method view is presented by the so-called decisional metamodel that is an instantiation of the static part of the generic view. This implies that decision is an instance of method block, purpose is an instance of objective, and p-approach is an instance of approach. The method is seen as a pair formed of purpose and p-approach.

During the method assembly, method components, defined in terms of things, is_composed_of and is_mapped_to relationships, are integrated. Method components are considered as constructs composed of things. There are simple and complex components. Simple components are atomic, whereas complex components can be broken up.

Song (1997) presents a framework for the integration of software engineering methods. The framework contains a method composition model that distinguishes between high-level components and low-level components. High-level components are: artifact models, properties (i.e. desired characteristics of the design artifacts), principles, representations, and processes (i.e. sets of steps used in developing software). Low-level components are: model components (i.e. components of the artifact model), criteria (i.e. rules),

[161] This view is not explicitly named in the architecture of Gupta *et al.* (2001).

guidelines (i.e. concrete statements, heuristics, or techniques advocated), measures, notations, and actions (i.e. processing steps). Compared to our scheme, the artifact model and the processes/actions correspond to two contextual domains, the ISD object domain and the ISD action domain. Other components concern non-contextual features. The framework defines structural relationships between the components.

To summarize, all five presentations regard the method as being composed of method components (called the method fragments in Harmsen 1997). The most comprehensive categorizations of the method components are suggested in Harmsen (1997) (with the dimensions of perspective, abstraction, and granularity level) and Zhang *et al.* (2001) (with the dimensions of level of abstraction, granularity level, and information type level). Gupta *et al.* (2001) recognize three views of method architecture, while Song *et al.* (1992) and Song (1997) consider partly only two of our dimensions.

From the processing layers, Harmsen (1997), Song *et al.* (1992) and Song (1997) recognize method components on the ME layer. The framework of Zhang *et al.* (2001) enables the consideration of components at three information type level. Gupta *et al.* (2001) is the only one to discuss the method components at the RW layer.

The domain-based classification is most closely followed in Harmsen (1997), which subdivides method fragments into product fragments and process fragments. Zhang *et al.* (2001) focus on data aspects only. Song *et al.* (1992), Song (1997) and Gupta *et al.* (2001) recognize some domain-specific components but also components which are independent from the contextual domains.

Harmsen (1997) distinguishes between conceptual and technical abstractions, of which the latter corresponds to our physical perspective. Zhang *et al.* (2001) use three levels of abstraction based on the ISD/ME stages. Song *et al.* (1992) deploy the classification of problem/solution domains, and Gupta *et al.* (2001) apply the view-based division of perspectives. Harmsen (1997) recognizes five granularity levels, Zhang *et al.* (2001) suggest three levels, and Song (1997) and Gupta *et al.* (2001) two levels. Song (1997) presents no criteria for the sub-division into levels.

In conclusion, the comparative analysis of the five presentations of method components showed that our classification scheme is the most comprehensive, distinguishing five different dimensions. The scheme with the well-defined structure also appeared to be profitable in the analysis and comparison of other presentations. We consider it important that the method components can be specified and deployed in a uniform and consistent fashion. This necessitates that the contextual domains, the perspectives, the processing layers, the model levels, and the granularity levels are uniformly defined and utilized.

At the end of this section, we want to complete the picture of how an ISD method component is seen in the ME literature by discussing the presentations of Hidding *et al.* (1993), Nuseibeh *et al.* (1996) and Vlasblom *et al.* (1995). We are particularly interested in how they see the nature and structure of an ISD method component. These presentations were not included in the above analysis because they don't contribute to the variety of method components.

Hidding *et al.* (1993) define 'task package' to mean a building block to be used for configuring large method processes. A task package is a set of activities that generate one or more deliverables that are of value to a client. A task package is characterized by sixteen key attributes. The attributes include e.g. (a) schematic, which reveals sub-processes of the task package; (b) inputs, which lists the prerequisite products for the task package; (c) deliverables, which specify the output products; (d) techniques, which describe generic approaches that can be or must be used; (e) business needs, which consider the major business requirements addressed; (f) competence and experience, which summarize skills and type of knowledge of the people who will execute the process; (g) objectives; and (h) tools. From the above we can conclude that a task package is actually a highly 'contextual' concept. Its core is composed of ISD actions and ISD deliverables. Its interface is determined by input and output deliverables, objectives, human aspects, and tools. We argue that the attributes specified for the task package are important but more clarity and coherence could be attained by explicitly applying the contextual approach in the organization of these attributes.

Nuseibeh *et al.* (1996) examine the engineering of a method from method fragments in the context of multi-perspective software engineering. They define the concept of ViewPoint to mean "loosely coupled, locally managed, distributable objects that encapsulate representation knowledge, development process knowledge and specification knowledge about a system and its domain" (ibid p. 268). Strictly speaking, a ViewPoint is divided into five 'slots': (a) style that contains a definition of the representation scheme deployed by the ViewPoint, (b) work plan that contains a description or model of the development process, (c) specification that contains a partial specification described in the notation defined in the style slot, and developed by following the process described in the work plan slot, (d) domain that identifies the area of concern of the ViewPoint, and (e) work record that contains the specification development status, history and rationale. Each method fragment describes how to develop a single ViewPoint. Because several ViewPoints may deploy the same notation and development process, Nuseibeh *et al.* (1996) define a ViewPoint template that is composed of the first two slots (i.e. style and work plan). A ViewPoint template is considered to be a single technique, and a method is regarded as a configuration or structured collection of ViewPoint templates. To conclude from the approach of Nuseibeh *et al.* (1996), we can say that also here the concept of a method component (cf. ViewPoint template) is established on fundamental constructs which clearly have the "contextual background". They involve primarily ISD deliverables (concepts and notation) and ISD actions.

Vlasblom *et al.* (1995) propose the three-level description of a method. The highest level, called the generic level, is composed of building blocks of a method. The building blocks are the ISD activities, the ISD products, the development strategy, the techniques, the tools, and the disciplines (i.e. particular areas of expertise). For establishing a development method that is optimally tailored to a particular project, a proper set of building blocks are selected and, if necessary, customized and then integrated. The subdivision into the building blocks follows, to some degree, the boundaries between contextual domains (i.e. the ISD activities and the ISD products). The techniques encompass representation and working practices. The selection of a development strategy puts a time-sequence to the activities. Disciplines concern ISD actors whose expertise is needed in techniques. Thus, also here several contextual domains are involved through the notion of a building block, but in a way that makes the setting rather messy and difficult to manage in enhancing and integrating methods.

9.8.7 Summary of Method Components

Parts of existing methods can be and should be reused as ISD method components. But not any part of the methods suits a reusable component. In this section, we defined the notion of an ISD method component with the aim of ensuring that the integration of ISD method components produces coherent and consistent ISD methods. An essential means for fulfilling this goal is the use of contextual interfaces of method components. We also defined multi-dimensional classification scheme for recognizing and relating method components on five dimensions. ISD method components and their integration through contextual interfaces were illustrated with examples. Finally, we made the comparative analysis of method components, which indicated that our view of a method component is much more comprehensive and better structured than those suggested in the ME literature.

9.9 Summary and Discussions

In this chapter our main purpose was to present the ISD method ontology. We started with a short review of the empirical research into the method use in order to answer the question 'Why' the method is needed altogether. The review indicated that in spite of severe problems with ISD methods and method use, organizations have clearly benefited from using ISD methods. They are considered artifacts, which convey the "best" practices on ISD and help achieve better outcomes through more efficient, effective and manageable ISD processes.

Second, we discussed the method as a 'carrier' of ISD knowledge and derived two classifications of ISD methods. The first classification is based on the kind of knowledge the methods convey, and includes generic ISD methods, domain-specific ISD methods, organization-specific ISD methods, and project-specific ISD methods. The other classification is based on the form in which the methods are presented. With these classifications we can better recognize and understand a large variety of ISD methods. They also help us figure out how to structure and support the engineering of methods of different types.

Third, we recognized seven basic views on the ISD method, called the methodical views. These views, anchored on the semantic ladder (Stamper 1973, Stamper 1996), are the historical view, the application view, the generic view, the contents view, the presentation view, the physical view, and the structural view. Building on these views we gave an integrative definition of the ISD method that comprehensively reflects multiple aspects of the method.

Fourth, we established, grounding on the defined views, the ISD method ontology in seven parts. One of the parts, related to the conceptual contents of the ISD method, has already been established as the ISD ontology in Chapter 8. For the rest of the parts, the concepts and constructs were defined and presented in meta models.

Fifth, we applied the ISD method ontology to consider, from a broader perspective, a range of methodical support and recognized three types of artifacts that are not acknowledged to be methods although they provide some methodical support. These artifacts are the ISD methodical framework, the ISD methodical skeleton, and the ISD methodical tool kit. We defined the notions and compared them to one another, to four types of ISD methods, and to artifacts presented in the literature. The methodical skeleton is important to this work, because MEMES (Method Engineering MEthodical Skeleton) presented in Chapter 11 is a specialization of it. Finally, we discussed the criteria for acknowledging an artifact as the ISD method. Although it is not possible to set the definite criteria, we brought out a set of requirements, structured according to the ISD method ontology. We stated, for instance, that an artifact has to provide knowledge about its history, domains and ways of applying it, as well as fundamental assumptions underlying it. Its conceptual foundation should cover at least three ISD domains (i.e. ISD purpose domain, ISD actor domain, and ISD actions domain), three ISD perspectives (i.e. ISD systelogical perspective, ISD infological perspective, and ISD conceptual perspective) and two model levels (i.e. meta model level and type model level).

Sixth, we made a comparative analysis of seven well-known frameworks and models that are aimed at either the comparison and evaluation of the ISD methods, or categorizing method knowledge. The analysis showed that the ISD method ontology, better than the others, covers the contextual features of the ISD and the ISD method. In addition, our way of structuring the parts and features of the method makes the ontology more explicit and easier to apply.

Seventh, we defined the notion of an ISD method component with the contextual interface. We also presented a multi-dimensional classification of method components based on the contextual ontologies. We illustrated the notions with examples of ISD method components and method integration. Finally, we made a comparative analysis of the conceptions of method components in the literature. The analysis showed that our view of a method component is more comprehensive and multifaceted, and it enables better than the others to deal with the semantic and pragmatic nature of method components.

The ISD method ontology has been derived from, and structured according to, the underlying ontologies. This becomes quite obvious in looking at the overall picture of the ontology (Figure 90). The division into the presentation view, the contents view, and the physical view is a specialization of the semiotic ontology. The internal structure of the presentation view results from the language ontology. The historical view and the application view, defined in terms of prior and target ISD contexts and ME contexts, is an application of the context ontology. The conceptions of ISD contexts and ME contexts can be further elaborated by the concepts and constructs provided in the ISD ontology and the ME ontology. The structural view decomposes the method into parts, which are recognized in the model level ontology. This alignment of the ISD method ontology with the underlying ontologies is a clear indication of the coherence of OntoFrame.

The ISD ontology is a vital component of OntoFrame as it defines the nature, contents, structure and use of an artifact, which is the focal target of method engineering. The theory-based and well-structured ISD method ontology furthers our work of defining concepts and constructs to conceive, understand, structure and present the essential aspects of the ME context. This is what we will do in the next chapter.

10 ME ONTOLOGY AND ME METHOD ONTOLOGY

Hitherto we have constructed a multi-layered structure of ontologies for conceiving, understanding, structuring and presenting contextual features of information processing in information systems and information systems development. We have also defined concepts and constructs with which an ISD method can be conceived from several viewpoints and structured in a comprehensive manner. In this section we continue the construction of OntoFrame by building the remaining two of its components, the ME ontology and the ME method ontology. The ME ontology provides concepts and constructs to conceive, understand, structure, and represent contextual features of method engineering. It comprises two parts: ME domains and ME perspectives. The ME method ontology is composed of concepts and constructs, with which various aspects of the ME method can be conceived, understood, structured, and represented. Its structure is based on the seven methodical views defined in Section 9.5. The contents view of the ME method corresponds to the ME ontology. Both of the ontologies have been derived specializing from the underlying ontologies, in particular from the ISD method ontology (see Figure 103).

The chapter proceeds as follows. First, we describe needs for method engineering and reasons behind them. Second, based on a short literature survey we present basic classifications of ME strategies and ME processes, and derive a fundamental categorization of ME contexts. Here we also provide the definition of the ME context that integrates contextual aspects of method engineering. Third, we present the first main part of the ME ontology addressing four ME domains. For each ME domain, the concepts and constructs are defined and described in ME meta models. Fourth, we provide the second main part of the ME ontology including four ME perspectives. Fifth, we define the notion of an ME method and derive the ME method ontology from the ISD method ontology established in Chapter 9. The chapter ends with a summary.

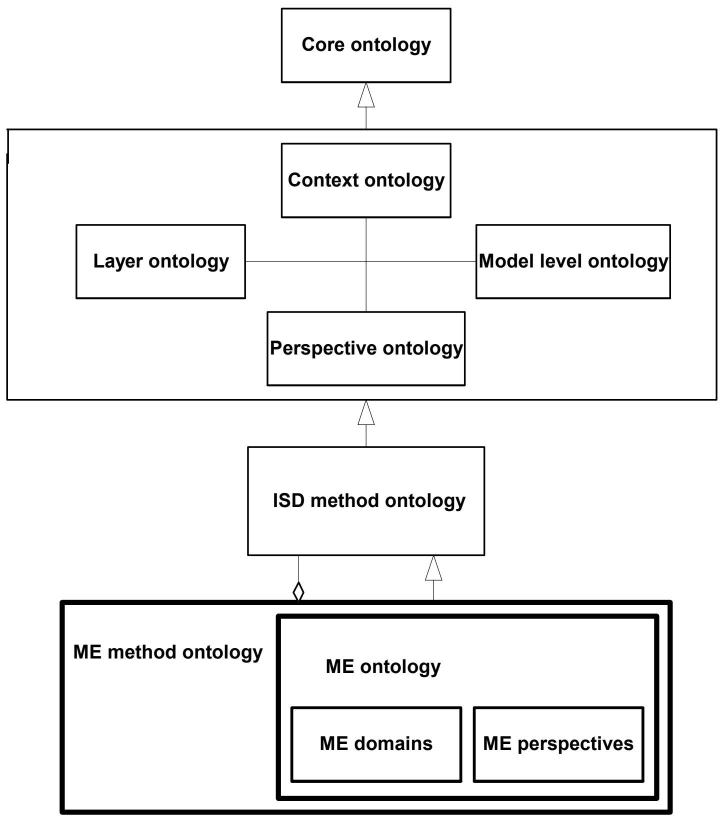


FIGURE 103 Bases and structures of the ME ontology and the ME method ontology

10.1 Motivations for Method Engineering

The purpose of this section is to bring out reasons and motives for why method engineering is needed. We start with making a survey of the ISD literature to find out how ISD methods and method use are seen in practice. In Section 8.1 we already described benefits reported in practice. Here, we are interested in problems perceived in methods and method use. One of our claims is that problems resulting from deficiencies in the methods should be tried to overcome by investments to method engineering. Second, we consider those needs for method engineering which emerge from continuous changes and evolution in business, application areas, and approaches and technologies of development environments.

10.1.1 Problems in Methods and Method Use

ISD methods are believed to further ISD work in many ways. It is believed that methods facilitate the acquisition, accumulation, use and dissemination of ISD knowledge (e.g. Hardy *et al.* 1995; Hidding 1997; Rahim *et al.* 1998; Middleton 1999; Schönström *et al.* 2003), help the management of ISD projects (e.g. Chatzoglou 1997; Fitzgerald 1998a), reduce needs for money, labor and time in the ISD process (e.g. Jones *et al.* 1988; Hardy *et al.* 1995; Chatzoglou 1997), and improve the quality of ISD deliverables (e.g. Hardy *et al.* 1995; Chatzoglou 1997; Rahim *et al.* 1998) (see more Section 9.1). In some situations these beliefs have turned out to be justified. But there are numerous situations in which the use of methods has been experienced to be problematic. Here, we first make a literature survey to find out how common method use is in practice. Second, we describe, in a structured way, problems in ISD methods and method use, as reported in empirical studies.

There are plenty of studies reporting that method use is not so frequent as believed. This was the case in the 80's (e.g. Jenkins *et al.* 1984; Sumner *et al.* 1986; Necco *et al.* 1987; Chikovsky 1988; Carey *et al.* 1988; Danzinger *et al.* 1989; Smolander *et al.* 1990) and this seems to be the case in the recent years as well. For instance, Hardy *et al.* (1995) found that only 44 % of respondent organizations reported using a recognized structured method or using formal specifications. 18 % of the cases used no method at all. In the study of over 100 organizations by Russo *et al.* (1996), only 49 % reported that a method was used consistently although 84 % of the organizations reported having a method. According to the study of Chatzoglou (1997), nearly half of 72 projects in UK used no method. Fitzgerald (1998a) concludes that only 40 % of 162 organizations used some formalized method, of which 14 % used a commercial method and the rest (28%) some in-house method. In the study of Rahim *et al.* (1998) one third of 36 organizations in Brunei and in the study of Iivari and Maansaari (1998) 23 % of the organizations did not use any method. Holt (1997) reports that nearly one-third of UK organizations did not embrace any structured methods.

To interpret the numbers right it is important to notice that method use is an ambiguous notion in many ways. First, as Hidding (1997) points out, even if practitioners tell researchers not having used a method, they produce deliverables demonstrating they do indeed use it. Practitioners have internalized the method through training and repeated on-the-job use. They no longer "interpret" the method, as they have "compiled" it (Hidding 1997, 105). Second, the purpose and form of usage depends on the role a person plays in ISD. Hidding (1997) distinguishes between sellers, planners, doers, and managers. Sellers market projects. A planner in his/her role is responsible for project planning and control. A doer is responsible of executing actions. Depending on the role in and objectives of utilizing the method, practitioners have different needs for the method. Third, the significance of the method varies with knowledge a practitioner has about the method. Novices follow strictly the method as such, while practitioners with several years experience take freedom to deviate from instructions of the method when reasonable. Hence, when asking whether the method is used or not, it is necessary to make at least the following questions: Who is using the method? What is the part of it used? For what purposes is it used?

A large variety of explanations are given for problems in method use in the literature. We start describing them by considering issues related to the implementation of the method into the organization. Veryard (1987, 470) lists three possible failure scenarios. First, the method may have been poorly chosen for the organization. Second, the method may be appropriate for the organization but insufficient start-up resources have been allocated to it. Third, the implementation may be badly managed, that is to say, there are failures in planning and/or controlling. Roberts *et al.* (2001, 635) suggest that one possible answer to failures at implementing and adopting the ISD method is the lack of in-house expertise and difficulties in transferring technical 'know-how' that allows one to use the method efficiently and effectively.

Next, we consider problems and explanations that are more related to characteristics of ISD methods. Systems development is not actually an orderly rational process, even though most ISD methods treat it as such (Wastell *et al.* 1993). Curtis *et al.* (1988) found that ISD methods influence the mental model that ISD actors have about how ISD should occur, and they were frustrated that conditions surrounding their project prohibited them from following this model. This was also noticed in Nandhakumar *et al.* (1999). The method can be applied in a ritualistic way, which inhibits creative thinking (Stolterman 1992; Kautz *et al.* 1994; Wastell 1996). ISD actors often proceed in slavish and blind adherence to methods and lose sight of the fact that development of an actual system is the real objective (Fitzgerald 1994, 371-380; Wastell 1996; Nandhakumar *et al.* 1999). Complex, highly detailed and norm-based descriptions of development routines do not fit the needs of ISD actors, nor support their work (Fitzgerald 1996a; Russo *et al.* 1996; Middleton 1999).

Methods are often perceived as prescriptive, burdensome and difficult to apply (Thomson 1990; Middleton 1994). Rahim *et al.* (1998, 957) found out that the difficulty in learning the method was the most pressing problem. Hidding (1997) reports that methods are often too massive and complex to be easily adopted and adapted to a specific situation. This is also the conclusion in Fitzgerald (1996a) and Kautz *et al.* (1994). Many studies also report on the inability of methods to cover the whole life cycle of software projects (Fitzgerald 1996a; Russo *et al.* 1996; Rahim *et al.* 1998).

Instead of helping to reduce project completion time, some projects built with a recognized method actually increased scheduled project time (Rahim *et al.* 1998). A study of Saarinen (1990) concludes that success of projects was not affected by the use of any method. The same kinds of results have been reported by Orr (1993) and Masters *et al.* (1992). Avison *et al.* (2003) noticed disappointments in productivity. There can be several reasons for this; e.g. learning may require extra time (Rahim *et al.* 1998), method use may induce unnecessary documentation (Kautz *et al.* 1994; Wastell 1996), or too much attention is given to notations (Wastell 1996).

Sometimes the methods are seen to be too detailed to efficiently support the planning of a project (Hidding 1997). Many methods are documented only on paper, making their use awkward and their customization difficult. Tools advocated by method proponents can be costly and they may require highly technical skills.

Methods are often regarded as monolithic (Hidding 1997) or one-dimensional (Avison *et al.* 2003, 81), advocating a single path or approach, which is often perceived as one-size-fit-all. They are not contingent on the type or size of a project, nor upon the technology environment and organizational context (Avison *et al.* 2003). Adaptation of a rigid method to project-specific circumstances is difficult (Middleton 1999; Henderson-Sellers 2003).

Part of the problems in method use can be traced back to human, organizational or technical settings. For example, without proper training the method can be totally ignored or only partly utilized. Incompatibility of a method approach with the culture and traditions of an organization may also cause unsolvable problems. Nevertheless, there are many problems that result from the nature, contents and structure of methods. To improve this, method engineering is needed.

10.1.2 Other Factors Propelling Method Engineering

Besides for improving existing methods, method engineering is needed for many other reasons. Here, we consider these in terms of changes and evolution in (a) business and its environment, (b) application areas, and (c) approaches and technologies of development environments.

Business processes are changing on various dimensions (e.g. flexibility, interconnectivity, coordination style, autonomy) due to market conditions, organizational models, and usage scenarios of information systems. They are required to act more effectively in shorter time-frames (Fitzgerald *et al.* 2002). At the same time, business processes are getting more complex and difficult to manage. Businesses are increasingly moving to extensive automation of their private and public processes. Increasing domestic and global competition and changing economics are creating pressures to deliver information systems "yesterday" to exploit business opportunities (Wynekoop *et al.* 1997).

Resulting from evolution in business and its environment as well as from advancements in IT, novel application areas have emerged and are emerging. Examples of the new areas are: e-commerce, m-commerce, web-information systems, multimedia information systems, trustworthy systems, and ubiquitous systems with time-aware, location-aware, device-aware and personalized services. Typical for new areas is that they amalgamate organizational, conceptual and technical issues from several research fields. For these areas new approaches and concepts are needed.

Rapid progress of technology has resulted in new architectural frameworks and platforms for information systems, e.g. J2EE, Visual Studio .NET, XML-based technology, service-oriented architectures, peer-to-peer technology (P2P), model-driven architecture (MDA), and grid computing technology. This has led to the birth and diffusion of new computing and development approaches and paradigms, e.g. agent-based approach, fuzzy approach, anywhere/any time/any means paradigm, generative programming approach, aspect-oriented approach, ontology & service oriented (OSO) programming approach, soft computing approach, peer-to-peer computing paradigm, etc. Especially, the component-based approach with reusable components has established a firm foothold in ISD. Companies rely far less on in-house development of systems, but are pursuing to buy software packages, or outsourcing ISD. This might be referred to as the industrialization of ISD.

Some of the systems are less likely to require large-scale, long-term development projects, and more likely to be smaller, short term, incremental projects (Baskerville *et al.* 1992; Fitzgerald *et al.* 2002). With the emergence of light web-based applications, new birth of ″quick and dirty″ approaches, currently called agile approaches or short cycle time systems development, are getting popular (Agile Alliance 2002; Cockburn 2001; Astels *et al.* 2002; Baskerville *et al.* 2004). They emphasize e.g. individuals and interactions over processes and tools, working software over comprehensive documentation, and customer collaboration over contract negotiation. Also emergent organizations require new practices for ISD, like continuous analysis of IS applications, dynamic requirements negotiations, and continuous redevelopment (Truex *et al.* 1999).

10.1.3 Summary

Although the ISD methods have mostly appeared to be useful both to the ISD process and its outcomes, severe problems have been perceived in the implementation and deployment of the methods. Some of the problems clearly result from drawbacks and deficiencies in existing methods. For this reason there is a need for ME efforts to overcome these problems. ISD is in constant change and present trends suggest that this dynamic nature of practice will persist. Regardless of thousands of methods already engineered and deployed in organizations, still more methods with novel features and functionalities are desired. Engineering of new methods is propelled by everlasting changes in organisational and technological environments of ISD. Demands and expectations on both the process and the outcome of ISD have become harder and harder (Roberts *et al.* 2001). Phenomena, such as Software Process Improvement or the Capability Maturity Model (CMM) (Paulk *et al.* 1993) and ISO 9000-3 set of quality standards (ISO 1991), require disciplined use of systematic practices. The rise of such phenomena stimulates the use of methods much more than before. Finally, a conception about "one method suiting any situation" has been buried a long time ago. To have a method applicable to a particular organization or project requires special ME actions for customizing and configuring existing methods.

10.2 ME Context

Method engineering (ME) and ME context are ambiguous and multifaceted notions about which there are quite different conceptions in the ME literature.

The purpose of this section is first to make a short review of the conceptions and terminologies. Second, we define main ME strategies and ME processes, and present a fundamental categorization of ME contexts. Third, we give a generic definition of the ME context and discuss how the ME context is functionally and temporally linked to other contexts.

10.2.1 About the Notion of Method Engineering

Kumar and Welke (1992) are regarded as "godfathers" of method engineering. They used the term 'methodology engineering' to mean "a meta-methodology for designing and implementing information systems development methodologies" (ibid p. 257). Since then ME has been referred to with different meanings. On one hand, method engineering is regarded as a discipline to "build project-specific methods" (Brinkkemper *et al.* 1999, 209) or to "design, construct and adapt methods..." (ter Hofstede *et al.* 1997, 401). On the other hand, method engineering is seen as "an approach to configure project specific methods for the development of information systems" (van Slooten *et al.* 1993, 167) or as "the systematic analysis, comparison, and construction of information systems engineering methods" (Harmsen 1997, 25). Tolvanen (1998) defines ME to mean "a change process taken with respect of an ISD object system in a set of ISD environments by a method engineering group using a meta method and supporting tools to achieve or maintain methods for ISD" (ibid p. 66). ME is defined to address only the methods (Brinkkemper *et al.* 1999, Harmsen 1997) or also techniques and tools (ter Hofstede *et al.* 1997; Tolvanen *et al.* 1996; Tolvanen 1998; Brinkkemper 1996).

In many studies ME is seen to be analogous to ISD (Olle *et al.* 1983; Kumar *et al.* 1992, 262; Tolvanen *et al.* 1996; Tolvanen 1998). As early as in 1983 it was stated:

> "Designing of methodologies and of application systems are very comparable exercises in human endeavour. In both cases one has to decide what data is needed and what processes are to be supported. This recursivity (if that is the right word) means that one should be able to specify a design methodology in itself – an assertion that was made in earlier days about programming languages" (Olle *et al.* 1983, vii).

According to this view, ISD yields an IS model and its implementation, whereas ME yields an ISD method and its implementation. On a general level, we can agree on this. But there are several intrinsic differences between ISD and ME. First, the targets of actions are somewhat different. An IS model and its implementation are commonly more concrete and better defined than an ISD method. Second, resulting from the above fact, the process with which the outcomes are produced in ISD is more perceivable and structured than in ME. Third, a variety of contexts, functionally, organizationally and temporally, is much larger in ME than in ISD.

Besides 'method engineering' several other terms are used in the ME literature. Some of these terms used are customization (e.g. Cronholm *et al.*

1994; Hardy *et al.* 1995; Hruby 2000b), tailoring (e.g. Basili *et al.* 1987; Mayer *et al.* 1995; Henderson-Sellers *et al.* 1999c; Kruchten 2000; Demirors *et al.* 2000; Fitzgerald *et al.* 2003), configuration (e.g. Brinkkemper *et al.* 1995; Kruchten 2000; Karlsson *et al.* 2001; Karlsson 2002), adaptation (e.g. Tolvanen *et al.* 1993; Russo *et al.* 1995; ter Hofstede *et al.* 1997; Backlund *et al.* 2003; Carroll 2003), modification (e.g. Kruchten 2000), implementation (e.g. Veryard 1987; Roberts *et al.* 1998; Roberts *et al.* 2001; Yadav *et al.* 2001; Backlund *et al.* 2003), and integration (e.g. Short 1991; Nuseibeh *et al.* 1992; van Slooten *et al.* 1993; Kronlöf 1993; Ryan *et al.* 1996; Song 1997; Goldkuhl *et al.* 1998; Saeki 1998; Wieringa *et al.* 1998).

Various prefixes are used to highlight specific aspects of method engineering: e.g. situation specific methodology construction (Kumar *et al.* 1992), situational method engineering (e.g. Harmsen *et al.* 1994; van Slooten 1995; Harmsen 1997; ter Hofstede *et al.* 1997; Brinkkemper *et al.* 1999; Ralyte 2002), incremental method engineering (Kelly *et al.* 1994, Tolvanen 1998), context-specific method engineering (Rolland *et al.* 1996), simulation-based method engineering (Peters *et al.* 1996), ontology-based method engineering (Rosemann *et al.* 2002), and assembly-based method engineering (Ralyte *et al.* 2003).

Depending on, or regardless of, the used terms, quite different things are designated with those terms. The method under construction may be quite general (e.g. Unified Process (Jacobson *et al.* 1999)), at one extreme, or highly specific, intended to the use of a particular project, at the other extreme. Actions in ME may be scheduled for execution primarily before the starting of an ISD effort, or it is emphasized that ME is an organic part of an ISD project and thus it should be timed in parallel to the ISD actions (cf. blueprint ME vs. evolutionary ME in Rossi *et al.* 2004). There are also different approaches to organizing an ME effort and to deploying computer-aided tools in ME (Tolvanen *et al.* 1996))

To summarize, there is a large variety of terms and conceptions with which diversified features of method engineering are conceived, understood, structured and presented in the ME literature. There is clearly a need to define a unified vocabulary with which the vague domain of ME can be perceived in a more consistent and structured manner. For this reason, we next define categorizations of ME strategies and ME processes and derive a fundamental categorization of ME contexts from them. After that we are ready to give our integrating definition for the ME context.

10.2.2 Categorization of ME Contexts

ME contexts can be categorized according to ME strategies, ME approaches, ME actors, ME actions, ME deliverables, etc. Here, we use ME deliverables, ME strategies and ME processes to derive a fundamental categorization of ME contexts.

In Section 9.3 we classified the ISD methods into generic methods, domain-specific methods, organization-specific methods, and project-specific methods. Because the generic ISD methods and the domain-specific ISD method mostly differ from one another only in the specificity of concepts and constructs with which they refer to special features of application domains, we treat them here together under the name 'generic methods'.

We define a generic way of accomplishing an ME effort to mean an *ME strategy*. We distinguish between three ME strategies. They are creation, integration, and adaptation. *Creation* means the "greenfield" or "from scratch" strategy of engineering the ISD method in a situation where no ISD existing method is available to be used as a basis for ME. Although this strategy is never literally applied as an overall strategy in practice, it is important recognize it for the cases where some part of the method has to be engineered without the support of existing methods. *Integration* means an ME strategy according to which the ISD method is engineered by assembling components of existing methods. The more reusable components the ISD methods are composed of, the easier the process of integration is. *Adaptation* means an ME strategy according to which the ISD method is engineered by dropping off or modifying some part(s) of an existing ISD method, or extending an existing ISD method with some new part(s).

The three ME strategies correspond to three ISD strategies distinguished in the ISD literature. The creation strategy means a traditional way of building an IS from "hand-designed" and "hand-coded" parts in ISD (Yourdon 1989). The integration strategy in ME corresponds to the COTS (components-of-the-shelf) strategy (Bertolazzi *et al.* 2001; Morisio *et al.* 2002; Dogru 2003; Olarnsakul *et al.* 2003) in ISD. Finally, the adaptation strategy can be seen as a counterpart of "software package" strategy (Kirchmer 1999), according to which a software package is acquired and implemented by tailoring it. Tailoring involves the selection of optional modules, the specification of parameters, and the like (cf. ERP packages) (Parr *et al.* 2000).

Based on the classifications of the method types and the ME strategies we can construct an overall framework, which brings out and relates different kinds of ME processes (Figure 104). In the framework the ISD methods under engineering are presented in the central "column". There are three kinds of methods under engineering (i.e. generic method, organization-specific method, and project-specific method) and the method in use. The last one stands for actual performance in ISD, also known as an action world (Jayaratna 1994, 228-229), project performance (Harmsen 1997, 39), method-in-action (Fitzgerald *et al.* 2002) and 'doing and practice' (Vidgen 2002, 259). Because method_in_use does not appear in the representational form like the others, it is depicted in the dotted line in the figure. The two other "columns" represent existing methods to be either integrated or adapted. Next, we define the ME processes.

Because the classification of ISD methods follows the principles of predicate abstraction based on the criterion of realization independence (cf. Section 3.9.3), we regard the methods as belonging to four abstraction levels. Between the levels we can identify three kinds of ME processes, which derive an ISD method from another ISD method on the next higher level, and three

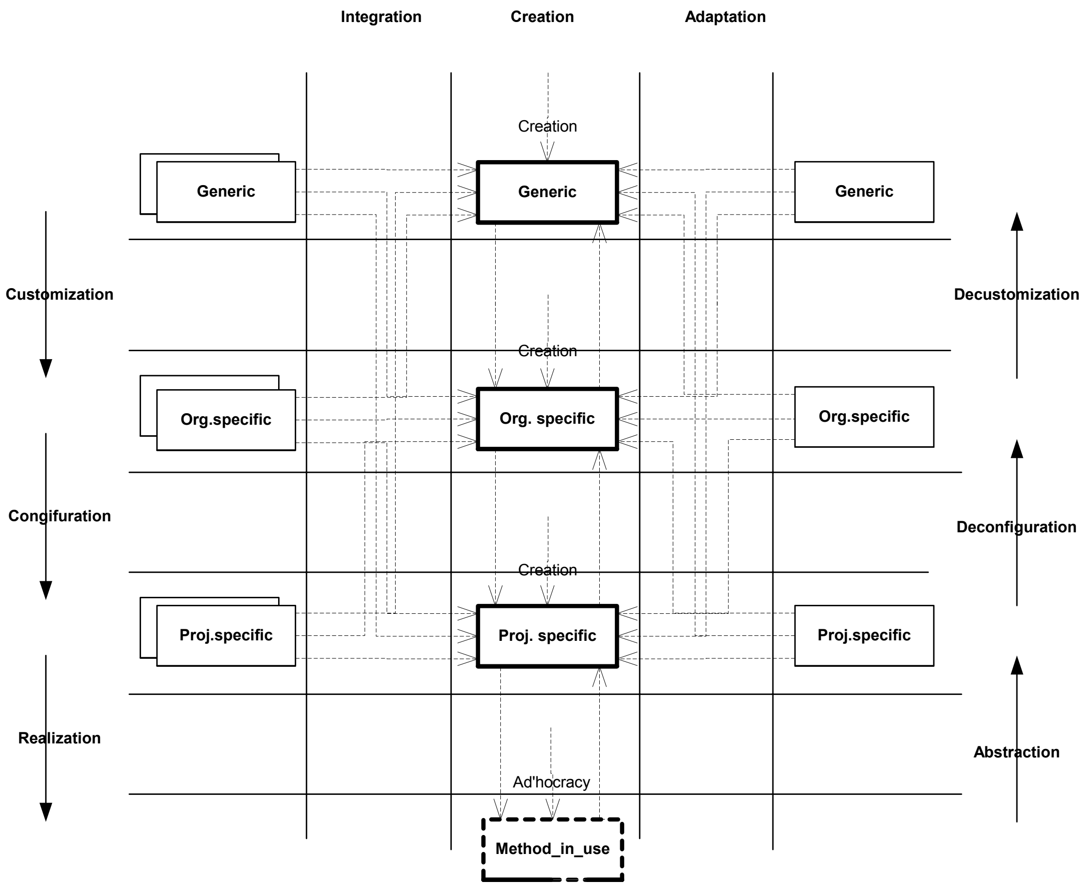


FIGURE 104 Framework of the ME strategies and ME processes

kinds of ME processes, which derive an ISD method from another ISD method on the next lower level. We refer to these processes with the common term 'adjusting'. First, we consider the ME processes by which an ISD method is adjusted from the next higher level.

Customization means an ME process by which an organization-specific ISD method is derived from some generic method (or domain-specific method) by adjusting it with organizational features that fit the traditions, culture, infrastructure, management policies, etc. of the target organization. In the customization, constructs within the ISD actor domain and the ISD facility domain are particularly specified and specialized. *Configuration*[162] means an ME process by which a project-specific ISD method is derived from an organization-specific ISD method. In this work, concrete concepts and constructs are used to prescribe e.g. who should do what, where and when. A project-specific ISD method in the most concrete form becomes very close to what we usually call a project plan. *Realization* means an ME process by which a project-specific ISD method is put into action. Method-in-use means concrete

162 The term 'configuration' is preferred here because in the ISD field the same term is used to mean the adjustment of a software package into a particular circumstance (e.g. Kirschmer 1999; Bertolazzi *et al.* 2001).

work that is done in ISD. As we know, ISD work in practice frequently deviates from the corresponding project-specific method. Recognizing this is important especially to empirical research of the method use.

Respectively, we can distinguish between three ME processes, by which a method is adjusted from another method on the next lower level. These processes are decustomization, deconfiguration, and abstraction. *Decustomization* is an ME process by which a generic ISD method is engineered by clearing an organization-specific ISD method from the knowledge specific to the certain organization. *Deconfiguration* means an ME process by which an organization-specific ISD method is engineered by abstracting project-specific knowledge from an existing method. For example, in order to harmonize ways of working inside a large organization, project-specific ISD methods followed in accomplished projects are deconfigured to achieve a common organization-specific ISD method. That method will act as a shared knowledge base on how an ISD should be, on a general level, performed in the organization. Finally, by observing, participating in, and interviewing about, ISD work in practice, one can extract "best practices" and *abstract* them into a project-specific ISD method to be adapted and utilized in forthcoming projects.

The ME strategies can be applied in quite a similar way regardless of a level on which the target method is. However, the lower the level is, the greater the number of specific concepts and constructs are and the more concrete the issues in the considerations are. On the lowest level the creation of a method without any underlying method corresponds to a situation which was typical in the so-called pre-methodological era (Hirschheim *et al.* 1995) where for each ISD action working procedures were "designed" in an ad hoc manner.

Besides engineering a method through either "vertical" processes or "horizontal" processes, there are situations in which the both kinds of processes are needed. We can recognize, for instance, the following three cases. In the first case, a new generic method is engineered integrating and adapting suitable components of existing methods, as well as decustomizing features of a certain organization-specific method. In the second case, an organization-specific method is engineered via three processes: (a) adapting an existing organization-specific method, (b) re-customizing parts of the generic method that has served as the basis for the current organization-specific method, and (c) deconfiguring some parts of project-specific methods applied in the organization. In the third case, a project-specific method is engineered through configuring the organization-specific method and adapting some parts of existing project-specific methods. The latter process transmits the experience from the finished or on-going projects into the use of the current project.

Depending on what the target of ME is, we can now distinguish between three main types of ME contexts: method development, method customization, and method configuration. *Method development context* aims to engineer a generic ISD method, or a domain-specific ISD method. Many EU projects (e.g. OSSAD (Conrath *et al.* 1989), Euromethod (Franckson 1994), EKD (Loucopoulos *et al.* 1998)) and work done for the UML (e.g. Booch *et al.* 1999, Jacobson *et al.*

1999; Kruchten 2000) are examples of method development. Also method engineering done in universities and research laboratories often belongs to this type of ME contexts (cf. Mathiassen *et al.* 1996; Mathiassen *et al.* 2000). *Method customization context* aims to attain an organization-specific ISD method. A typical example of this type is a case where a software house is "modernizing" its current method by customizing one or more novel generic methods. The generic ISD method is typically a commercial one (e.g. Rational Unified Process, Kruchten 2000). Customization is accomplished as an ME work, internally to the organization and possibly supported by consultants. This work is also known as organization-based ME (Tolvanen 1998, 21). Fitzgerald *et al.* (2003) describe one customization context which aimed to customize a software development process at Motorola. *Method configuration context* aims to engineer a project-specific ISD method. At the best case the basis for engineering is obtained from the organization-specific ISD method, but it is quite common that the ISD method for the project has to be engineered from some generic ISD method.

Next, we compare conceptions presented in the ME literature to our framework, first regarding ME strategies and then ME processes. Ralyte (2002, 129) identifies between four main ME strategies. The first strategy is 'From scratch' with the self-evident meaning. The second strategy, called 'completeness driven assembly', means enhancing the process part in the existing method by one or more new ways of working. The 'extension driven assembly' strategy is for situations where the project at hand implies adding a new functionality to the existing method, which is relevant in its other aspects. The fourth strategy, called 'restriction driven strategy', is used to select the functionalities that are significant in the project and to eliminate the others. Ralyte's (2002) first strategy corresponds to the creation strategy in our framework. The other strategies contain parts of our adaptation and integration strategies. Ralyte (2002) defines many other strategies (e.g. verification strategy, aggregation discover strategy, and state-based modeling strategy), but they are on too a detailed level to be discussed here. Ralyte *et al.* (2003) suggest a slightly different set of ME strategies categorized for setting ME goals (i.e. the 'from scratch' strategy, the 'method-based' strategy) and for constructing a method (i.e. the 'assembly-based' strategy, the 'extension-based' strategy, the 'paradigm-based' strategy). The 'assembly-based' strategy corresponds to our integration strategy, the 'extension-based' strategy is a part of our adaptation strategy, and the 'paradigm'-based' strategy is, to a degree, a counterpart of our creation strategy.

Kruchten (2000, 258) uses the term 'configuration' on two levels. In organization-wide configuration the method is modified, improved or tailored in a way which takes into consideration issues such as the domain of the application, reuse practices, and core technologies mastered by the company. Project-specific configuration, in turn, refines the method for a given project, taking into consideration the size of the project, the reuse of company assets, and the applied ISD approach. The outcome of this process is called development case (ibid p. 259). Firesmith (2002, 95) employs the term 'endeavor-specific' to characterize the development process for a project, a program of related projects, or an entire enterprise. He does not see any difference between an organization-specific method and a project-specific method. Tolvanen (1998, 21) distinguishes between the 'organization-based' method engineering and the 'project-based' method engineering.

Veryard (1987) uses the term 'method implementation' to refer to the process of taking a method into use in an organization. Karlsson *et al.* (2001, XIV-1) define method configuration to mean the adaptation of the particular method to various situated factors. Backlund *et al.* (2003) specify, on the basis of Nonaka *et al.* (1995), a generic process to adapt and implement the method into organizations and illustrate the use of the process with two case studies. Hruby (2000b, 22) describes the process of customization with which a certain subset of the best practices is identified and adopted within the organization. Vlasblom *et al.* (1995, 602-603) present a procedure for the construction of a development model (i.e. a kind of domain-specific method) through investigating completed projects and formalizing experiences into a structure form. This process corresponds to abstraction, deconfiguration and decustomization in our framework. Henderson-Sellers and Mellor (1999c) discuss the tailorability of methods created through differing ways of instantiating the OPEN framework (Graham *et al.* 1997).

From these examples taken from the ME literature we can clearly see that the ME field is very far from having a unified and shared terminology. There are discrepancies between conceptions about ME strategies, ME processes and ME contexts. We proposed the framework, which in a simple and unambiguous way distinguishes between three ME strategies, six ME processes and three ME contexts. We believe that our categorizations will substantially clarify the meanings of the concepts and differences between the conceptions presented in the literature. Based on these categorizations, we will next define the notion of the ME context.

10.2.3 Definitions of ME and ME Context

It is challenging to try to construct a single definition for a notion like ME, which has so many facets and aspects as demonstrated above. Here, we first give a general definition of ME and then elaborate a more detailed definition of an ME context. *Method engineering* means all those actions by which an ISD method is developed, and later possibly customized and configured to fit the needs of an organization and/or an ISD project. By applying the contextual approach, deriving from the ISD ontology, and rooting on the categorizations above we have elaborated the following integrating definition of an ME context:

> A *method engineering context* is a context in which ME actors carry out ME actions of (de)customization, (de)configuration, realization, and/or abstraction to produce a new or improved ISD method, with ME facilities, in a certain

organizational and spatiotemporal context, in order to satisfy ME goals set by ME stakeholders.

The definition above addresses all seven contextual domains. First, ME means intentional work guided by goals of ME stakeholders, partly predefined but mostly negotiated and agreed upon along the ME work. The ME stakeholders mean all thosee persons who have some interest in the ME process and/or deliverables, e.g. method experts, work experts, business experts, IT experts, etc. The ME work is carried out by ME actors with various expertise and backgrounds, in different ME roles, and in different organizational units. Expertise may concern research methodologies, ISD process, tools, application domains, human, social and organizational issues, etc.

Depending on the ME strategy applied (i.e. creation, integration, adaptation) and the nature of the ME context, ME actions are composed of different ME tasks and ME steps, and constitute various ME action structures. Common to all the ME approaches is the fact that the ME work starts with some requirements engineering and goal setting. After that come analysis and design of an ISD method, succeeded perhaps by implementation. During all the aforementioned ME workflows ME deliverables are evaluated with criteria derived from the ME goals. Depending on the nature of the ME context, the ISD method under engineering is either a generic method, a domain-specific method, an organization-specific method, or a project-specific method.

ME work may be supported by ME tools, comprising MetaCase tools and CAME tools. With MetaCase tools it is possible to customize CASE environments to support new ISD methods. CAME tools at best guide the process of ME and support, with basic access operations and mechanisms, its accomplishment through integrated method base and procedures for automatic derivation and verification. Examples of MetaCase tools are RAMATIC (Bergsten *et al.* 1989), Maestro II (Merbeth 1991), ConceptBase (Jarke 1992), and MetaEdit+ (Kelly *et al.* 1996). Examples of CAME's are MERET (Heym *et al.* 1992a), Decamerone (Harmsen 1997) and MERU (Gupta *et al.* 2001).

The ME context is bound to certain place and time. It is also organizationally, functionally and temporally linked to several other contexts. These links are illustrated in Figure 105. The ME context at hand is in the middle of the figure. It is connected to the prior contexts, on one hand, and to the so-called target contexts, on the other hand. The connections can be direct or indirect. The connected contexts comprise ME contexts and ISD contexts. Depending on the nature of the ME context at hand, the connected ME contexts mean development, customization and/or configuration.

Prior ME contexts mean contexts, which have contributed to the ISD method that is under consideration/engineering in the ME context at hand. The method can be a generic ISD method, a domain-specific ISD method, an organization-specific ISD method, or a project-specific ISD method. The prior ME context have created the ISD method, or adjusted it. Knowledge about the prior ME context(s) is important to understanding the 'hidden' assumptions of

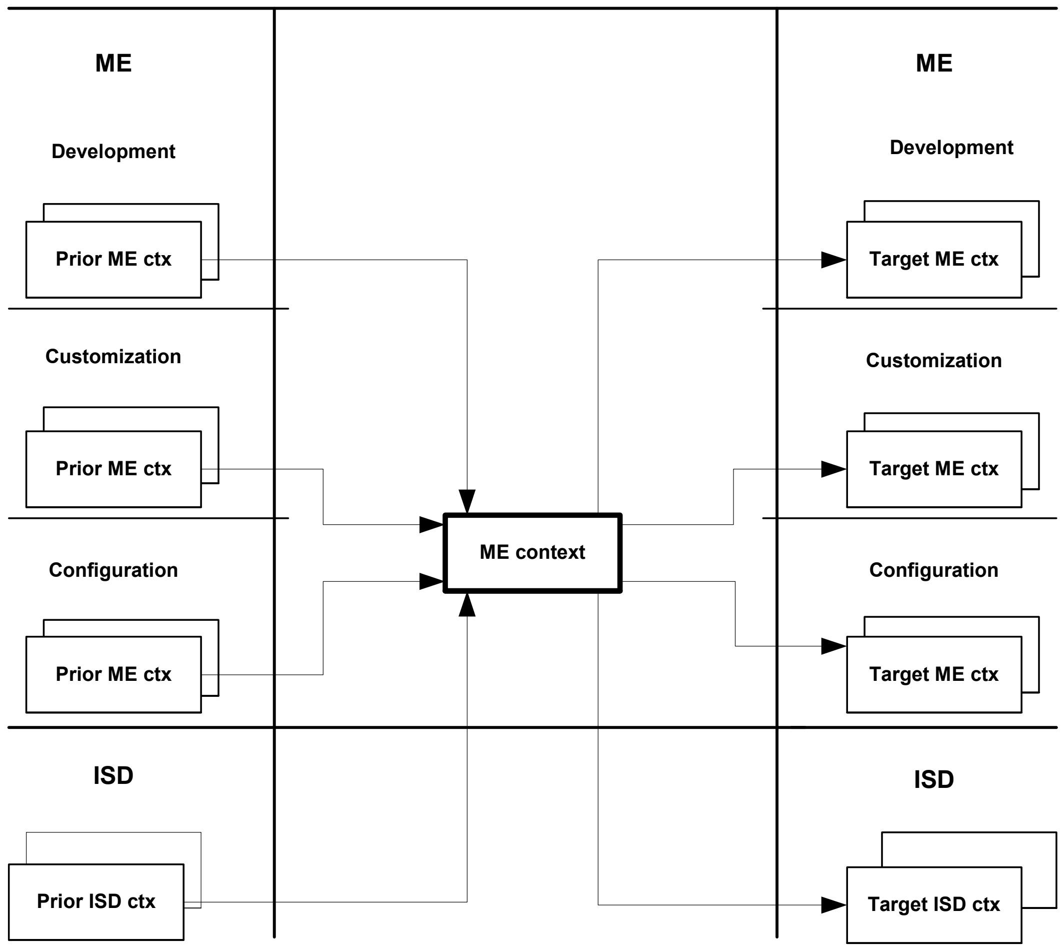


FIGURE 105 Relationships between the ME context and other contexts

the ISD method (cf. the context of creation, Jayaratna 1994) and making decisions on its suitability to the ME effort.

Target ME contexts mean contexts in which the ISD method under engineering is later to be customized, configured and/or realized into the use of certain ISD contexts. Especially, if the ISD method is a generic method, it is important to take into consideration how future customization and configuration contexts are provided with the sufficient knowledge of the original intentions of, the ME actors involved in, and the ME approaches applied in the ME context at hand.

The ME context is also related to ISD contexts. *Prior ISD contexts* mean contexts in which the ISD method(s) interested by the ME context have been applied. Experience from the applications is valuable to decisions about which ISD methods would serve as a suitable basis for the ME work and about with which improvements they would serve as such. *Target ISD contexts* mean contexts for which the ME effort at hand has been launched. Depending on the type of the ME context, the target ISD contexts are known or not. The target ISD context is exactly known if the purpose of the ME context is to engineer a project-specific ISD method. In this case, the ME context at hand and the target ISD context are scheduled at least partly in parallel to one another (Hruby

2000b). If the ME context is a customization context, target ISD contexts are seen as a family of ISD projects that share common features of the cultural, organizational and technical environment. For an ME context engineering a generic ISD method, the picture of the target ISD contexts is much more abstract.

Actually the ME context at hand has indirect connections to prior and target contexts on the IS layer as well. Prior IS contexts mean those IS contexts which have been developed in prior ISD contexts. Target IS contexts mean those application areas for which target ISD projects applying the engineered method are to be launched. Target IS contexts can be viewed from various perspectives. The IS systelogical perspective, for instance, reveals the types of business systems and intended support by IS's. Connections between the ME context and the contexts at the IS layer are, however, so thin that we do not take them into account in Figure 105.

10.2.4 Summary

Method engineering is, partly due to the short life time of the discipline, an ambiguous and obscure notion. In this section we have presented basic categorizations for ME strategies, ME processes and ME contexts. We have also proposed a framework which integrates all these categorizations and enables us to specify and analyze a large variety of ME contexts. We have applied the contextual approach to construct an integrating definition for the ME context. According to it, ME is viewed as a context, which possesses features of seven contextual domains, and is connected to contexts of different types, on different layers. We argue that this view provides a solid basis for engineering the ME ontology presented in the next sections.

10.3 ME Domains

The *ME ontology* provides concepts and constructs to conceive, understand, structure, and represent contextual features of method engineering. It is composed of two main parts, the ME domains and the ME perspectives. The purpose of this section is to present the first part. Here we focus on the ME purpose domain, the ME actor domain, the ME action domain and the ME object domain. The other three domains contain concepts that are related to the ME datalogical perspective and the ME physical perspective, which we are not interested in at the ME layer. In addition, we present an overview of ME inter-domain relationships.

For ME there are no unified presentations in the literature that would provide comprehensive frameworks, frames of references or the like, as is the case for ISD. In contrast, there are only a large variety of specific ME artifacts, including IS meta data models (e.g. ER (Chen 1976), NIAM (Nijssen *et al.* 1989), ASDM (Heym *et al.* 1992a), GOPRR (Kelly *et al.* 1996)), ISD meta process models (e.g. Bandinelli *et al.* 1993; Deiters *et al.* 1994, Kaiser *et al.* 1993), ME strategies (e.g. Kumar *et al.* 1992, Ralyte 2002; Ralyte *et al.* 2003), ME approaches (e.g. Harmsen 1997; Tolvanen 1998), ME techniques (e.g. Kinnunen *et al.* 1996; van Slooten *et al.* 1993; Grundy *et al.* 1996; Saeki 1998; Leppänen 2000), and ME procedures (e.g. Vlasblom *et al.* 1995; Harmsen 1997; Tolvanen 1998). The conceptual basis they provide are, however, quite thin and scattered. For this reason we have not been able to apply the integration strategy in engineering the body of the ME ontology. Instead, we have established the ME ontology mostly by deriving from the context ontology and the ISD ontology. Into that body we have then merged individual concepts and constructs found in the ME literature.

10.3.1 ME Purpose Domain

The *ME purpose domain* embraces all those concepts and constructs that refer to goals, motives, or intentions of someone or something in the ME context. The concepts may show a direction toward which it is due to proceed in the ME, a state to be attained or avoided, or reasons for them. Reasons can be expressed as requirements, problems, etc. The ME purpose domain is highly important because only through its concepts it is possible to express "Why" it is necessary to engineer an ISD method. In the following we define the main concepts of the ME purpose domain and present them in the meta model in Figure 106.

An *ME goal* expresses a desired state or event with qualities and quantities related to an ME context as a whole, or to some part of it. If related to the whole ME context, an ME goal may be expressed in terms of duration (ME process), money (ME resources), cooperativeness (i.e. ME organization), acceptability (i.e. ME deliverables), etc. *Hard ME goals* have pre-specified criteria for the assessment of the fulfillment of the ME goals, while *soft ME goals* have not. An example of the hard ME goals is "the ME effort should not exceed the stated budget". An *ME requirement* is some quality or performance demanded from an ME context or some part of it. An *ME problem* is a perceived deviation from a desired state or way of doing, which may lead to specifying one or more ME requirements and set up one or more ME goals.

ME goals, as well as ME requirements, are related to one another through the refinement and influence relationships. The causalTo relationship between two ME problems means that the appearance of one ME problem is at least a partial reason for the occurrence of another ME problem.

Some of the ME purposes are directly related to the ISD method (ISDM) under engineering. We use the term '*ISDM purpose*' to refer to the ME goals and the ME reasons pertaining the ISD method. For the evaluation and comparison of ISD methods a large variety of criteria are presented in the literature[163]. In the

[163] Criteria can be connected to any ME purpose. Here we are in particular interested in the criteria used to evaluate ISD methods. Therefore a criterion is related to an ISDM purpose.

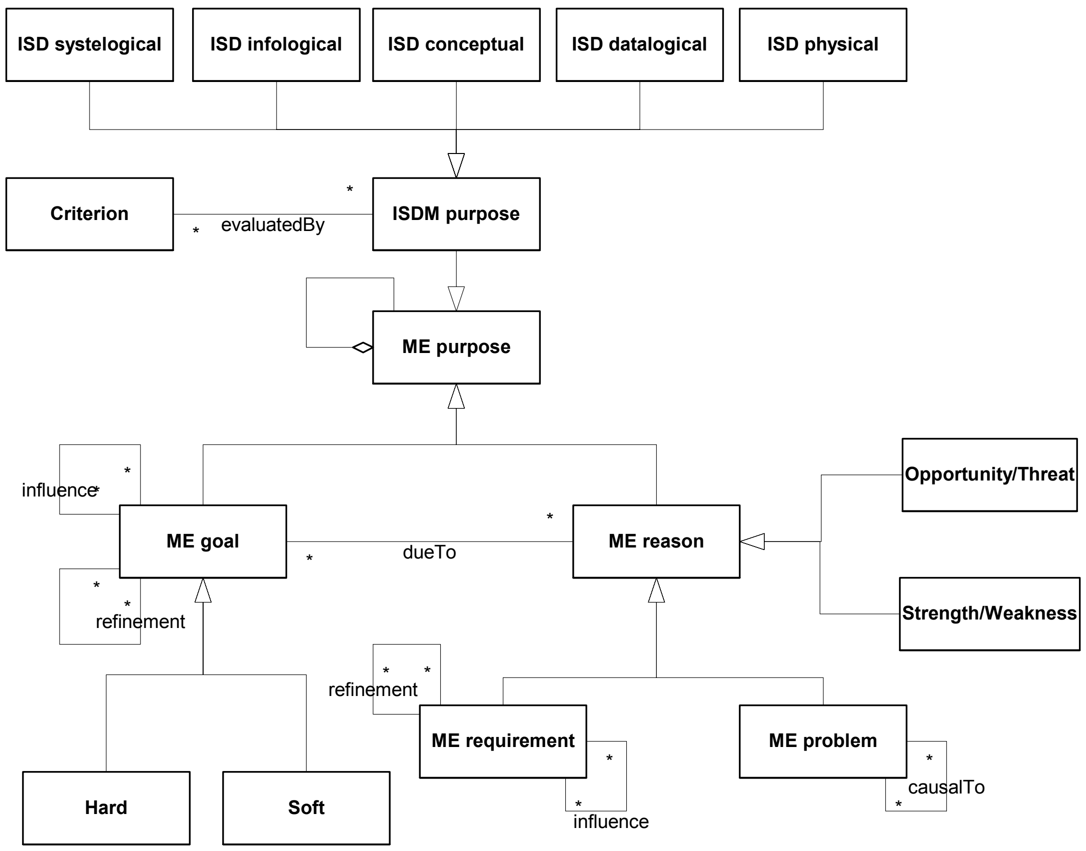


FIGURE 106 Meta model of the ME purpose domain

simplest form, criteria are given as a feature list (e.g. Rzevski 1983; Brodie 1983; Ang 1993; Karam *et al.* 1993) or as a taxonomy (e.g. Brandt 1983; Blum 1994). In a more advanced form, criteria are organized to constitute a framework (e.g. Iivari *et al.* 1983; Floyd 1986; Wand *et al.* 1989; Jayaratna 1994; Louridas *et al.* 1996; Weber *et al.* 1996; Opdahl *et al.* 2002).

Some sets of criteria are aimed at addressing the ISD method as a whole (e.g. Brandt 1983; Henderson-Sellers *et al.* 2001; Jayaratna 1994; Kabeli *et al.* 2002) while some sets of criteria only concern certain parts or features of ISD methods. The latter criteria may address paradigms or ISD approaches (e.g. Iivari *et al.* 1998a; Iivari 1991; Fitzgerald *et al.* 1998), ISD phases or ISD workflows (e.g. Castano *et al.* 1994; Bielkowicz *et al.* 2001; Bielkowics *et al.* 2002), or description models (e.g. Krogstie 1995; Krogstie *et al.* 1995; Chaves *et al.* 1996; Godwin *et al.* 1989a; Godwin *et al.* 1989b; Hommes *et al.* 2000; Moody 2003b; Shoval 1996; Weber *et al.* 1996; Kaasboll *et al.* 1996). Some sets of criteria are suggested for the evaluation of specific ISD methods; e.g. office analysis methods (e.g. Ang 1993; Auramäki *et al.* 1992b), object-oriented methods (e.g. Arnold *et al.* 1991; Hong *et al.* 1993; Iivari 1994; Liang 2000), component-based development methods (e.g. Forsell *et al.* 2000; Boertien *et al.* 2001) and agent-oriented methods (Shehory *et al.* 2001; Dam *et al.* 2004; Sturm *et al.* 2004).

Here, it is not possible to go into details of ISDM goals/requirements, nor to the related criteria. Instead, we apply the perspectives defined in Chapter 5 and in Chapter 8 to derive the classification of ISDM purposes into ISD systelogical, ISD infological, ISD conceptual, ISD datalogical, and ISD physical purposes. In the following we define them in relation to the ISDM requirements[164]. ISDM requirements from the *ISD systelogical perspective* concern information services, which ISD should provide, through the ISD method, to its utilizing system (US_{ISD}). Examples of issues addressed by the ISD systelogical perspective are: types of applications and ISD processes supported by the ISD method, ease of learning and use of the ISD method, and effectiveness of the ISD method. The ISDM requirements from the *ISD infological perspective* concern the support the ISD method provides for ISD actions and ISD deliverables. The support is expressed in terms of ISD workflows, ISD process models, and description models. The *ISD conceptual perspective* guides the focusing on ISDM requirements that concern the conceptual contents of the ISD deliverables. The requirements pertain, among others, semantic richness and complexity of the IS meta data models and abstraction structures supported by the ISD method. The *ISD datalogical perspective* addresses the support of the ISD method to establish ISD roles, ISD positions, ISD organization units, and ISD phase structure, as well as to consider in which situations CASE tools are used in ISD work. Finally, ISDM requirements from the *ISD physical perspective* involve the support the ISD method provides in detailed and concrete terms of ISD actors, ISD actions, ISD deliverables, ISD facilities, and ISD locations.

In the ME literature, there are only a few presentations which address the ME purposes in an explicit way. Rolland *et al.* (1999), Ralyte (2002) and Ralyte *et al.* (2003, 95) suggest an interesting way to describe intentions in method engineering. They introduce the ME process model, called MEMP, which is based on "the strategic process meta-model", known as the map (Rolland *et al.* 1999). The map contains two fundamental concepts: intention and strategy. An intention is "a goal that can be achieved by the performance of the process" (Ralyte 2002, 130). It refers to a task (activity) that is a part of the process and is expressed on the intentional level. A strategy represents the manner in which the intention can be achieved. The model is presented in a graph showing desired states (intentions) and possible transitions (strategies). Compared to our contextual approach, in the MEMP ME goals are expressed indirectly in terms of ME actions that actually refer to ME deliverables resulting from the ME actions. For example, 'Construct structural view' is an intention with which it is expressed that the ISD method should support the named ISD activity. In our view, it is much more natural and easier to explicitly express ME purposes (problems, goals, requirements) in separate terms, although related to other parts of the ME context.

In addition there are some suggestions for approaches to derive requirements for ISD methods. Tolvanen (1998) distinguishes between the problem-driven approach, the contingency-based approach, and the stake-holder value-based approach. Ralyte (2002, 131) define two requirements

[164] The ISD perspectives could be, correspondingly, defined in relation to ISDM goals, ISDM problems, etc.

eliciting strategies in their method engineering process model. The intention driven strategy is based on the analysis of the existing method and the detection of those development actions which must be included into the method or eliminated from it. The process driven strategy is based on the identification of those ISD actions which must be supported by the new method. The aforementioned presentations are, however, quite general. In conclusion, we can say that the ME purpose domain is in most of the presentations totally ignored, and also in those addressing the domain it is only generally considered.

10.3.2 ME Actor Domain

The *ME actor domain* consists of all those concepts and constructs that refer to human and active parts of the ME context. An *ME actor* is a human thing or an administrative thing that is, one way or another, involved in the ME context. A *human ME actor* means an individual person or a group of persons contributing to the ME work. An ME administrative actor is an ME position, or a composition of ME positions. An *ME position* is a post of employment occupied by a human actor in the ME context. It is identified with a title, composed of the defined ME roles, and equipped with a set of skill or capability characterizations. An *ME role* is a collection of ME responsibilities and ME authorities (see Figure 107).

In the ME literature several terms are used to denote ME roles and ME positions. Kumar and Welke (1992, 266) were the first to recognize the need for a specific organizational position for method resource management and engineering. They coined the term "methodology engineer/administrator". According to Kumar *et al.* (1992) a methodology engineer is a highly trained analyst who has a high-level view of systems development and systems development methodologies. He/She is responsible for the administration of a method base, which is called a component base. That means e.g. establishing equivalence mappings among the components and classifying the newly acquired components so that they can be indexed for retrieval. In addition, a methodology engineer monitors and evaluates the use of method components and stores reports of successful or unsuccessful experiences in the component base for future use (Kumar *et al.* 1992, 266).

Since Kumar *et al.* (1992), several suggestions for the role of method engineer have been presented (Harmsen *et al.* 1994, 175; Nuseibeh *et al.* 1996, 267; Odell 1996, 4; Tolvanen 1998, 67; Gupta *et al.* 2001, 156). Some of them distinguish between 'method administrator' and 'method engineer' to emphasis the significance of a method base (e.g. Odell 1996, 4; Harmsen *et al.* 1994, 175; Harmsen 1997, 115). In addition to the terms mentioned above, 'method architecture' (Iivari *et al.* 2001), 'process engineer' (Kruchten 2000, 259; Firesmith 2002, 95) and 'methodologist' (Falkenberg *et al.* 1998, 1; Firesmith 2002, 95) have been deployed.

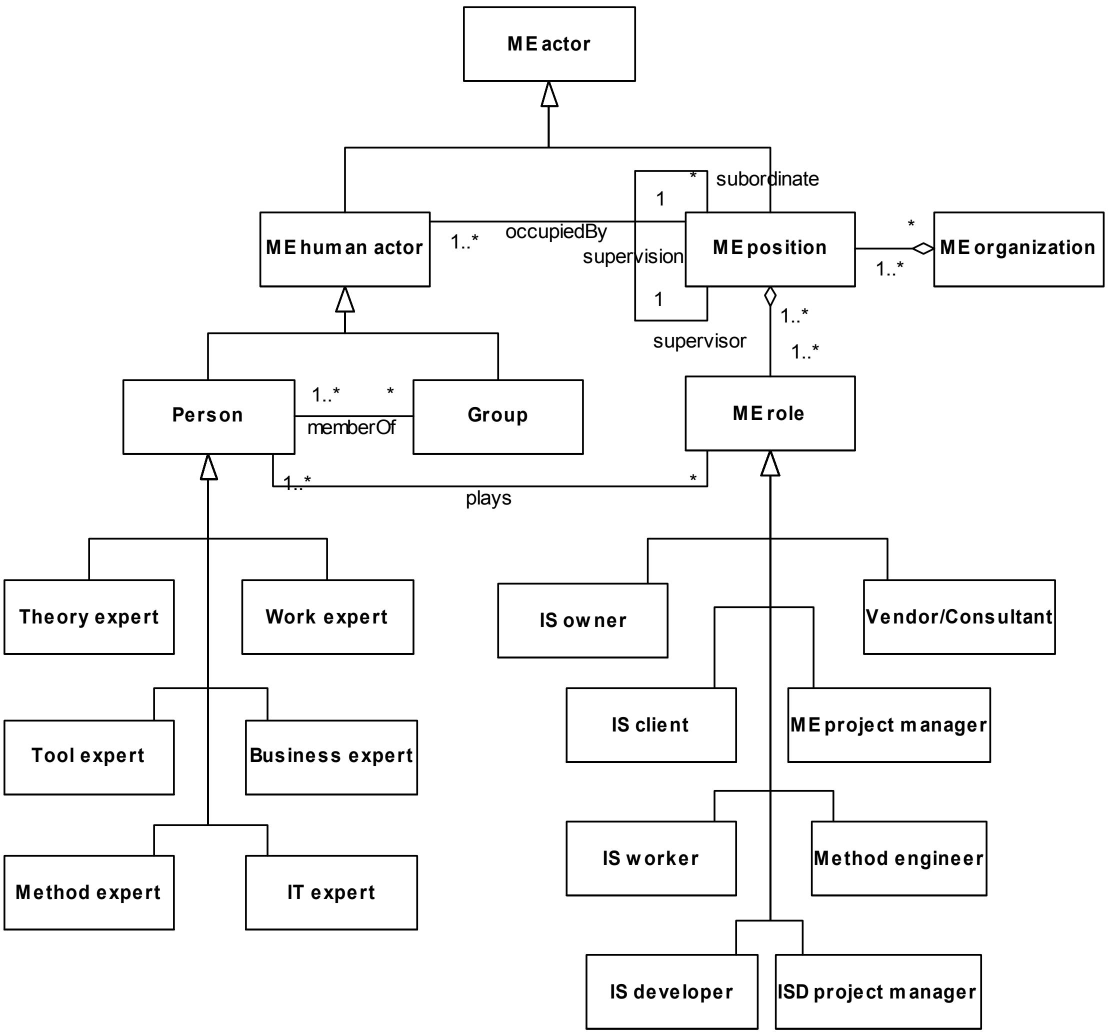


FIGURE 107 Meta model of the ME actor domain

In this study we distinguish between eight ME roles (see Figure 107). Six of them have already been defined in the ISD ontology in Chapter 8, namely the IS owner, the IS client, the IS worker, the IS developer, the ISD project manager, and the vendor/consultant. If the ME concerns method customization for an organization or method configuration for a specific ISD project, IS owners, IS clients, IS workers, IS developers and ISD project managers can provide experience about prior ISD projects and opinions about the quality of the IS's designed and implemented in those projects. They can also bring out requirements on a way of modeling of, a way of working in, and a way of organizing, the target ISD projects. Vendors and consultants may be needed if the organization has not enough expertise in the concerned method or in the ME process (Roberts *et al.* 2001, 635). These "mediating institutions" can be external consulting firms or universities. The seventh ME role is a *method engineer,* a change agent (Mathiassen 1998, 82) who has the main responsibility for ME actions in the ME effort. IS developers and ISD project managers, and to some extent IS owners, IS clients and IS workers, can be regarded as method engineers' customers (cf. Nuseibeh *et al.* 1996, 270)[165]. The eighth ME role is that of an *ME project manager* who makes plans of and decisions on how to organize the ME effort. This includes making decisions on ME phases, schedules, milestones, baselines, resource allocation, etc. We call the ME actors who play in any of the aforementioned ME roles the *ME stakeholders.*

An ME effort involves various persons with different backgrounds and expertise. According to their expertise, the persons can be categorized into IT experts, business experts, work experts, method experts, tool experts, and theory experts. The three of these have already been introduced in Section 8.3. A *method expert* is a person who has deep understanding of methods generally, and of some specific method(s) in particular. A *tool expert* is a person, who has familiarized oneself with tools used in method engineering (i.e. CAME tools and/or MetaCase tools) and in ISD (i.e. CASE tools). A *theory expert* is a person, who has special knowledge on theoretical and methodological issues of method engineering. Depending on the needs of expertise in the ME context and the availability of skilled persons, a person may act in one or more ME roles.

Method engineering can be a part time or full time activity for a person. In a large software company, for example, it is common that one or more ME positions with full time responsibilities are established for ME. There may even be a special group or unit for method engineering. Persons in such positions participate in the development of the organization-specific method, in its configuration for projects as well as in method training and consultancy. Also in multi-national projects, which aim to develop a new method (cf. the OSSAD method (Conrath *et al.* 1989), Euromethod (Franckson 1994; Euromethod 1996[166]), the ELEKTRA approach (ELEKTRA 1998)), full-time ME positions are established. In small companies responsibilities of method engineering are included in ISD positions. In this way, the expertise on daily IS development can be utilized in the customization and configuration of the ISD method.

An *ME organization* is a composition of ME positions with a coherent set of organizational goals, authorities and responsibilities. A way in which an ME effort is organized depends on the type of the ME context. In a method development context, an ME organization is composed of persons with the profound understanding of existing methods (method experts), CAME/CASE tools (tool experts), and theoretical and methodological issues of method engineering (theory experts). A customization context needs a method expert with in-depth knowledge about some generic method(s) and principles of customization. This person can come from the host organization or he/she can be a representative of a consultancy firm. In the situations like this, representatives of IS developers are also needed to bring forward knowledge of

---

165 Mathiassen *et al.* (1996) consider that the primary customers of method engineering are those studying methods to learn new ISD practices. Those who actually work with the methods (i.e. IS developers) are thought of in a secondary role. We disagree on this.

166 Euromethod is now marketed under the name ISPL (Information Services Procurement Library) (http://projekte.fast.de/Euromethod/)

the current culture, principles and ways of working in the ISD organization. In a configuration context an ME organization can comprise a method expert, an ME project manager, ISD project managers, IS developers and representatives of stakeholders with work and business expertise.

10.3.3 ME Action Domain

The *ME action domain* comprises all those concepts and constructs that refer to deeds or events in the ME context. ME actions are carried out to manage and execute an ME effort. They are performed to construct, integrate, customize, configure, and/or implement principles, models, techniques, guidelines, etc. of the ISD method. ME actions involve, for instance, the knowledge acquisition on problems encountered in prior ISD efforts, the specification of requirements for, designing improvements to, and making evaluations of the ISD method.

ME is commonly considered to be analogous to an ISD effort (Olle *et al.* 1983; Kumar *et al.* 1992; Tolvanen 1998). On this basis we argue that all the action structures defined in the ISD ontology (Section 8.3) hold, on a general level, also for the ME ontology. This means that the generic action structures (i.e. the decomposition structure, the control structures, and the temporal structures) are intrinsic to the ME actions as well. Likewise, the management-execution structure and the problem solving structure are typical to the ME actions. Because these action structures have already been defined in Section 4.4.3 and Section 8.3, they are not considered here. In the ISD ontology, the ISD phase structure is specialized, based on the Unified Process Model (Jacobson *et al.* 1999), into four phases: IS inception, IS elaboration, IS construction, and IS transition. This structure would also be applicable to the ME context following the creation strategy. But to enable the phase structure to apply to other kinds of ME contexts as well, we refrain from suggesting any specialized ME phase structure. Instead, we define the generic ME phase structure being composed of two or more unspecified ME phases. Phases may include several ME sub-phases, which in turn are composed of ME steps.

In this section we consider two ME action structures in more detail. They are the ISDM modeling structure and the ME workflow structure (see Figure 108). We discuss also how the ME action structures are intertwined. But before doing this, we make a short survey of the ME literature to find out which kinds of concepts and structures of the ME action domain are deployed.

In the literature ME actions are structured through steps (e.g. Vlasblom *et al.* 1995; Nuseibeh *et al.* 1996; Harmsen 1997; Song 1997; Tolvanen 1998), phases (e.g. Gupta *et al.* 2001), activities (e.g. Mi *et al.* 1996), or strategies (Ralyte *et al.* 2003). Vlasblom *et al.* (1995) present steps for the construction of a development model based on the existing approaches / projects, and steps for deploying development models in specific situations. Nuseibeh *et al.* (1996) decompose the ME process of method design and construction into six steps. Harmsen (1997) suggests steps to characterize the ME situation, to select method fragments and to integrate them into the ISD method. Song (1997) outline steps for function-driven and quality-driven method integration.

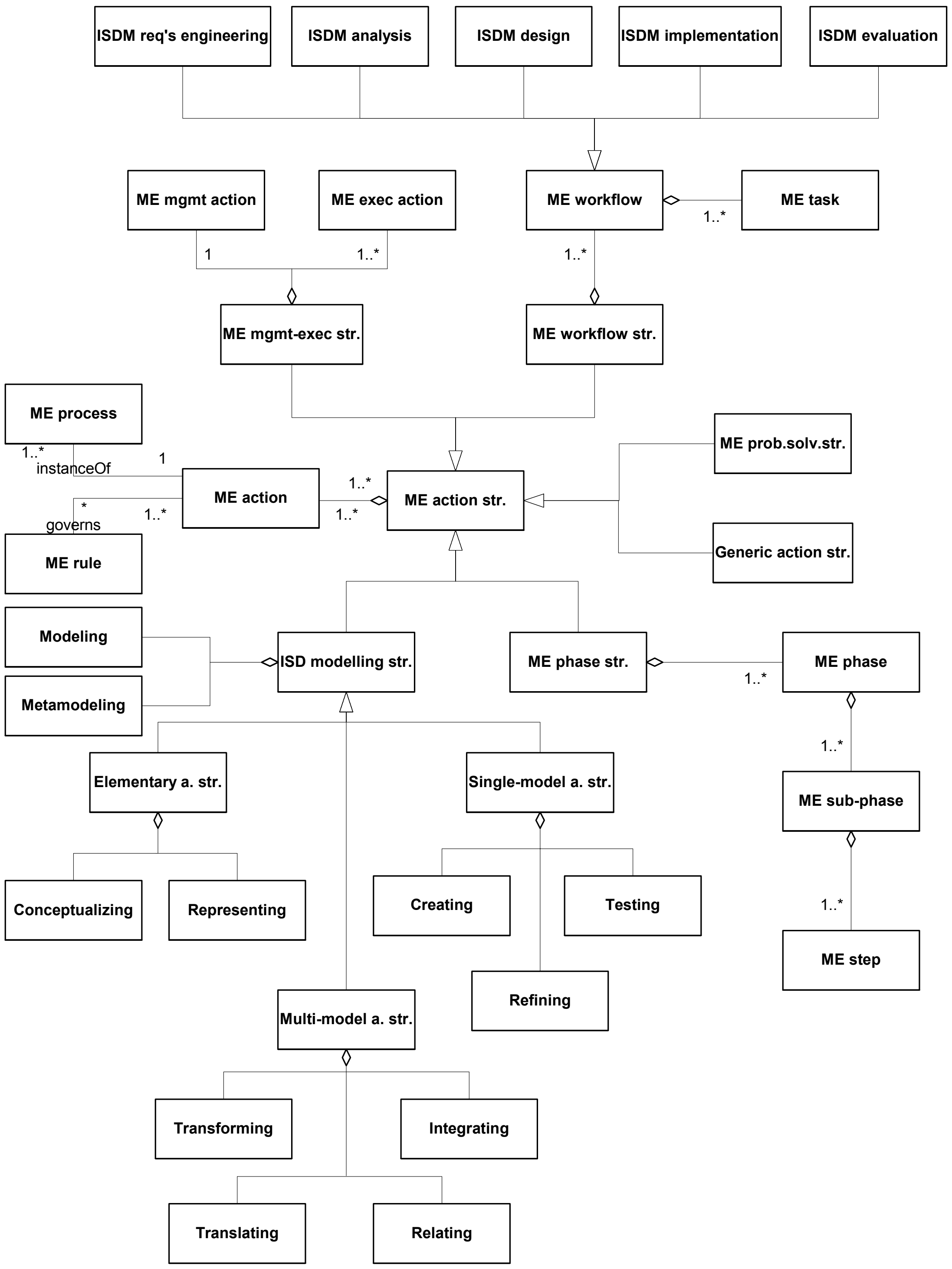


FIGURE 108 Meta model of the ME action domain

Tolvanen (1998) sub-divides the process of incremental method engineering into two kinds of steps: a priori steps and a posteriori steps. The former steps are executed before the use of a method, while the latter steps are carried out during or after the method use. In Gupta *et al.* (2001) method engineering is decomposed into three main phases: method requirements engineering, method design, and method construction and implementation. Mi *et al.* (1996) list a heterogeneous set of activities of construction and manipulation process of software development models. The list contains activities such as meta-modeling, model definition, analysis, prototyping, administration, integration, and evolution. Ralyte *et al.* (2003) describe the ME process in terms of intentions and strategies. Intentions are goals that can be achieved by the performance of the processes. Strategies represent manners in which the intentions can be achieved. In addition, Ralyte *et al.* (2003) suggest a classification of operators, such as unification, transformation, abstraction/instantiation, specialization/generalization, aggregation/decomposition, addition, and cancellation.

To summarize, we can state that propositions for the functional division of ME efforts are quite heterogeneous in the ME literature. We cannot avoid concluding that actions structures, both conceptually and terminologically, lack careful considerations and specifications. What we aim to do next, is to define the ISD modeling structure and the ME workflow structure, anchored on the underlying ontologies, and inter-relate them with more elementary ME action structures.

A. ISD Modeling Structure

Modeling plays a focal role also in ME. The main outcome of ME is the ISD method, which typically consists of various models (cf. Section 9.5). ISD models describe/prescribe ISD goals (ISD purpose models), ISD actors (ISD actor models), ISD actions (ISD action models), ISD deliverables (ISD deliverable models and ISD data models) and the like. IS meta models, in turn, provide the concepts and constructs from which IS models are instantiated during ISD efforts. Thus, ME involves modeling on two levels, on the type model level and on the meta model level. We call the structures that are composed of ME actions modeling ISD on these two levels the *ISD modeling structures.*

Modeling the structure and behavior of ISD does not essentially deviate from modeling the structure and behavior of the IS considered in Section 8.3.3. That implies that all the modeling actions and structures defined for IS modeling are relevant to ISD modeling too. These action structures comprise the elementary modeling structure (conceptualizing, representing), the single model action structure (creating, refining, and testing), and the multi-model action structures (transforming, translating, relating, and integrating). These are taken as granted here.

The process by which a meta model is produced is called metamodeling. *Metamodeling* is a modeling process, which takes place on one level of abstraction and logic higher than the standard modelling process (Tolvanen *et al.* 1996). Since the meta models also are models, they "inherit" the generic properties of the models, including the structures of modeling actions that involve the models. Thus, a meta model can be transformed or translated from another meta model, and two meta models can be related to one another and integrated to make a new meta model. There are, however, some modeling actions that are specific to metamodeling. These actions are targeted to models at two model levels. These actions are classification and instantiation (cf.

Section 3.9.2.1). Because these actions are so elementary to metamodeling, we include them in the actions of conceptualizing (cf. elementary action structure). This is justifiable because the process of classifying instance concepts and constructs of IS models into meta concepts and meta constructs actually means conceptualizing. The same is true when instantiating the meta concepts and constructs of the IS meta model.

B. ME Workflow Structure

According to the *ME workflow structure,* ME is composed of various ME workflows. An *ME workflow* is a coherent composition of ME actions, (a) which are organized to accomplish some ME process, (b) which share the same target of action, and (c) which produce results valuable for ME stakeholders. A part of an ISD workflow is called an *ME task.* We distinguish between five ME workflows: ISD method requirements engineering, ISD method analysis, ISD method design, ISD method implementation, and ISD method evaluation. In the following we use the abbreviation 'ISDM' to stand for 'ISD method' in the names of the ME workflows.

The *ISDM requirements engineering* means an ME workflow, which aims to identify and elicit ME stakeholders' requirements concerning the nature, contents and structure of the ISD method. It also seeks to establish and maintain, at least to some extent, an agreement on the essential aspects of the ISD method, and to express them as part of the ME goals. The ISDM requirements can be brought out in nearly every phase of the ME effort.

The *ISDM analysis* denotes an ME workflow, which aims to produce high-level descriptions of the ISD method, meaning that the ISD method is considered from the ISD infological perspective and the ISD conceptual perspective. Consequently, in this ME workflow concepts and constructs of the ISD purpose domain, the ISD action domain and the ISD object domain are used to make descriptions and prescriptions of what is to be done, for which, and why in the target ISD context.

The *ISDM design* refers to an ME workflow, which aims to produce more elaborated descriptions of the ISD method. Here, the ISD method is considered from the ISD datalogical perspective, uncovering "How" an ISD effort is to be accomplished. This means that the following kinds of questions are answered: What kinds of ISD roles and ISD positions are established? How the ISD actions are decomposed at a detailed level? Which part of the ISD work is to be supported by computer-based tools?

The *ISDM implementation* means an ME workflow, which aims to produce concrete descriptions/prescriptions of the ISD context from the ISD physical perspective. That means that the descriptions/prescriptions made earlier are realized and instantiated into an ISD project plan that dictates who does what, why, how, for what, when and where.

The *ISDM evaluation* means an ME workflow, which aims to produce assessments of one or more ISD methods according to the defined criteria. An ISD method can be evaluated at any point of its life cycle. It can be just a roughly outlined artifact, like that resulting from the ISDM analysis workflow, or it can be a complete ISD method already used in an ISD project. The criteria used vary from logical to technical and from general to detailed, depending on the nature of the ISD method and the ISD perspective applied.

In the ME literature, it is very uncommon to recognize ME action structures that correspond to our ME workflow structure. In their conceptual framework for evolving software processes, Conradi *et al.* (1993) distinguish between software production processes and software meta-processes. The former carry out software production activities, and the latter improve and evolve the whole software process. The software meta-processes are in charge of e.g. process requirements analysis, process design, and process assessment. Gupta *et al.* (2001) distinguish between three "ME phases": method requirements engineering, method design, and method construction and implementation.

C. Synthesis

The ME action structures are highly intertwined with one another. A manner in which ME actions appear and are interrelated to one another depends on the type of the ME context (cf. development, customization, configuration) and the ME strategy applied (cf. creation, integration, adaptation). To illustrate interrelations of ME action structures, we consider one example of the ME context. The ME context is a development context, which aims to engineer a domain-specific method, mainly with the integration strategy. The ME work covers the ISDM requirements engineering workflow, the ISDM analysis workflow, the ISDM evaluation workflow, and perhaps part of the ISDM design workflow. The ME work may start with the goal to have a proper method for a novel field, such as agent-oriented information systems or ubiquitous information systems. In the first phase, conceptions of the novel domain are concretized by IS modeling from the IS conceptual perspective. Derived from those, requirements for IS meta models, concerning special concepts and constructs, are specified. Further, requirements are expressed for such ISD approaches, ISD principles and ISD processes that support the production of ISD deliverables, based on those IS meta models. Taking these requirements as a point of departure, existing ISD methods are analyzed to find out parts that could be accepted and integrated into the body of a new method. The last step in this first ME phase is to set up goals and refine a time schedule with milestones and baselines for the ME context.

The ME work continues with refining requirements, modeling an ISD context from the ISD infological perspective, and modeling the contents of ISD deliverables from the ISD conceptual perspective. ISD modeling can be first targeted on the most essential part of the ISD process (e.g. IS analysis workflow) or on some major ISD deliverables (e.g. extended class diagram). The ISD models produced so far are tested with some case material and further refined based on the experience from testing.

In the next ME phases, the ISD context is considered from the ISD datalogical perspective in order to compose ISD actions into ISD roles and ISD positions. Also some suggestions are given for a generic ISD phase structure. Quality of the engineered ISD method, or parts thereof, can be evaluated conceptually and/or through pilot testing.

The ME context described above with a scenario comprises several ME phases, ME workflows, ME problem solving actions, and ISD modeling actions, as well as many kinds of generic action structures. ME actions in the ME action structures are highly inter-related to one another. With the well-defined action structures in the ME ontology it is possible to focus on one or two structures at a time, and thus decrease the complexity related to the ME context.

10.3.4 ME Object Domain

The *ME object domain* comprises all those concepts and constructs that refer to something to which ME actions are targeted. We call them *ME deliverables.* Based on the ME management-execution action structure we can distinguish between *ME management deliverables* and *ME execution deliverables*. Here, we confine ourselves to consider only the latter. In the following we shortly call them the ME deliverables, when there is no risk of misunderstanding, and define the concepts and relationships related to them (Figure 109).

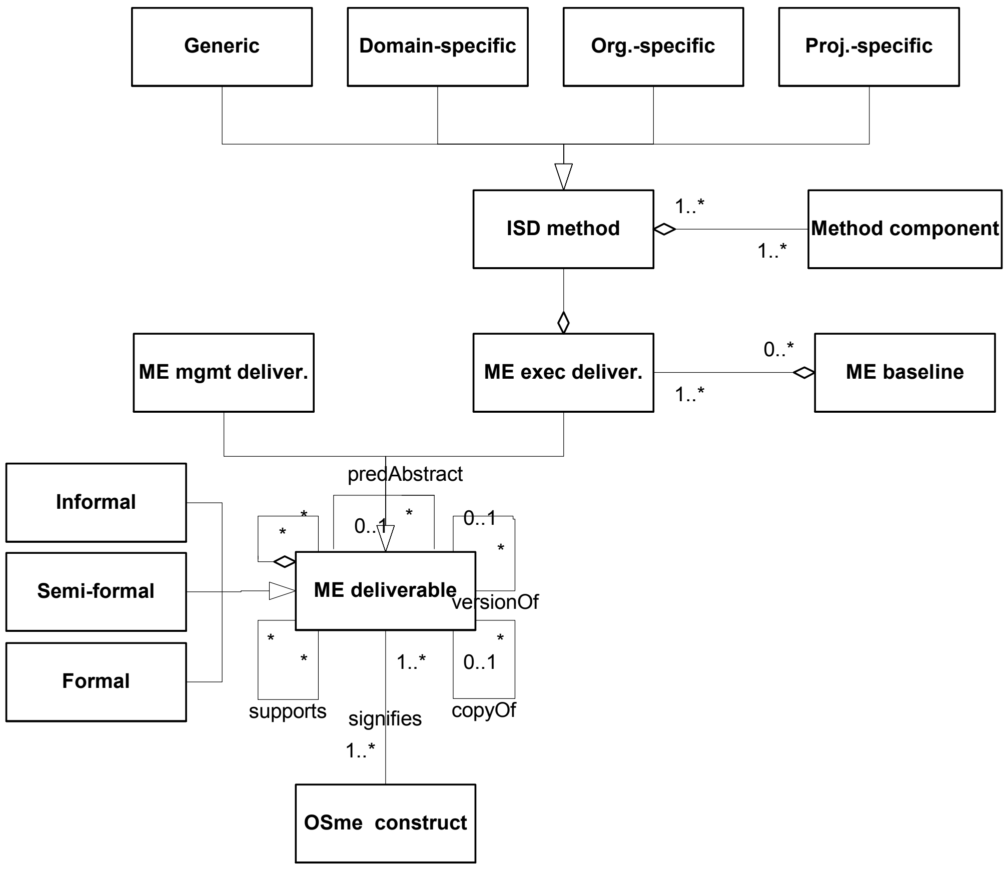


FIGURE 109 Meta model of the ME object domain

ME deliverables inherit all the features and relationships of the generic notion of an informational object specified in Section 4.4.4. This means, for instance, that an ME deliverable can be an assertion, a prediction, a plan, a rule, or a command. An ME deliverable signifies certain phenomena in OS_{ME}. The OS_{ME} is very large and heterogeneous, comprising prior ME contexts, prior ISD contexts, the ME context at hand, target ME contexts, target ISD contexts, existing ISD methods, the ISD method under construction, etc. We use the term '*OS_{ME} construct*' to denote any part of the object system of ME. The signifies relationship expresses the relationship between an ME deliverable and an OS_{ME} construct.

ME deliverables can be informal (e.g. the ISD systelogical requirements of the ISD method), semiformal or formal (e.g. the IS meta data model). Some of the ME deliverables are specified as parts of ME baselines. An *ME baseline* is a set of reviewed and approved ME deliverables. The main deliverable of ME is the ISD method, which can be a generic method, a domain-specific method, an organization-specific method, or a project-specific method. We have presented a comprehensive decomposition of the ISD method based on seven methodical views in Section 9.5. Accordingly, the ISD method, realizing a set of ISD paradigms, ISD approaches and ISD principles (the generic view), can be seen as descriptions/prescriptions, presented in one or more languages (the presentation view) and materialized in some physical form(s) (the physical view), refer to the prior contexts (the historical view) and the target contexts (the application view) with the concepts and constructs (the contents view) that constitute the conceptual foundation of the method and its components (the structural view). Because we do not want to repeat considerations already presented in Section 9.5, the only concepts we have included in the meta model in Figure 109 from the ISD method ontology (Section 9.5) are an ISD method and a method component. A method component means a well-defined part of the ISD method that can be integrated to other method components to form a coherent and consistent method.

ME deliverables are related to one another with five kinds of relationships. An ME deliverable can be composed of other ME deliverables. An ME deliverable can be used as an input to, or as a prescription for, another ME deliverable (i.e. the supports relationship). For example, the selected contingency framework can provide a structure for the characterization of the target ISD context. An ME deliverable can be a version of another ME deliverable (i.e. the versionOf relationship). An ME deliverable can be a copy of another ME deliverable (i.e. the copyOf relationship). Finally, an ME deliverable can be more abstract than another ME deliverable in terms of predicate abstraction (i.e. the predAbstract relationship). This kind of relationship exists, for instance, between an infological description of and a datalogical description of the ISD method.

In the ME literature there are no generic concept corresponding the notion of an ME deliverable. Instead, the notions of a method and a method component (or some of their counterparts, see Section 9.8) are used to refer to the target of ME efforts. The comparative analyses made in Section 8.5.4 (the ISD ontology), in Section 9.7 (the general structure of the ISD method) and in Section 9.8.6 (the component structure of the ISD method) showed that frameworks, meta-models and the like presented in the ME literature adopt rather limited views of the contents and structure of the ISD method. Without repeating here the details reported from the analyses, we state that the ME object domain in our ME ontology is much more comprehensive and better structured than in any other ME artifacts in the literature.

10.3.5 ME Inter-Domain Relationships

In the sections above the ME concepts and the ME constructs have been defined separately for each of the four ME domains. The ME domains are, however, inter-related in many ways. Figure 110 presents the general-level meta model, which illustrates the most essential ME inter-domain relationships. It has been derived from the meta models presented in Section 4.5 and Section 8.3.5. We assume that the meanings of the relationships are self-evident on the basis of the definitions given in the aforementioned sections. Here we only consider one relationship in more detail. That is the strivesFor relationship.

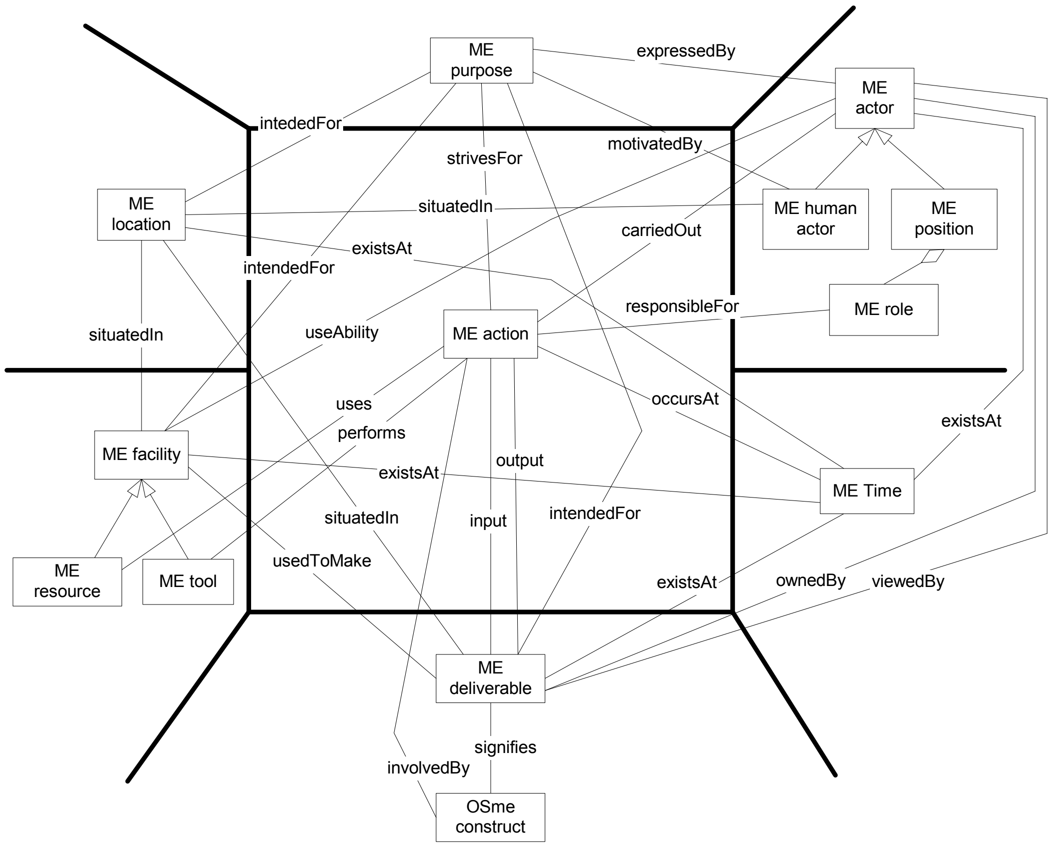


FIGURE 110 Meta model of ME inter-domain relationships

The strivesFor relationship between an ME action and an ME purpose means that an ME action is to be conducted, is conducted, or was conducted for satisfying certain goal(s). The goal(s) may be inferred from encountered ME problems, specified ME requirements, observed opportunities, or perceived threats. From the historical viewpoint, the strivesFor relationship, together with the input and output relationships between the ME actions and the ME deliverables, can be used to express method engineering rationale (cf. method construction rationale in Rossi *et al.* 2004). Method engineering rationale "garners a history of method knowledge evolution as part of the method engineering process", which develops, customizes and configures the method (cf. Rossi *et al.* 2004). With particular methods (e.g. IBIS (Conklin *et al.* 1988), REMAP (Ramesh *et al.* 1992), QOC (MacLean *et al.* 1991), PDR (Carroll *et al.* 1991) it is possible to model and reason from the knowledge on produced ME deliverables, conducted ME actions, stated ME goals, and reasons for them (i.e. arguments and justifications). This knowledge enables to trace reasons for the made decisions and actions, which is especially important in ISDM requirements engineering.

10.3.6 Summary

In this section we have defined the first main part of the ME ontology. This part is composed of concepts and relationships within and between four ME domains: the ME purpose domain, the ME actor domain, the ME action domain, and the ME object domain. For each domain, a meta model and definitions of concepts and constructs have been provided. Despite a large array of ME literature, there appeared to be no generic representation that would have helped us apply the integration strategy in ontology engineering. In contrast, engineering the ME domain part of the ME ontology had to be founded on the context ontology and the ISD ontology. Into that body we have then merged individual concepts and constructs found in the ME literature. Due to the lack of coherent representations in the literature, we were not able to make a comparative analysis.

10.4 ME Perspectives

In this section we define the second main part of the ME ontology that concerns the ME perspectives. These perspectives are important to managing the complexity related to the structure, function and behavior of the ME context. They also help us structure the process of engineering the ME method. The ME perspectives are derived from those defined on the ISD layer (see Section 8.4). Here we consider only the ME systelogical, ME infological, ME conceptual and ME datalogical perspectives. After discussing the perspectives, we describe the ME inter-perspective relationships.

10.4.1 ME Systelogical Perspective

The *ME systelogical perspective* reveals the support that method engineering provides to its utilizing system (US_{ME}). The utilizing system includes the target ISD contexts and the US_{ISD}. The definition implies that the following questions are relevant from the ME systelogical perspective:

- What kind is/are the target ISD context(s) for which the ME is to produce the ISD method?
- What kinds are the IS's for which the aforementioned ISD projects are to be launched?
- What kinds are the US_{IS} contexts which the IS's should provide with information services?
- What kinds are the services the ME should provide to the US_{ME}?
- Derived from the answers to the above questions, what are the goals and constraints for ME approaches, ME organizations, ME actions, ME deliverables, etc. in the ME context?

From the characterizations above we can infer the meta model of the ME systelogical perspective and present it in Figure 111. We see that the ME systelogical perspective concerns, one way or another, four kinds of contexts: the ME context, ISD contexts, IS contexts and US_{IS} contexts. The last three contexts are included in the US_{ME} (see Section 5.3). The ME context provides ME services to the ISD contexts, called the target ISD contexts. *ME services* means all those material and immaterial ME deliverables that are produced in the ME context and delivered to be utilized in the target ISD contexts. The main part of the ME services is, of course, the ISD method. From the ME systelogical perspective the ME context is seen as a black box, meaning that only the ME purpose domain, in addition to the aforementioned ME services, is addressed in the perspective. ME purposes are related to desired ME approaches, ME principles, ME actors, ME actions and ME deliverables, but they are expressed in a way that does not detail the concepts of the other ME domains.

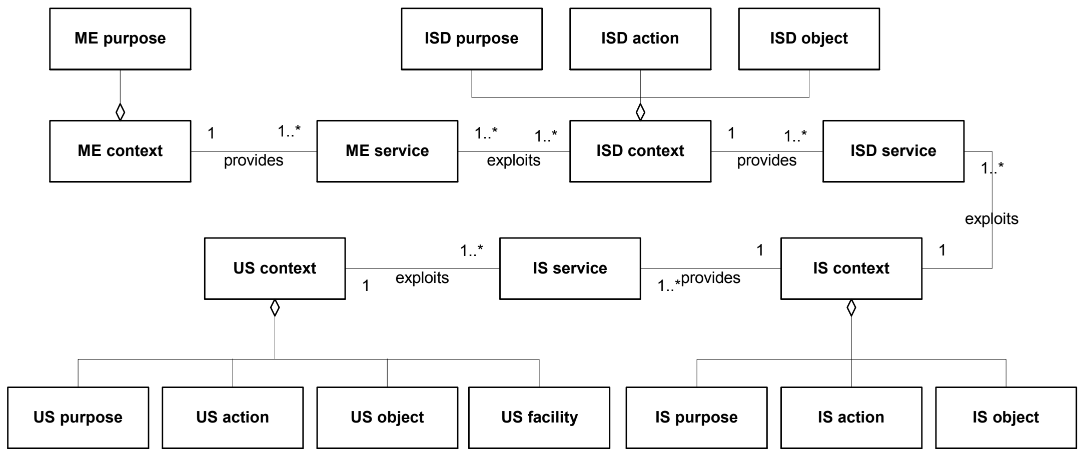


FIGURE 111 Meta model of the ME systelogical perspective

Resulting from the ME systelogical perspective, the target ISD contexts, in turn, are perceived from the ISD systelogical perspective. Depending somewhat on the kind of the ME context, features related to ISD purposes, ISD actions, and ISD deliverables of the target ISD contexts are concerned.

A variety of ISD methods under engineering is quite large. At one extreme end there are ISD methods, which focus, in particular, on the alignment of the IS with the business system. At the other extreme end, there are ISD methods, which provide special support, for instance, for technical improvements in the CIS. An example of this kind of ISD method is an architecture design method. Instead of establishing an extensive classification of ISD methods, we content ourselves with categorizing ISD methods according to those IS perspectives which are seen important in the concerned ISD methods. Thus, we have systelogical methods, infological methods, conceptual methods, datalogical methods, and physical methods. Next we characterize these methods as regards their IS perspectives. Systelogical methods provide IS meta models to analyze and design the support the IS should provide to its utilizing system. Infological methods address especially the functional structure of information processing and informational objects in the IS. Conceptual methods support ISD work with specific means to conceptualize the contents of informational objects of the IS. Datalogical and physical methods help the analysis and design of organizational structures of the HIS and/or technical structures of the CIS. Depending on the kind of the ISD method, there are differences in which IS domains are considered from the ME systelogical perspective in the ME. Assuming that the ISD method under engineering addresses especially the IS systelogical and IS infological aspects of the IS, the target IS contexts are perceived through concepts of the IS purpose domain, the IS actions domain, and the IS object domain (see Figure 111).

The ultimate goal of the ME is to benefit US_{IS} contexts with improved information and information processing through better information systems that, in turn, result from improvements in the ISD's deploying the improved ISD method. Therefore, it is necessary to include also the essential features of the US_{IS} contexts in the systelogical perspective of the ME context.

The ME systelogical perspective is not specified, not even deployed, in its entirety in any ME artifact in the literature. There are, however, artifacts that, to some extent, address some aspects related to this perspective. van Slooten *et al.* (1996), for instance, define a number of contingency factors to be used in characterizing ISD projects to help in the selection of ISD methods. The contingency factors are related to the US_{IS} context (e.g. impact = to which extent the information system will change business operations after implementation), the IS context (e.g. complexity = to what extent the functional components of the information system are complex), and the ISD context (e.g. management commitment, time pressure, knowledge and experience). Harmsen (1997, 215-221) presents a procedure for method fragments selection and assembly. The first two steps in the procedure, namely determination of the project goal and determination of the preliminary scenario, address aspects that clearly belong to the scope of the ME systelogical perspective.

10.4.2 ME Infological Perspective

From the *ME infological perspective* the ME context is seen as a functional structure of information processing and informational objects. The perspective ignores the features related to how the informational objects are presented and implemented. The relevant ME domains are: the ME purpose domain, the ME action domain, and the ME object domain. The concepts of the ME purpose domain are used to elaborate the conceptions, already drafted with the ME systelogical perspective, about why method engineering is carried out or is to be carried out. The concepts of the ME action domain are used to establish decomposition hierarchies of ME actions and to define control structures among the ME actions. The concepts within the ME object domain are used to express what kinds of ME deliverables are produced in the ME context. In the following, we consider more closely the ME infological perspective on the basis of the meta model presented in Figure 112.

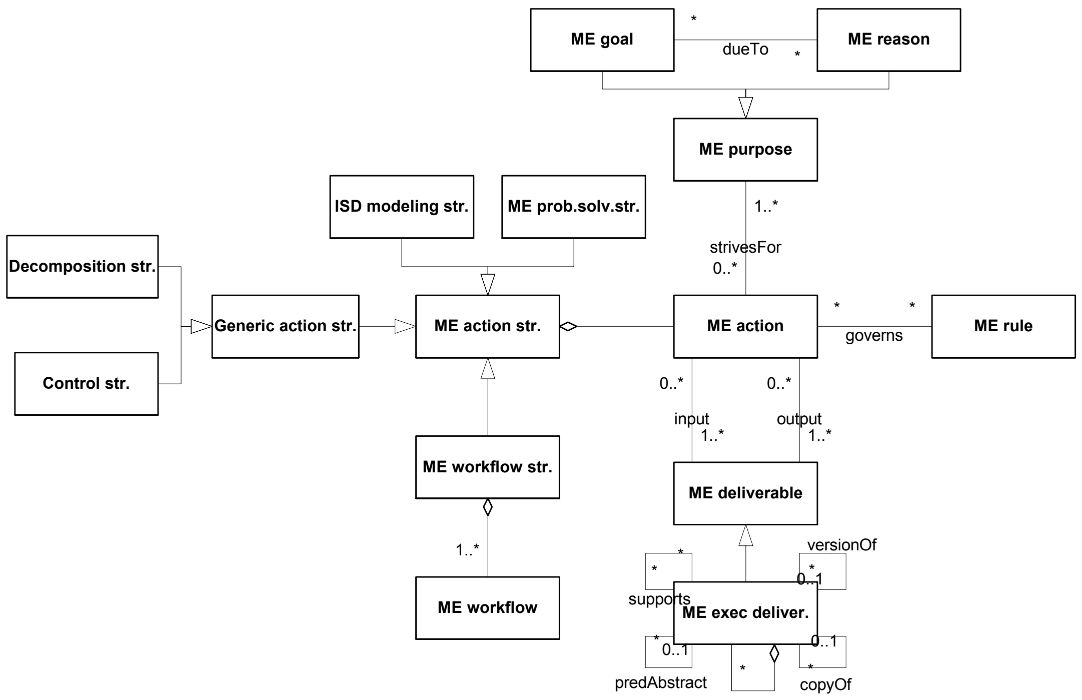


FIGURE 112 Meta model of the ME infological perspective

The ME reasons and the ME goals are used to express problems in the prior and current ISD projects, requirements for the new ISD method, and goals for the ME effort at hand. The influence, refinement, and causalTo relationships show how the concepts of the ME purpose domain are related to one another.

The ME actions are organized according to the generic action structures (i.e. the decomposition structure, the control strucutres), the ME problem solving structure, the ISD modeling structure, and the ME workflow structure. The ME management–execution structure as well as the ME management deliverables are not considered here. ME rules for ME actions are specified on a general level. The ISD method and method components on various granularity levels are, of course, the most essential ME execution deliverables. The partOf, versionOf, copyOf, supports and predAbstract relationships are recognized among the ME deliverables.

The ME infological perspective is most commonly applied in describing ME processes in the literature. Harmsen (1997), for instance, describes the process of situational method engineering composed of five steps. ME actions of, and ME deliverables resulting from, the steps are outlined. Harmsen (1997) also specifies the MEL language (Harmsen 1997), which provides constructs to administrate, query and manipulate method fragments. Tolvanen (1998) decomposes the process of incremental method engineering into six steps, which are described on a general level. Gupta *et al.* (2001) decompose ME into three phases and for each of the phases tasks and outcomes are outlined.

10.4.3 ME Conceptual Perspective

The *ME conceptual perspective* addresses the conceptual contents of the ME deliverables. Here, we consider the perspective only in relation to the ME execution deliverables. The conceptual contents of the ME deliverables contain constructs on multiple processing layers and on multiple model levels. To clarify this we first consider the perspective with Figure 113.

In Figure 113 the columns and the rows stand for the model levels and the processing layers, respectively. On each layer, the 'producer' context and the 'target' context are mentioned. For instance, on the ME layer, the producer (of deliverables) is the ME context and the target (to which the ME deliverables refer) is the ISD context. In the cells, examples of conceptual models are presented in a graphical form. Let us first consider examples at the type model level. The ISD data model (the second row and the leftmost column) is an ME deliverable expressing the conceptual contents of ISD deliverables, showing in this case that "a Conceptual schema is owned by a Database designer". The IS data model on the third row is a part of the conceptual schema of the IS database, expressing that "a Customer issues an Invoice". At the meta model level (the column in the middle) there are meta data models specifying concepts and constructs that are allowed in data models at the type model level. For instance, on the second row the IS meta data model contains the concepts EntityType and RelationshipType, of which Customer and Invoice on the third row are instances.

The topmost row corresponds to the conceptual contents of the deliverables that the RW context produces. One of the RW deliverables is the IS meta meta data model specifying the concepts and constructs allowable on the next lower level. The IS meta meta data model contains the concepts Class,

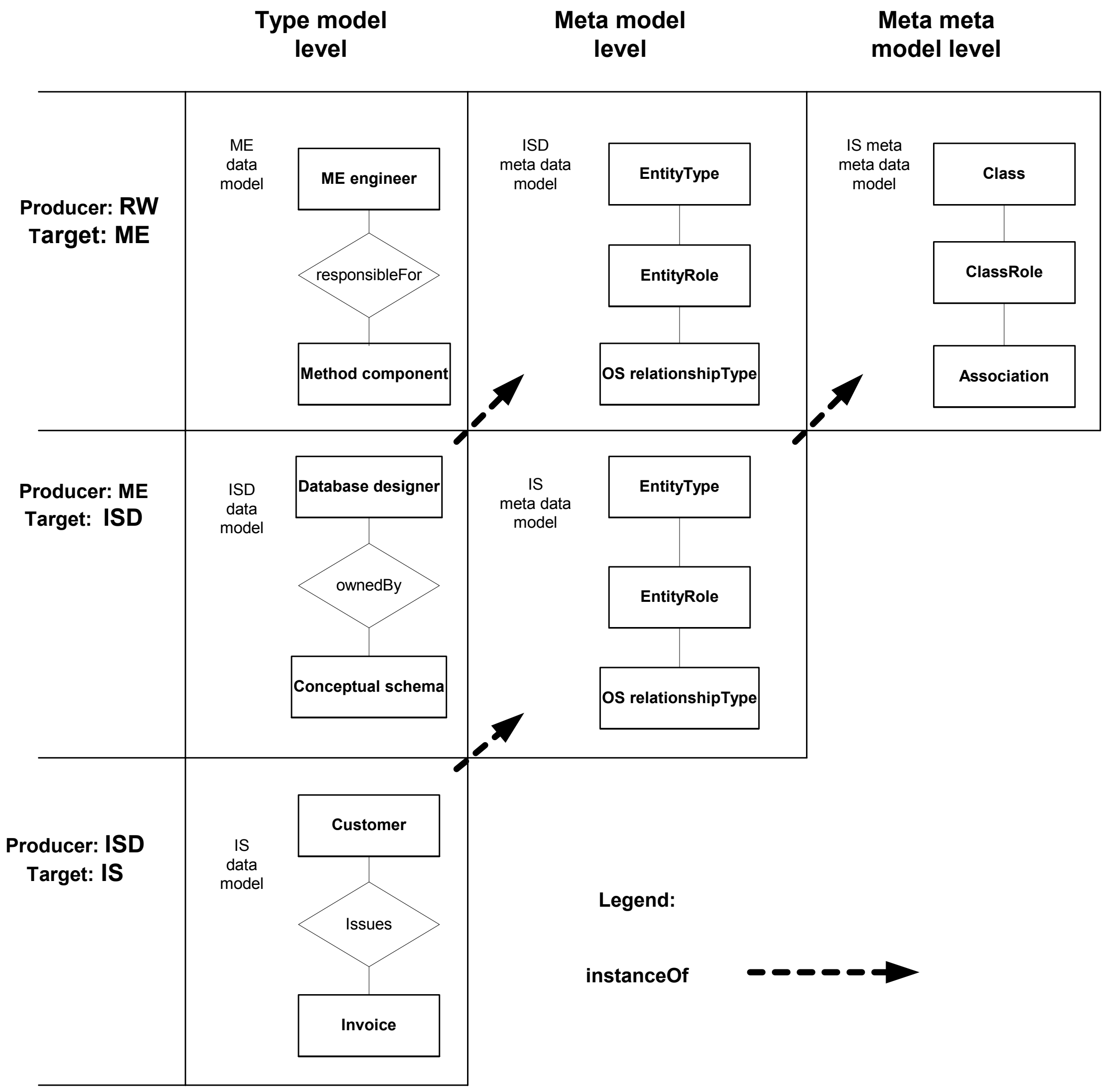


FIGURE 113 Processing layers and meta levels with some examples

ClassRole[167] and Association, according to the MOF model (OMG 2002). To help us distinguish between the (type) models, the meta models, and the meta meta models we have applied different graphical notations. The data models on the type level are presented in the ER notation (Chen 1976). The concepts and constructs of the ER model and the IS data meta meta model are denoted by the UML notation (Booch *et al.* 1999).

Based on the perspective ontology (Chapter 6), we can say that the ME conceptual perspective concerns the second row in the setting above. The perspective recognizes the concepts and constructs to be used at the type model level, as well as at the meta model level. At the type model level, there are a large number of ISD models. In Figure 113 we have used the ISD data model as an example of these models. Other possible ISD models are a data flow diagram

[167] We use the term 'ClassRole' to differentiate it from 'Role' in the generic ontology (Section 3.3) and 'EntityRole' in the ISD ontology (Section 8.4.3).

(Yourdon 1989) expressing ISD workflows and ISD deliverables, an organization chart showing the organizational structure of an ISD project, a deployment diagram (Booch *et al.* 1999) describing software and hardware architecture, etc. At the meta model level, there is the IS meta data model in Figure 113. Correspondingly, there could be several other IS meta models, such as the IS meta action model, the IS meta actor model, and the IS meta architecture model. It is impossible for us here to describe the ME conceptual perspective in a way that would cover all the ISD models and IS meta models. Therefore, we have selected only those models presented in Figure 113. To give a more detailed picture of the ME conceptual perspective, we present the meta models of the ISD data model and of the IS meta data model in Figure 114 (cf. the topmost row in Figure 113).

Figure 114 (a) presents some of those concepts and constructs with which the conceptual contents of the ISD deliverables (OS_{ISD}) can be specified (cf. the ISD data model). Because we apply the same meta data model as at the next lower level (see Section 8.4.3), the basic concepts (i.e. Entity type, Entity role, OS relationship type, Attribute) are applicable also here. We do not consider them any more here. An *OS_{ISD} construct type* in OS_{ISD} means a conceptual construct composed of specific entity types related to one another with OS relationship types and characterized by attributes. An example of the OS_{ISD} construct type is "a Conceptual schema is owned by a Database designer". An *OS_{ISD} state type* means a state type of the object system or its parts, composed of OS_{ISD} construct types. An *OS_{ISD} transition type* is a generic concept corresponding to the specification of all those features that are shared by OS_{ISD} transitions. An OS_{ISD} state type may involve entity types, OS relationship types and/or attributes. The OS_{ISD} transition types can be composed to establish OS_{ISD} transition structures like those defined in the state transition ontology. An *OS_{ISD} event type* means a generic concept corresponding to the specification of all those features that are shared by OS_{ISD} events, which may trigger an OS_{ISD} transition and which may be caused by another OS_{ISD} transition. An *OS_{ISD} constraint* specifies allowed OS_{ISD} states (static OS_{ISD} constraint) and/or allowed OS_{ISD} transitions (dynamic OS_{ISD} constraint) between the OS_{ISD} states.

Figure 114 (b) presents some of those concepts and constructs with which the conceptual contents of the IS meta data model can be specified. Because we apply the MOF model (OMG 2002) as the meta meta model in this work, the figure contains concepts such as Class, Class role, Association, and Attribute. We do not consider them here in more detail (see Appendix 2). We have also included one more concept in the IS meta meta model. That is an OS_{is} construct meta type. An *OS_{is} construct meta type* in OS_{ME} is composed of classes related through associations and class roles to one another. It is a type specification of those construct types defined in the meta model of the IS data model from the conceptual perspective in Figure 83.

From various portions of the ME conceptual perspective the IS meta meta models are most commonly considered in the ME literature. Several candidates for the IS meta meta models have been suggested (e.g. ER model (Chen

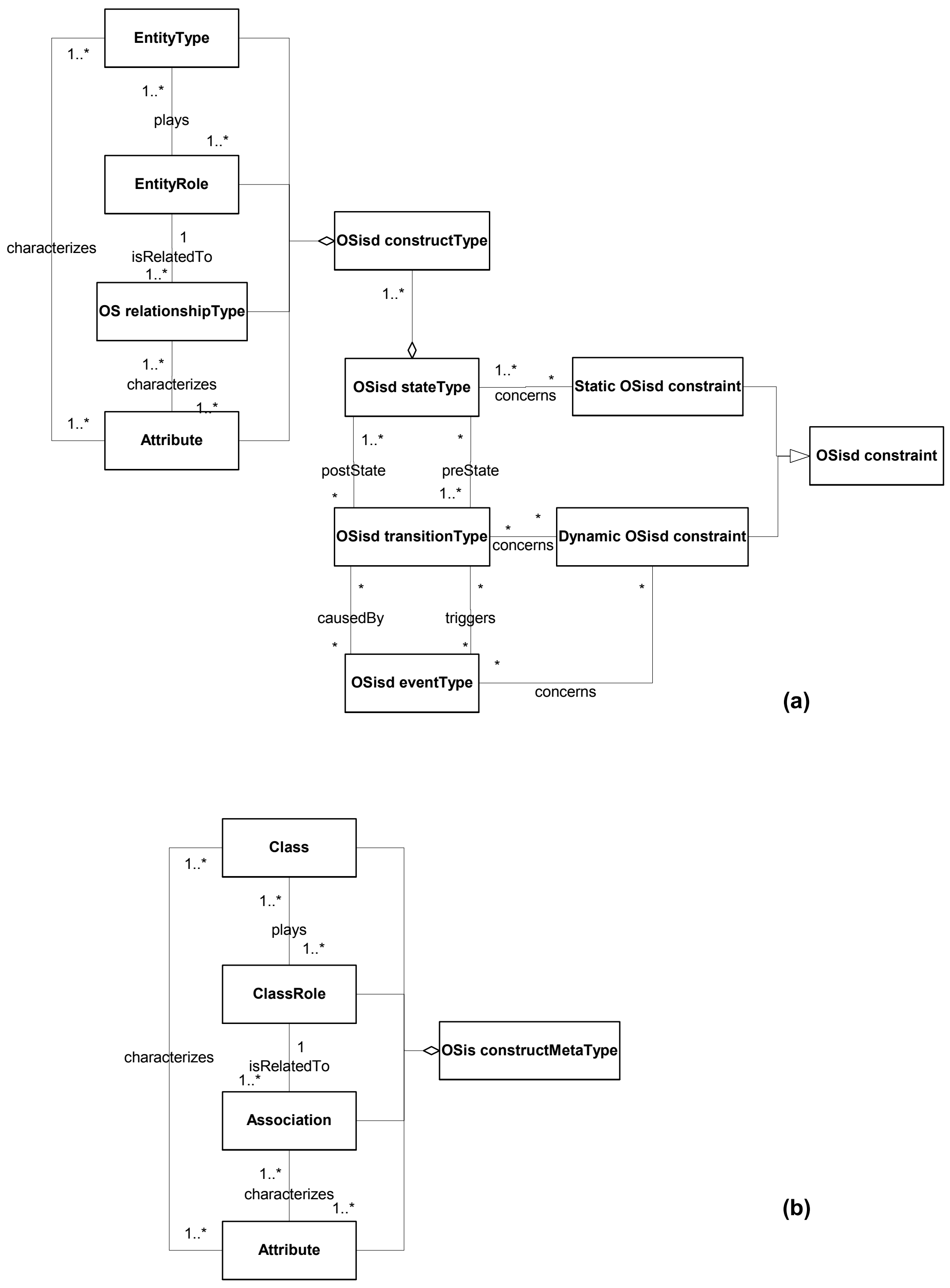


FIGURE 114 Meta models of the ME conceptual perspective concerning (a) the ISD data model and (b) the IS meta data model

1976, NIAM (Nijssen *et al.* 1989), ASDM (Heym *et al.* 1992a), GOPRR (Kelly *et al.*1996) and MEL/MDM (Harmsen 1997). There are also some presentations that specify ISD meta models, in particular those that provide concepts and constructs of the ISD action domain and the ISD deliverable domains (e.g.

Bandinelli *et al.* 1993; Deiters *et al*. 1994; Christie 1993; Shepard *et al.* 1992; Dutton 1993; Kaiser *et al.* 1993).

10.4.4 ME Datalogical Perspective

From the *ME datalogical perspective* ME is seen as the context in which ME deliverables, represented in some language, are processed for certain purposes by ME actors with some computer-aided ME tools. The ME datalogical perspective makes no reference to data carriers, nor to other physical things in the ME context. The perspective elaborates conceptions about the ME domains already recognized from the ME infological perspective. In addition, it addresses two other ME domains, namely, the ME actor domain and the ME facility domain. Since the number of the concepts needed to describe all the datalogical features of the ME context is huge, we discuss in the following only the most essential concepts and constructs, and from them only those which have not been addressed in the preceding sub-sections. The meta model of the ME datalogical perspective is presented in Figure 115.

ME actions are aggregated to constitute ME roles, which are further composed to form ME positions. An ME role equipped with skill requirements can be a part of several ME positions. An ME organization consists of ME positions that are related to one another through the supervision relationships.

In the ME action domain, the generic action structures, the ME workflow structure, the ME problem solving structure, and the ISD modeling structure are refined from those considered within the ME infological perspective. In addition, the ME management–execution structure as well as the ME phase structure are established to enable the understanding, structuring and representing of the management and coordination of the ME project. The ME phase structure is decomposed into ME sub-phases and ME steps. The relationships between phases and between sub-phases are based on the control structures, yet not on the temporal structures.

In the ME object domain, still more refined decompositions and specializations of ME deliverables, including the ME management deliverables, are recognized. Some ME execution deliverables are specified to be parts of the baselines of the phases.

In Figure 115 the concepts and constructs of the ME facility domain are abstracted into the notion of an ME tool (cf. CAME tools). It would be possible to present a more tool-centered view showing essential components of the tools, as well as the interaction, through dialogs, between the human actors and the tools (see Section 6.3.5). We have to exclude this view from our consideration here.

10.4.5 Inter-Perspective Relationships

The ME perspectives are inter-related to one another. Figure 116 illustrates the contents of the ME perspectives in terms of contexts and domains concerned, as well as the relationships between the ME perspectives. To

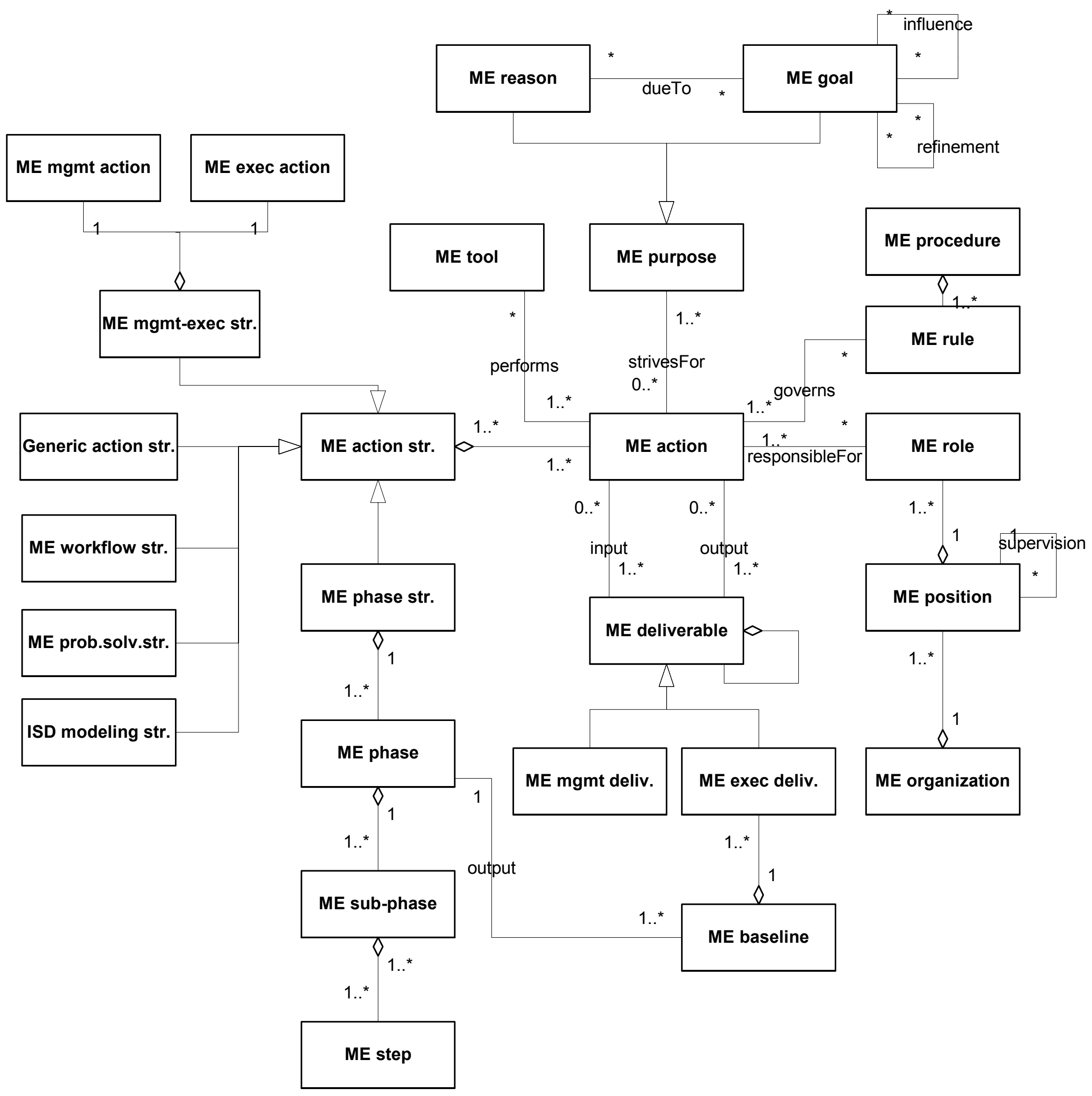


FIGURE 115 Meta model of the ME datalogical perspective

distinguish between the concerned contexts, we depict them with rectangles in bold whenever there are more than one context involved by the ME perspective. As said above, the ME systelogical perspective involves four kinds of contexts, the ME context, the ISD contexts, the IS contexts, and the US_{IS} contexts. The corresponding domains are shown in the figure. The ME infological perspective, the ME datalogical perspective, and the ME physical perspective concern the ME context only. The ME conceptual perspective designates, on a general level, things to which the ME deliverables, including the ME management deliverables, refer. For each relevant thing both the structural and dynamic features are identified. The ME deliverables are recognized and conceptually elaborated within four perspectives. In each of them, the contents of the deliverables can be specified and analyzed through the conceptual foundation provided by the ME conceptual perspective.

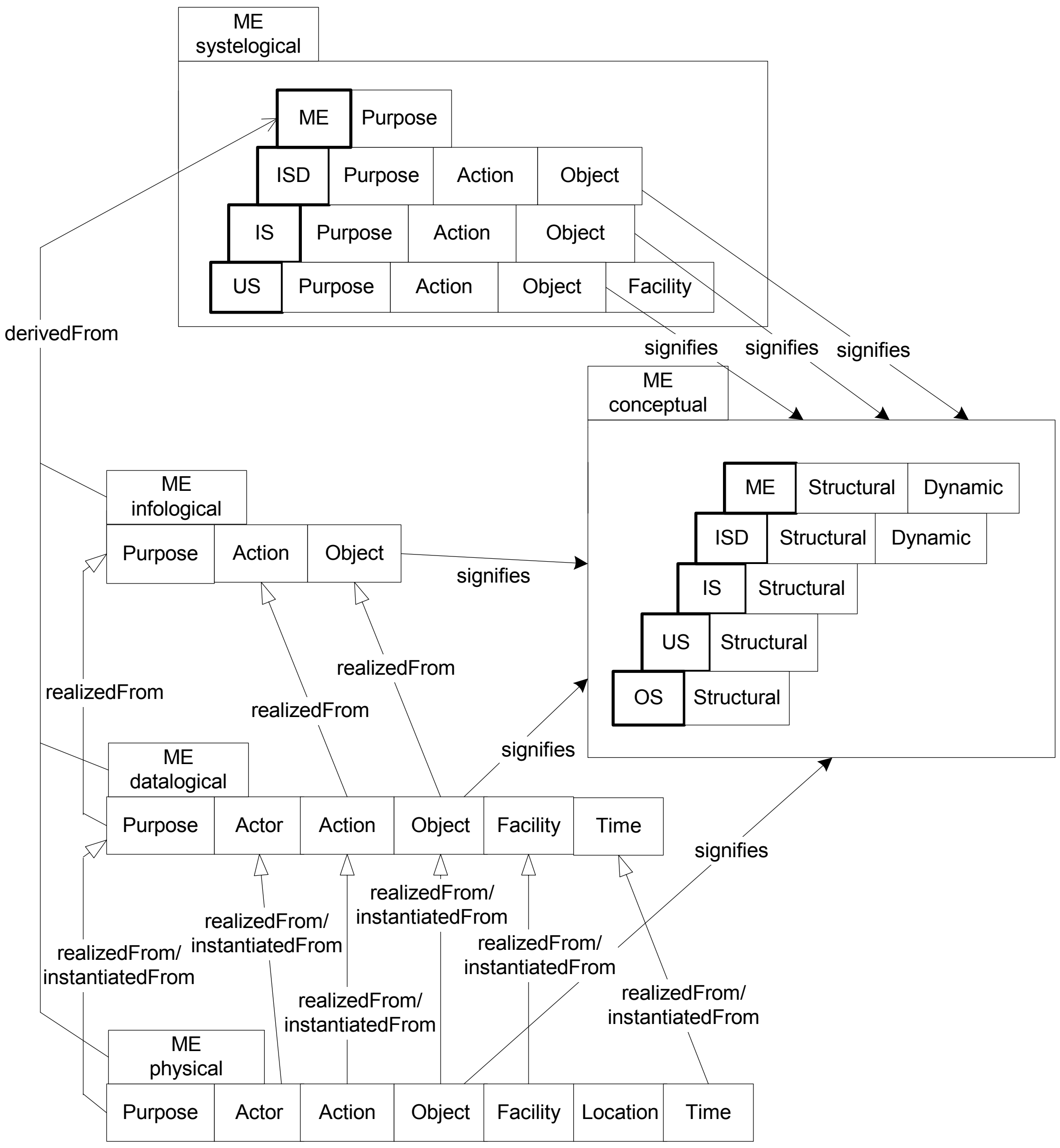


FIGURE 116 ME inter-perspective relationships

10.4.6 Summary

In this section we defined four ME perspectives (i.e. the ME systelogical perspective, the ME infological perspective, the ME conceptual perspective, and the ME data perspective) to help us cope with the complexity related to the structure, function and behavior of the ME context. For each perspective we provided the concepts and constructs and described them in meta models. In addition, we referred to the ME literature to show how issues pertaining to the ME perspectives are addressed. At the end of the section, we outlined the ME inter-perspective relationships.

10.5 ME Method Ontology

The purpose of this section is to characterize and define the notion of an ME method as well as to present the ME method ontology. We had a comprehensive discussion about the notion of an ISD method in Chapter 10. Because the ME method inherits the generic features from the ISD method, we consider these issues here only shortly.

10.5.1 Definition of the ME Method

The method contains collective knowledge and experience that are made 'visible' to enable their exploitation and advancement (Tolvanen 1998; Fitzgerald *et al.* 2002; Schönström *et al.* 2003; Backlund *et al.* 2003). In the context of ME this knowledge concerns ME process, application domain, IC technology, and human and social issues (cf. Freeman 1987; Iivari *et al.* 2001). The knowledge of ME process means all that information that pertain to how to accomplish ME work. The knowledge of application domain means all the information that concerns ISD efforts to be accomplished according to the ME method (i.e. the target ISD contexts), as well as information systems and contexts for which the IS's are to be developed. The knowledge of IC technology means all that information that concerns the search, acquirement, installation, and deployment of hardware and software for IS's, ISD's and ME. Finally, the knowledge of human and social issues concerns human characteristics and behavior as well as social and organizational aspects that should be taken into account in prescribing the ME.

Following the categorization of the ISD methods defined in Section 9.3, we distinguish between generic ME methods, domain-specific ME methods, organization-specific ME methods, and project-specific ME methods. A *generic ME method* provides general approaches, principles, models and guidelines to conduct ME efforts in a wide range of ME contexts. A *domain-specific ME method* provides more domain-specific support to conduct ME efforts in a specific application domain. An *organization-specific ME method* provides customized support to conduct ME efforts in a specific organization. A *project-specific ME method* provides configured and instantiated support to accomplish a particular ME effort.

In Section 9.4 we defined seven methodical views from which the notion of an ISD method can be conceived and understood. The views are the historical view, the application view, the generic view, the contents view, the representation view, the physical view, and the structural view. Each view sheds light on different aspects of the method. Here we apply the views to clarify the notion of the ME method in two ways. First, we present a holistic definition of the ME method, and then we define the ME method ontology.

As far as we know, no definition of the ME method is provided in the ME literature. That is quite surprising given that ME has been seen important for years. Actually, although a large collection of various ME strategies, ME approaches, ME principles and ME procedures have been suggested, there exists no complete ME method either, as we will conclude in the comparative analysis in Chapter 12. We define the notion of the ME method, based on the methodical views and the definition of the ISD method (see Section 9.4), as follows:

> An *ME method* is an artifact anchored on historical, intentional and functional backgrounds and aimed to be applied and deployed as a prescription in the intended kinds of ME contexts, in order to make organizational and technical changes in ISD contexts possible or more productive. The ME method, presented and materialized in several forms, contains knowledge bringing out how ME actors carry out ME actions to produce ME deliverables, by means of ME facilities, in an organizational and spatiotemporal context, in order to satisfy ME goals set by ME stakeholders. The ME method is composed of descriptive and prescriptive parts in a large variety.

In the next section we elaborate this definition by showing which concepts and constructs are embodied by each of the methodical views in the context of ME.

10.5.2 ME Method Ontology

The *ME method ontology* provides concepts and constructs for conceiving, understanding, structuring and representing contextual aspects of the ME methods. It is decomposed into seven parts based on the seven methodical views. The overall structure of the ME method ontology is presented in Figure 117. Next, we describe the views and define concepts involved by them.
The historical view considers the backgrounds of and experience from the engineering and use of the ME method. It involves prior RW contexts, prior ME contexts and prior ISD contexts. *Prior RW contexts*[168] mean those contexts that have contributed to the creation and engineering of the ME method. *Prior ME contexts* mean contexts in which the ME method has been deployed. Implied from the former, the ME method has to include knowledge of the intentions, approaches and principles by which the ME method has been constructed, of the RW actors who have been responsible for the construction, and of the RW actions by which the ME method has been constructed, etc. This knowledge is called method engineering rationale (Rossi *et al.* 2004). The descriptions of the prior ME contexts constitute an "experience base" which helps make and justify decisions on whether to use the ME method and how. This knowledge is known as method use rationale (Rossi *et al.* 2004). Included in the background is also knowledge about ISD contexts, called *prior ISD contexts,* where ISD methods engineered with the ME method have been applied. This knowledge is

[168] We call these contexts the RW (Research Work) contexts to distinguish them from those contexts where the ME methods are deployed. Another way to differentiate the two notions would be to use the terms ME^2 context and ME^1 context, correspondingly.

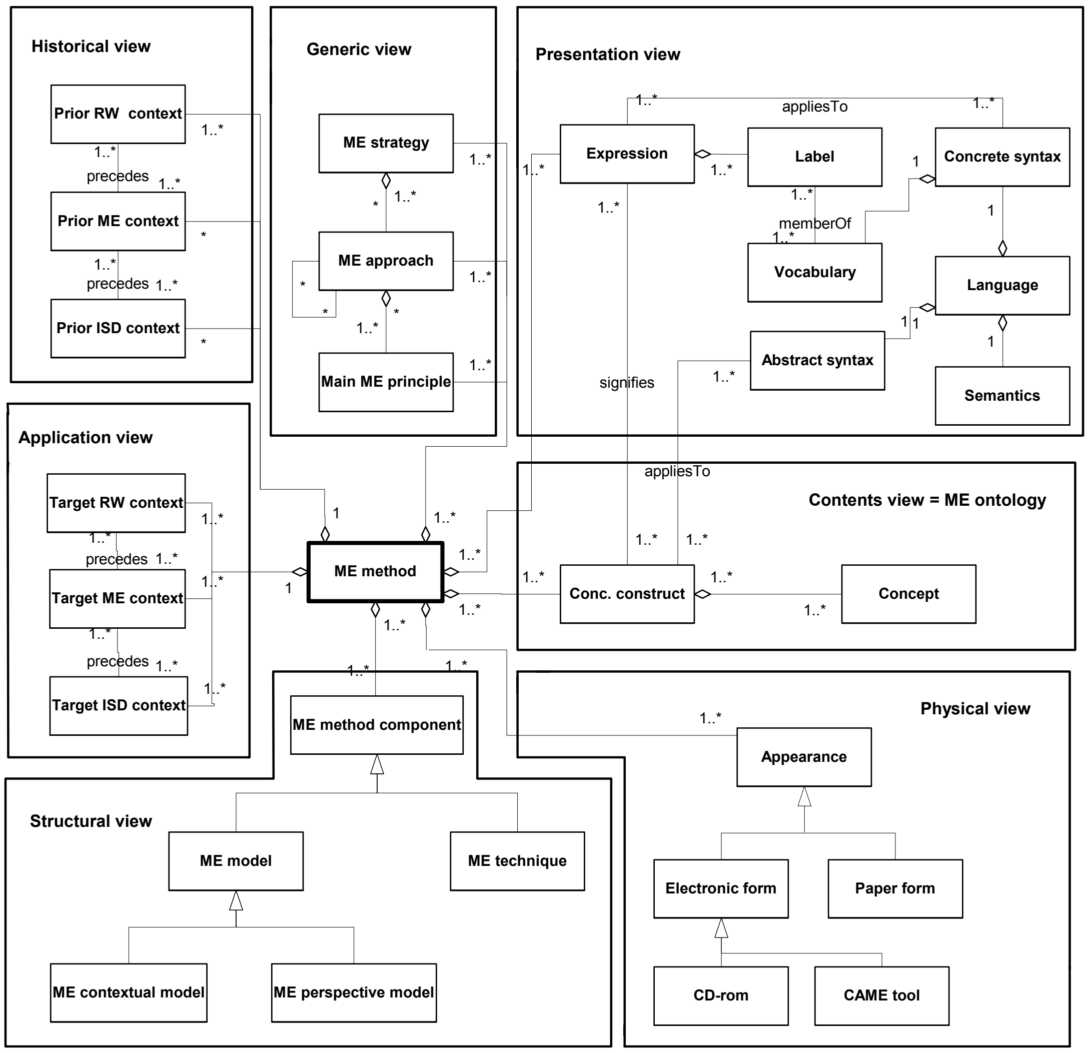


FIGURE 117 An overall structure of the ME method ontology

important because the quality of the ME method is an aggregation of the qualities of ME processes and ME deliverables (i.e. ISD methods), and the quality of ISD methods, in turn, can be empirically assessed only by analyzing experience obtained from their deployment in some ISD effort(s).

The application view outlines where and how the ME method can be or is to be applied. The target contexts can be recognized on three layers in the ME method ontology. *Target RW contexts* mean those contexts in which the ME method is to be elaborated, customized, configured and/or instantiated for the use of a particular organization or project. *Target ME contexts* mean those contexts for the use of which the ME method is originally intended. *Target ISD contexts* mean those contexts which are to deploy the ISD method that will be engineered in the target ME context. The arguments for the applicability to certain kinds of RW contexts, ME contexts and ISD contexts should be justified with appropriate evidence. Evidence can be based on logical arguments derived from the perceived match between the ME method and the suggested application areas, or on empirical experience got from the prior usages.

The generic view provides the general understanding of the nature of the method. In this case it reflects ME strategies, ME approaches, and main ME principles to be followed in the target ME contexts. An *ME strategy* means a generic way of accomplishing an ME effort or a part thereof. Examples of ME strategies are creation (i.e. "from scratch"), integration, and adaptation (cf. Section 10.2). An *ME approach* means a generic way of perceiving certain aspects of ME and/or a way of working in ME. Examples of the ME approaches are: problem-driven approach, functional approach, and conceptual approach (see Chapter 11). A *main ME principle* expresses essential aspects of a specific way to structure, accomplish and/or manage the ME process. Examples of the main ME principles are iterative engineering and contingency-based engineering. The ME strategies, ME approaches and main ME principles are inter-related to one another.

The contents view reveals the conceptual contents of the ME method. According to the view, the ME method is composed of concepts and conceptual constructs referring to the ME contexts, as well as to parts of the RW context(s). The former contexts correspond to the prior and target ME contexts. The conceptual contents of the ME context have already been established in the form of the ME ontology in the preceding sections. The latter contexts mean the prior and target RW contexts. The conceptual contents of the RW context is, to a large extent, similar to that defined in the ME ontology, in particular for the part we are here interested in. That is why we do not provide any separate ontology for the RW context.

From the presentation view the ME method is seen as a set of expressions presented in some language(s). Expressions signify conceptual constructs constituting the contents of the ME method. Each language is defined by the abstract syntax, concrete syntax (or notation) and semantics. Abstract syntax states allowed conceptual constructs composed of concepts (ter Hofstede *et al.* 1998). Concrete syntax gives notational elements, including labels, of the language and rules for connecting them with one another and with the concepts. Semantics specifies meaning of notational elements.

The physical view reveals the appearance of the ME method, that is to say, the media on which the ME method is made visible or "functioning". The ME method can appear in a paper form (e.g. text books, manuals, pro forma documents), or in an electronic form (e.g. CD-rom, Word-Wide Web). It can be presented with e.g. Power Point slices and implemented in CAME tools. The CAME tools may support the creation and editing of ME models, their implementation, ME process management, and ME process enactment.

From the structural view the ME method is seen as a modular structure of parts with a large variety. Some of the parts are considered ME method components. An *ME method component* is a well-defined part of the ME method that can be integrated to other ME method components to form a coherent and consistent ME method. In Figure 117 we recognize two kinds of ME method components: ME models and ME techniques. Because the discussion about the notions of method component and interface in Section 9.8 also applies to the ME method, we do not consider them any further here.

An *ME model* is a model that describes/prescribes structural, functional and/or behavioral features of the ME context. An *ME technique* is a technique, which guides the accomplishment of specific actions in the ME context. The technique can be presented as a set of precisely described procedures that help the achievement of certain outcomes if executed correctly (cf. Kettinger *et al.* 1997, 58; Iivari *et al.* 2001, 186). ME techniques may also be presented in heuristics, guidelines or rules of thumb. An ME technique may involve one or more ME models (e.g. a technique to integrate ISD models, e.g. O/A matrix technique in Kinnunen *et al.* (1996)), and an ME model can be involved by one or more ME techniques.

ME models are further specialized into ME contextual models and ME perspective models. *ME contextual models* mean ME models that can be classified into eight categories according to which ME domain(s) they address. The categories are: ME purpose models, ME actor models, ME action models, ME deliverable models, ME data models, ME facility models, ME location models, ME time models, and ME inter-domain (ID) models. The nature and contents of the ME contextual models can be, in a straightforward manner, derived from those presented for the ISD contextual models (see Section 9.5).

ME perspective models mean ME models that can be classified into five categories according to the ME perspective(s) they address. The categories are: ME systelogical models, ME infological models, ME conceptual models, ME datalogical models, ME physical models, and ME inter-perspective (IP) models. The nature and contents of the ME perspective models can be derived from those presented for the ISD perspective models (see Section 9.5).

10.6 Summary

The purpose of this chapter was to establish the conceptual foundation of the method engineering and the ME method. This foundation is vital to the analysis of ME efforts in practice, to the analysis and comparisons of empirical and conceptual studies on ME, as well as to the construction of methodical support to ME.

In the chapter we first brought out the reasons and motives for why method engineering is needed. We made a survey of the ISD literature reporting on problems in ISD methods and method use and discussed how the evolution and changes in business, application areas, and approaches and technologies of ISD environments influence ISD. We concluded that severe problems have been perceived in the implementation and deployment of the ISD methods, and some of those problems result from drawbacks and deficiencies in existing methods. Resulted from changes in business processes, application areas and technology, needs for new kinds of methods have also emerged. These together set high demands on method engineering.

Second, we demonstrated, through a short survey of the ME literature, that there is a large variety of conceptions about ME, and quite different terms are used in the community. The need to clarify the conceptual bases and terminology of ME became clear. To satisfy this need we defined fundamental classifications of ME strategies, ME processes and ME contexts. We also presented a framework which integrates these classifications and enables the specification and analysis of a large array of ME contexts. In addition, we applied the contextual approach to construct the holistic definition of the ME context. According to it, ME is seen as a context which possesses features of seven contextual domains, and is connected to different types of contexts on several layers.

Third, we defined the first part of the ME ontology that is composed of four ME domains (i.e. the ME purpose domain, the ME actor domain, the ME action domain, and the ME object domain). For each domain the meta model and definitions of concepts and constructs were presented. In addition, the ME literature was frequently referred to and compared to our concepts and constructs.

Fourth, we provided the second part of the ME ontology comprising four ME perspectives (i.e. the ME systelogical perspective, the ME infological perspective, the ME conceptual perspective, and the ME datalogical perspective). The ME perspectives help us cope with the complexity related to the structure, function and behavior of the ME context. For each perspective the concepts and constructs were defined. In addition, references to the ME literature were made to show how issues pertaining to the ME perspectives are addressed there. At the end of the section, the ME inter-perspective relationships were outlined.

Fifth, we characterized and defined the notion of an ME method and presented the ME method ontology. Both the definition and the ontology were derived applying seven methodical views established in Section 9.4. The use of the views ensures that no important aspects of the ME context and of the ME method are excluded from the considerations.

The ME ontology and the ME method ontology are the lowest-level components in OntoFrame. Discussions and definitions given in this chapter completed the construction of the multidimensional framework that is aimed as a conceptual foundation for the analysis, comparison, and engineering of ISD methods. We did not make any unified comparative analysis of the ME literature in this chapter, because there is no comparable presentation available for these ontologies. Instead, we provided a number of references to the ME literature on individual issues and compared conceptions and terminology with ours. To have a more comprehensive picture about existing normative ME artifacts, we will analyze and compare them with MEMES in Section 12.4.

OntoFrame has been built in a layer-by-layer fashion, anchoring each ontology firmly on underlying ontologies and applying comprehensively fundamental approaches, particularly the contextual approach. In this way we have wanted to assure the coherence, consistence and modularity of the framework. OntoFrame has been constructed by concepts and constructs that reflect common conceptions shared by various communities. This enables its use for divergent purposes. To demonstrate the applicability of OntoFrame in the construction of artifacts, we will deploy it in the engineering of a methodical skeleton for ME (MEMES) in the next chapter. In this work we will extensively utilize the concepts and constructs defined in the ME ontology and the ME method ontology.

11 MEMES - METHODICAL SKELETON FOR ME

In the previous chapters we have built, piece-by-piece, OntoFrame to cover contextual features of reality from multiple perspectives, on multiple layers and on multiple model levels. The lowest parts in this ontology framework are the ISD ontology, the ISD method ontology, the ME ontology, and the ME method ontology. The purpose of these parts is to offer concepts and constructs for the analysis and construction of artifacts on the ISD and ME layers. In this chapter we will use these parts to construct a methodical skeleton to support the accomplishment of the process of ME. The methodical skeleton, called MEMES, is firmly grounded on the ontological framework. This becomes evident in the suggested approaches, principles, concepts and constructs.

The chapter is organized into ten sections. First, we justify the need for methodical support for ME. Second, we define MEMES in terms of its intention, basis and contents. Also relations between MEMES and OntoFrame are, in a concrete fashion, demonstrated. Third, we highlight the background of MEMES describing those contexts on the ME and ISD layers which have affected the construction of MEMES. Fourth, we specify the application area for MEMES. Fifth, we state the goals of MEMES. Sixth, we present the overall structure of MEMES in terms of ME workflows. In the next three sections we describe three of those ME workflows (i.e. the ISD method requirements engineering, the ISD method analysis, and the ISD method evaluation). Descriptions of the first two workflows are more detailed including approaches and steps of engineering an ISD method. The chapter ends with a summary.

11.1 Need for Methodical Support to ME

Method engineering has become more and more vital in practice, as concluded in Chapter 10. At the same time, most of the ME efforts in practice are accomplished in an ad-hoc manner. So, we have not progressed very far from the stage that is known as the pre-methodological era in the ISD field (cf.

Hirschheim *et al.* 1995). The ME literature does provide a large variety of artifacts for ME. There are, for instance, numerous meta models and metamodeling languages that can be used to model methods, or parts thereof. Meta data models (e.g. ER (Chen 1976), NIAM (Nijssen *et al.* 1989), OPRR (Smolander 1991), ASDM (Heym *et al.* 1992a, Heym *et al.* 1992b), CoCoA (Venable 1993), GOPRR (Kelly *et al.* 1996), Telos (Jarke *et al.* 1995) and MEL/MDM (Harmsen 1997)) are suggested to model the conceptual contents and notations of data models, and meta process models (e.g. Bandinelli *et al.* 1993; Deiters *et al.* 1994; Christie 1993; Shepard *et al.* 1992; Dutton 1993; Kaiser *et al.* 1993) are provided to model process models. There are also proposals for ME strategies (e.g. Ralyte 2002; Ralyte *et al.* 2003), ME approaches (e.g. Kumar *et al.* 1992; Oei 1995; Harmsen 1997), and ME techniques (e.g. van Slooten *et al.* 1993; Kinnunen *et al.* 1996; Leppänen 2000; Saeki 2003). Further, the ME literature contains ME artifacts that provide an overall structure of ME processes, or alternatively simplified procedures for the accomplishment of some parts of ME work (e.g. Vlasblom *et al.* 1995; Nuseibeh *et al.* 1996; Song 1997; Harmsen 1997; Tolvanen 1998; Gupta *et al.* 2001; Ralyte *et al.* 2003). Finally, there are presentations that describe actual processes of engineering a method (e.g. Song *et al.* 1992; Mayer *et al.* 1995; Vidgen 2002; Polo *et al.* 2002; Fitzgerald *et al.* 2003; Serour *et al.* 2002; Backlund *et al.* 2003, Bajec *et al.* 2004).

Regardless of the large variety of the ME artifacts proposed, there is no single artifact that could be seen to come even close to an ME method as the notion is generally understood, and as it has been defined in this work. Either the artifacts are on too a general level, or they cover only a small part of the ME life cycle. To justify this claim, a comprehensive literature analysis was carried out (see Section 12.4). Due to the significance of ME to ISD practice and the scarcity of methodical support to it, our aim is to provide a methodical support that is more comprehensive and conceptually more uniform than any other artifact in the ME literature. This support has been "packaged" and served in the form of a methodical skeleton.

11.2 Definition of the ME Methodical Skeleton

In this section we define the notion of the ME methodical skeleton and describe its intention, basis and contents.

The *method engineering methodical skeleton,* called MEMES, is a normative prescription of the ME context that structures and guidelines the accomplishment of ME work. According to Webster's Dictionary, a skeleton is something that "is reduced to the essential parts". The purpose of the ME methodical skeleton is to provide the essential parts of prescribing the ME context. MEMES is an abbreviation from the phrase 'Method Engineering

MEthodical Skeleton'. Interestingly, in Memetics memes[169] mean the basic building blocks of our minds and culture, in the same way, as the genes are the basic building blocks of biological life. Memes are ideas, habits, skills, stories or any kind of behavior or information that is copied from person to person by imitation (Blackmore 2000). A well-known behavioral meme is, for instance, "how to make a fire". Once the meme was out there, it spread like wildfire. Individual slogans, catch-phrases, melodies, icons, inventions, and fashions are typical memes (Dawkins 1976). An idea is not a meme until it causes someone to replicate it, to repeat it to someone else. This is what we hope our MEMES will do: to bring out memes of method engineering that can be adopted and effectively deployed in practical ME work.

MEMES is firmly grounded on the ontological framework defined in Chapters 3 - 10. Its intention, basis and contents can be illustrated in relation to this framework. In Figure 118 the left side describes MEMES in its intentional and functional environment (cf. Figure 94). Research work, here referred to as the RW context, has produced MEMES, which is to be applied in an ME context for the engineering of an ISD method. The ISD method, in turn, is to be applied in an ISD context to develop an IS. The right side in the figure describes the structure of OntoFrame from the viewpoint of the ME method ontology. Included in the ME method ontology there are the methodical views (i.e. the historical view, the application view, the generic view, the presentation view, and the physical view) of the ME method, the ME ontology, the ISD method ontology, the ISD ontology, and the IS ontology. The ME ontology, the ISD ontology and the IS ontology are composed of seven contextual domains and five perspectives.

The arrows denote how OntoFrame has been deployed in the engineering of the components of MEMES, ISD methods, and IS models. The structure of MEMES has been adapted from the ME method ontology. The main components of MEMES are the methodical views, ME models, ISD meta models, and IS meta models. The historical view, the application view and the generic view have been given specific contents. The ME models have been specialized and instantiated from the ME meta models corresponding to the ME domains and the ME perspectives. The ME models describe / prescribe what is done, for which and why in the ME context. The concepts and constructs of the ISD meta models have been selected and adapted from those belonging to the ISD ontology. Likewise, the concepts and constructs of the IS meta models have been chosen and adapted from those belonging to the IS ontology. This process of instantiation, specialization, and adaptation can be accomplished in any

[169] Oxford zoologist Richard Dawkins is credited with the first publication of the concept of a meme in his 1976 book The Selfish Gene (Dawkins 1976). Currently the research field concerning memes is called Memetics. There are a large variety of definitions of Memetics (cf. Brodie 1996). Generally speaking, Memetics is the science that studies the replication, spread and evolution of memes (http://pespmc1.vub.ac.be/MEMES.html).

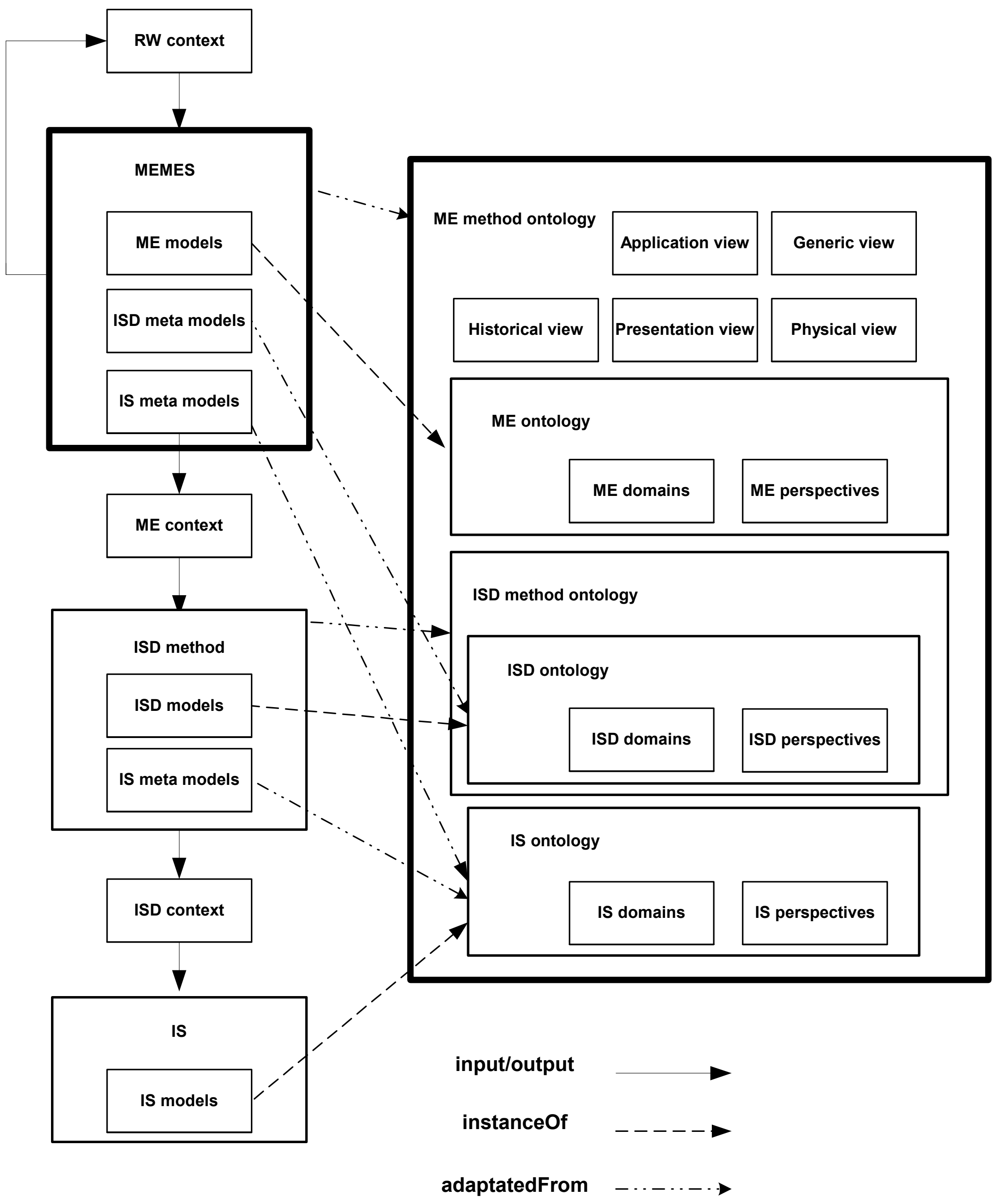


FIGURE 118 Intention, basis and contents of MEMES

situation which aims to engineer ME methods, or a part thereof, with the help of OntoFrame.

Figure 118 also shows, on a general level, how to engineer the ISD method in the ME context. The structure of the ISD method is adapted from the ISD method ontology in the following fashion. The ISD models are specialized and instantiated from the ISD meta models concerning the ISD domains and the ISD perspectives. The concepts and constructs of the IS meta models are selected and adapted from those belonging to the IS ontology. Finally, IS models are specialized and instantiated from the IS meta models.

Consequently, MEMES provides descriptions / prescriptions covering the methodical views, a set of ME models in the form of prescriptions, and a collection of ISD meta models and IS meta models, all in a well-structured and integrated body. MEMES integrates and gives normative meanings for the ISD meta models as well as instantiates and specializes the ME meta models. MEMES is not, however, an ME method. To elaborate the scope and contents of MEMES we apply the perspective ontology on two processing layers. Table 32 presents three ME perspectives (rows) and three ISD perspectives (columns). The cells in the table express questions reflecting the issues which are addressed in the ME context. Next, we consider those issues in the order of ME perspectives.

TABLE 32 Issues addressed in the ME context, structured through ME perspectives and ISD perspectives (Syst = systelogical, Info = infological, Conc = conceptual)

ISD perspectives/ ME perspectives		**Systelogical** Why is ISD accomplished and for whom?	**Infological** What are the ISD actions undertaken, and what ISD deliverables are produced?	**Conceptual** What is it that the ISD information refers to?
Syst	Why is ME accomplished and for whom?	What are the ISD purposes that ME is accomplished for and why?	What are the ISD actions and the ISD deliverables that ME is accomplished for and why	What are the IS domains that ME is accomplished for and why?
Info	What ME actions are done and what ME deliverables are produced?	What is done in ME to specify ISD purposes?	What is done in ME to specify ISD actions and ISD deliverables?	What is done in ME to specify IS domains (i.e. IS ontology)
Conc	What is it that the ME information refers to?	What are the concepts that are used to refer to ISD purposes?	What are the concepts that are used to refer to ISD actions and ISD deliverables?	What are the concepts that are used to refer to phenomena in the ISD contexts?

In applying the ME systelogical perspective one tries to find out the kinds of ISD contexts that MEMES is targeted to. The ISD contexts can be characterized in terms of ISD purposes (ISD systelogical perspective), ISD actions and ISD deliverables (ISD infological perspective), and conceptual contents of the ISD deliverables (ISD conceptual perspective). The ME infological perspective pertains to the ME actions and the ME deliverables needed to yield and present descriptions / prescriptions from the ISD systelogical, ISD infological, and ISD conceptual perspectives (cf. Section 8.4). For example, to yield descriptions from the ISD infological perspective it is necessary to elaborate ISD purposes and to specify main ISD actions and ISD deliverables. The ME conceptual perspective is related to the ontologies needed to perceive the ISD context from each of the five ISD perspectives. For example, it is decided what concepts and constructs are needed for understanding, structuring and presenting of ISD actions and ISD deliverables.

On the basis of Table 32 we can now specify the scope of MEMES and the level of detail in which we describe MEMES. First, MEMES is mainly described from the ME systelogical, ME infological and ME conceptual perspectives. The ME systelogical perspective covers the historical view, the application view, and the generic view. In addition, it addresses, on a general level, the presentation view and the physical view. The ME infological perspective covers those ME models that describe/prescribe ME purposes, ME actions and ME deliverables, as well as inter-domain relationships between them. The ME conceptual perspective exposes the ME ontology, the ISD ontology, and the IS ontology. Second, MEMES adresses only those parts of the ISD ontology that are related to the ISD systelogical, ISD infological and ISD conceptual perspectives. Together these two statements imply that we describe in which ME situations, under which kinds of requirements and goals, certain kinds of ME actions are carried out to yield ME deliverables that refer to ISD purposes, ISD actions and ISD deliverables. We do not discuss who should do what in ME, in which temporal order and where. Neither we take into account what tools and resources are needed in ME work. The IS meta models are not explicitly presented in this chapter, because they are assumed to be available in the underlying ontological framework (OntoFrame). In practice, there is a large variety of ME contexts. MEMES has not been customized to fit any particular type of ME context. The concepts and constructs are organized in a way that enables us to make easy adaptations into MEMES in order to get it follow any of the main ME strategies (i.e. creation, integration, adaptation).

The description of MEMES is structured into sections according to the three ME perspectives in the following way. First, we will describe the background of MEMES (cf. the historical view, Section 11.3). Second we will introduce the application area as well as the basic assumptions and approaches (cf. the application view and the generic view, Section 11.4). Third, we will formulate more precisely the goals of MEMES (Section 11.5). Fourth, we will present the overall structure of the ME workflows in MEMES (Section 11.6). Fifth, we will describe the ME actions and ME deliverables in the first, second and third ME workflow. These workflows are called, respectively, the ISDM requirement engineering (Section 11.7), the ISDM analysis (Section 11.8), and the ISDM evaluation (Section 11.9). The chapter ends with a summary.

11.3 Background of MEMES

According to the method ontology each method has to contain knowledge of the background of, and the experience from, the engineering, as well as on the use of that method (cf. the historical view in Section 9.5). Although MEMES is not a complete ME method, we describe the intentions of, work for, and evolution of MEMES in this section.

A short description of the cyclic process to engineer MEMES was already given in Section 1.4. The process was characterized as highly iterative crossing four subfields: ME practice, evaluation, ME method engineering, and ontology engineering. ME practice stands for all those efforts in which the researcher has been involved to engineer an ISD method. Evaluation comprises two kinds of RW actions: (1) reflecting and analyzing experience from ME practice, and (2) making comparative analyses of definitions, classifications, models, frameworks, and methods presented in the literature. ME method engineering and ontology engineering here mean theoretical work aimed to construct conceptual artifacts, either descriptive (cf. ontologies) or prescriptive (cf. models and techniques), that suit as parts for MEMES. The process reflects a learning cycle (Checkland 1981, 254), in which each RW deliverable (i.e. an ontology, a ME technique, and a methodical skeleton) is created, applied, learned from and further refined.

The historical view of MEMES involves prior contexts at three processing layers, namely prior RW contexts, prior ME contexts, and prior ISD contexts (cf. Section 9.5). Prior RW contexts mean contexts that have contributed to the creation and engineering of MEMES. Prior ME contexts mean contexts in which MEMES or its parts, in some of its versions, have been deployed or from which experience have been used in the RW context. Prior ISD contexts mean contexts in which an ISD method, engineered in some of the prior ME contexts, have been used to develop an IS. In the following we shortly describe the prior contexts of MEMES (Figure 119).

The first modest effort in method engineering was carried out in 1980-1984 to develop a language and a "design model" for conceptual schema design, called CSDM (Conceptual Schema Design Model), based on a linguistic approach (Leppänen 1984a). The design model was composed of generic constructs including the workflow structure, the system decomposition structure, abstraction structures, and the problem solving structure. The model was built on two layers: the "frame layer" consisting of generic concepts and constructs, and the "core layer" standing for instantiated concepts and constructs. The idea was that at the beginning of a design effort the core layer is empty, and in engineering a method for conceptual schema design, the generic concepts and constructs at the frame layer are instantiated based on the knowledge of the project at hand. The main emphasis in this work was, however, on the development of the language, not a method, and therefore this context is only of minor importance as a prior ME context.

The second ME effort aimed to engineer a method[170] for office support systems analysis and design (Conrath *et al.* 1989). This effort was accomplished in the large Esprit project, called the OSSAD project, into which a research group (Vesa Savolainen, Mauri Leppänen) from the University of Jyväskylä was

[170] Actually the artifact was called a methodology in the OSSAD project (Conrath *et al.* 1989).

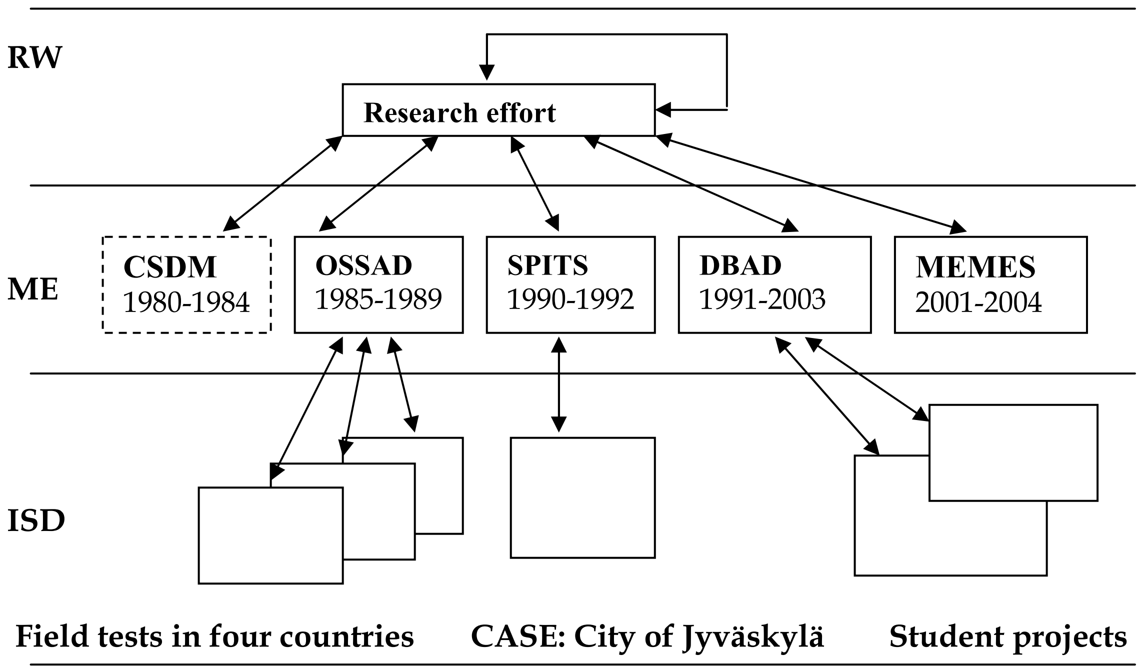


FIGURE 119 Prior contexts related to MEMES

accepted in 1986. Other partners came from France (D. Conrath CETME Aix-en-Provence, P. Dumas CETMA Toulon, G. Charbonnel CETMA Toulon), Italy (V. de Antonellis University of Milan, C. Simone University of Milan, G. de Petra IPACRI Rome, C. de Santis IPACRI, Rome), and Germany (S. Sorg IOT Munchen, E. Beslmuller IOT, Munchen). This author's role in the project was to comment on, ideate, and contribute to the conceptual foundation of the method, to make constructions for some specific parts of the method, as well as to field test the method in a Finnish case organization (Leppänen *et al.* 1989a, Leppänen *et al.* 1989b). The OSSAD method was also field tested in France, German and Italy (Baron *et al.* 1989). Highlights and lessons from the field tests were analyzed and documented (Baron *et al.* 1989). The project engineered the comprehensive method that was published in the manual (Conrath *et al.* 1989), and in numerous articles (e.g. Conrath *et al.* 1988; Charbonnel *et al.* 1991; Conrath *et al.* 1992; Vincent *et al.* 1992; Conrath *et al.* 1999; Savolainen 1999).

The third attempt dates back to 1988/1989 when this author participated in the large consultancy project, which planned the information technology and service strategy for the City of Jyväskylä. For the project, the method called SPITS (Strategic Planning of Information Technology and Services) (Leppänen *et al.* 1991) was engineered. In the method the process of strategic planning is decomposed into three parts: analyzing the service strategy, planning the IS strategy, and planning the implementation of the IS strategy. For each part several design techniques were selected from the existing methods and customized to fit the needs of the project. Researchers acted as teachers and mentors and partly also as planners in the project. Experiences from the project were collected via interviews, although they were not reported in public.

The fourth attempt in ME practice concerned engineering a method for the database application design, here referred to as DBAD, for the purposes of teaching. The first version of the method was constructed during 1985-1993 (Leppänen 1993). This method was based on a view of the centralized data base architecture in the environment of the Ingres database management system. The second version was engineered for web-based database application design with Oracle 8i (Leppänen 2001). Both methods were constructed through selecting and customizing models and techniques from existing data base design methods. The methods were deployed by students in their designing and implementing small-scale database applications in the projects of 2 – 4 members. The course has been lectured seven times so far, and about 100 student projects have been accomplished following the method.

Summing up experiences from the prior ME contexts we can say that in too many cases method engineering was started with obscure ideas of a new or improved method and accomplished in a way that is best described by the phrase ‘engineering-by-trials’. This does not mean that ME efforts would have failed. What it does mean is that the methods were not exactly what was pursued and, in particular, the processes by which they were engineered were neither efficient nor effective. The more people an ME effort involves, the more structured and pre-planned the process should be. A way of working in engineering and a form of presenting outcomes vary depending on numerous situational factors related to e.g. the application domain, availability of existing methods, resources (time, personnel, tools) of engineering work, and skills of method users (cf. developers, consultants, end users, students).

Regardless of differences between the prior ME contexts, we had a strong feeling that there are clearly approaches, principles, and ways of working that are common to all kinds of ME efforts. With those common “ingredients” it should be possible to recognize fundamentals of ME contexts, as well as to organize and to accomplish ME work in a “rational” and efficient manner. Some of these ME “ingredients” were already recognized and outlined during the ME efforts but it was not until in 2001 when the systematic work to engineer MEMES was actually started. In this work we utilized the conceptual foundation, which we had earlier constructed (cf. ontology engineering).

Based on a comprehensive analysis of ME literature (see Chapter 12) we found that there is no complete method for ME, only some ME strategies, ME approaches, meta models, ME procedures, and ME techniques. Consequently, we set up the objective to engineer more comprehensive methodical support for ME. We named it the methodical skeleton to acknowledge that the artifact is not aimed to be a complete ME method.

Engineering the methodical skeleton for ME is actually an instance of ME effort as well. This engineering process should thus apply the same fundamental structural and dynamic “ingredients” as those in the aforementioned prior ME contexts and those that are to be included in the skeleton itself. This observation made us to decide to systematically apply

MEMES during the incremental and cyclic process. Instant feedbacks substantially contributed to the theoretical part of method engineering.

It is not possible here to extensively discuss the experiences, ideas, and contributions related to all the prior ME contexts. What we will do in Chapter 12 is to give some examples of considerations in and outcomes from two ME efforts, the OSSAD project and the MEMES effort, to illustrate the deployment of MEMES. We will also present reflections, in the sense of retrospective analysis, on those prior ME contexts to learn from these experiences.

11.4 Application Area

In this section we describe the intended application area of MEMES deploying the application view defined in Sections 9.5 and 10.5. The application view addresses where and how the method can be applied. The view is expressed in terms of target contexts on two layers. In this case the target contexts are on the ME layer and the RW layer. Target ME contexts mean contexts in which MEMES is to be deployed. Target RW contexts mean contexts in which MEMES is to be elaborated, customized and/or configured, in order to make it suitable for the use in the target ME context(s). According to the ME method ontology, the ME method should also provide backing arguments for the applicability of the method in the contexts and justify them with appropriate evidence. MEMES is a method skeleton and it does not include descriptions/ prescriptions to identify the target RW contexts, nor to carry out actions there. Neither does MEMES provide any empirical evidence on the applicability in ME contexts. Instead, we justify its applicability by using it with analytical and constructive intentions. Next, we describe the application area of MEMES in terms of target ME contexts.

The ME contexts are much more varied than the ISD contexts. Therefore, it is not possible – or not even reasonable – to specify the application area for an ME method with such a specificity as for an ISD method. We specify the application area of MEMES in terms of ME context types and ME strategies distinguished in Section 10.2.

There are three main ME context types: method development, method customization, and method configuration. Method development aims to engineer a generic method or a domain-specific method. Method customization and method configuration strive for engineering an organization-specific method and a project-specific method, respectively. MEMES supports engineering of ISD methods, viewing the ME from the ME systelogical, ME infological and ME conceptual perspectives. This means that MEMES mainly provides methodical support to method development. We have, however, engineered the ME ontology that also includes the ME datalogical perspective. With the concepts and constructs of the ME ontology and the overall structure of MEMES, it is rather easy to elaborate MEMES toward an artifact that supports method customization and method configuration as well.

We have distinguished between three ME strategies: creation, integration, and adaptation (see Section 10.2). Creation means the "greenfield" or "from scratch" strategy to engineer an ISD method in the situation where no current ISD method exists to be used as the basis for ME. Integration means an ME strategy according to which an ISD method is engineered by assembling parts of current methods. Adaptation means an ME strategy according to which an ISD method is engineered by changing, one way or another, the current ISD method. As mentioned in Section 10.2, none of the aforementioned ME strategies is applied as such in practice. In contrast, strategies appear to be mixed. Figure 120 gives examples of mixed ME strategies. In the first case (1), it is seen necessary to make adaptations into the method components before integrating them into the body of a new ISD method. In the second case (2), a new ISD method is primarily constructed as a "green field" product, yet utilizing some existing components. In the third case (3), an ME effort starts as an adaptation process but due to the lack of some functionalities in the current methods, the ISD method is enhanced with the creation strategy. In the fourth case (4), an ISD method is engineered applying all three strategies.

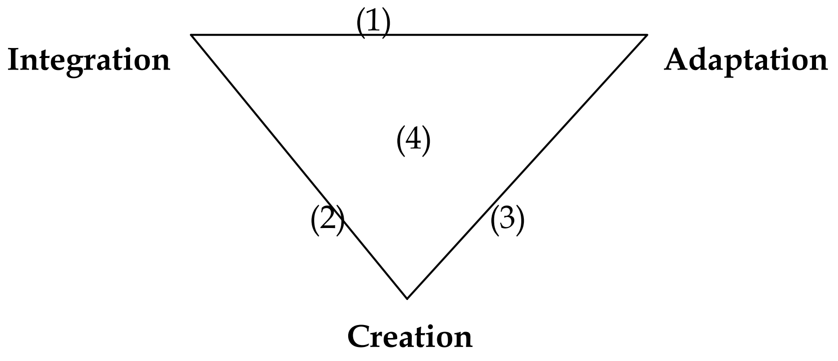


FIGURE 120 Mixed ME strategies

MEMES has not been engineered to support any specific ME strategy. As a method skeleton it aims to provide generic support for all the ME strategies. To offer the support as structured and utilizable as possible, MEMES has been built up deploying the most fundamental structures of contextual ME domains. With these structures it is easy to elaborate MEMES to suit more specifically some particular strategy needed in the certain ME project.

To summarize, we specify the application area of MEMES as follows: *MEMES is aimed to offer methodical support for ME contexts, the purpose of which is to engineer a generic ISD method or a domain-specific ISD method with any ME strategy. MEMES has to be elaborated, customized and configured to make it fit the needs of a particular ME organization or project. This process is not supported by MEMES.*

Concluding from the prior ME contexts involved by this research work we can make the following reflections. The OSSAD project clearly aimed at the development of a domain-specific method with special features of office support systems. It followed the ME strategy close to the type (2), as its purpose was to create something specific for office support systems, yet utilizing components of current methods for those parts of the method which could be found from the existing methods. Some adaptations of those parts were naturally needed during the ME process. The SPITS project actually aimed to engineer ´the project-specific method, but because there was neither the appropriate organization-specific method, nor a generic method, the project applied a mixed strategy of the type (1) in order to first develop a generic method, mainly by integration, and then to configure it into a project-specific method. Some adaptations had to be made to get the pieces fit together. The DBAD method was engineered for teaching purposes. The purpose was to provide students with a coherent set of DB design techniques for the whole range of design life cycle, from requirements engineering to implementation. The main body of the engineered method was generic, but it was instantiated to be applied, in a straightforward way, by groups of students in the course context. Hence, the final outcome can be regarded as a "project-specific" method. The main body of the method was engineered by integrating components from the existing methods with minor adaptations (cf. the type (1)).

We were not able to find any existing ME method that could be accepted as the basis for engineering MEMES. Instead, we found a large number of ME strategies, ME approaches, ME techniques, and ME procedures which could be utilized as components of MEMES, either as such or somewhat adapted. In many cases we had also to develop new constructs. For instance, the overall structure of MEMES was established without any support from existing artifacts. Consequently, this RW effort followed the mixed strategy of type (4).

11.5 Goals of MEMES

In this section we define the goals of MEMES, structured according to the ME method ontology (see Section 10.5). Hence, we define the goals from the contents view, the structural view, and the application view. In addition, we shortly characterize MEMES from the presentation view and the physical view.

The goals of MEMES are:

1. *MEMES should be based on a solid and sound view of the relevant sub-domains*.
   MEMES should be built upon a conceptual foundation composed of information processing contexts on four layers and on three model levels. To satisfy this goal we anchor MEMES on OntoFrame.
2. *MEMES should be modular and flexible*.
   MEMES should be composed of structural and functional components that facilitate the elaboration, customization and configuration of the skeleton

toward a specific ME method. Despite the modular structure, MEMES should still maintain its uniformity, consistence and coherence.

3. *MEMES should be applicable*.
   MEMES should be applicable for framing, constructive and analytical intentions. The framing intension means that MEMES should provide concepts and constructs to help make sense of and structure phenomena in the actual ME. The constructive intention means that MEMES should support the engineering of an ISD method, or parts thereof. The analytical intention means that MEMES should provide main concepts and constructs for the analysis and comparison of existing ME artifacts. ME artifacts here mean ME strategies, ME approaches, ISD meta models, ME techniques, and ME procedures.

MEMES is presented in natural language, supported with diagrams illustrating structural and functional features of the skeleton (cf. the presentation view). It appears in paper form (cf. the physical view). If elaborated and formalized, MEMES can be embedded in a CAME environment.

11.6 ME Workflows

The purpose of this section is to present the overall structure of ME work in terms of ME workflows. The ME workflows are described from the ME infological perspective, meaning that ME purposes, ME actions and ME deliverables are recognized. At the end we show which of the ME workflows are included in MEMES.

We distinguish between five ME workflows (cf. Section 10.3.3): ISD method (ISDM) requirements engineering, ISDM analysis, ISDM design, ISDM implementation, and ISDM evaluation (Figure 121). They can be classified into two categories. The first category comprises the ME workflows that contribute directly to the construction of a new method or an improved method. These workflows are ISDM analysis, ISDM design and ISDM implementation. These differ from one another in their independence of realization issues (cf. Section 6.1). The second category comprises the ME workflows that specify requirements and goals for the ISD method or alternatively evaluate whether the ISD method satisfies the specified requirements and goals. These workflows are ISDM requirements engineering and ISDM evaluation. They cover the whole range of aspects, extending from realization independent to realization dependent ones. In the following, we first give short characterizations of the ME workflows and then consider how they manifest themselves in different kinds of ME contexts.

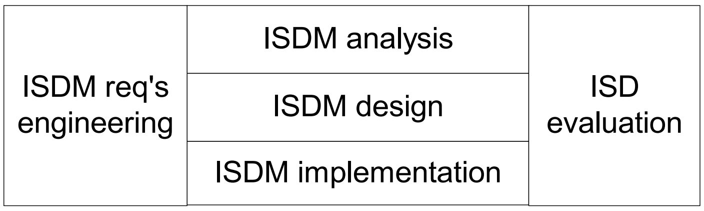


FIGURE 121 ME workflows

ISDM requirements engineering (ISDM RE) means an ME workflow which aims to identify and elicit ME stakeholders' requirements concerning the nature, contents and structure of the ISD method. Requirements can be classified according to the ISD perspectives into ISD systelogical requirements, ISD infological requirements, ISD conceptual requirements, ISD datalogical requirements, and ISD physical requirements. ISDM RE workflow extends from the very beginning to the last steps of the ME effort. In this study we consider only the first tasks of this ME workflow in more detail.

ISDM analysis denotes an ME workflow which aims to produce high-level descriptions of the ISD method. Descriptions are produced from the ISD infological perspective and the ISD conceptual perspective, revealing what ISD actions are performed, for what purposes, and what ISD deliverables are produced. In addition, intra-domain and inter-domain relationships in and between the three ISD domains are, on a general level, specified.

ISDM design refers to an ME workflow which aims to produce more elaborated descriptions of the ISD method. Here the method is considered from the ISD datalogical perspective, uncovering "How" the ISD is to be accomplished. This means that the following kinds of questions are answered: What kinds of ISD roles and ISD positions are established? How ISD actions are decomposed at a detailed level? Which part of ISD work is to be supported by computer-aided tools?

ISDM implementation means an ME workflow which aims to produce concrete descriptions/prescriptions of the ISD method from the ISD physical perspective. That means that the descriptions/prescriptions made earlier are realized and instantiated into a project plan that dictates who does what, why, how, for what, when and where.

ISDM evaluation denotes an ME workflow which aims to produce assessments of one or more ISD methods according to the defined criteria. The criteria are derived from the ISDM requirements specified in the ISDM RE workflow. The ISD method to be evaluated can be at any point of its life cycle. It can be just a roughly outlined artifact, like the one resulting from the ISDM analysis workflow, or it can be a complete ISD method already used in the ISD. The applied criteria may vary from logical to technical and from general to detailed, depending on the applied perspective(s) and the nature of the method. Evaluation is tightly associated to the other ME workflows. For instance, the evaluation of existing methods is done before deciding on ME goals in the ISDM RE workflow.

Figure 122 illustrates the ME workflows (rectangles) and their main deliverables (ellipsis) on a general level. Arrows between the ME workflows stand for the sequence relationships. The ME process is highly iterative in the sense of re-doing, refinement and repeating (cf. Section 4.4.3). Re-doing means that with improved knowledge some tentative aspects of the ISD method are later changed. Refinement means that generic aspects of the ISD method are elaborated from those sketched during the preceding iteration cycle. Repeating implies that ME actions are accomplished iteratively, each time for different ME deliverables, for instance, due to a lack of ME resources. In the figure the area embraced with the bold line shows the scope of MEMES. Thus, the ISDM design and ISDM implementation workflows are excluded from MEMES.

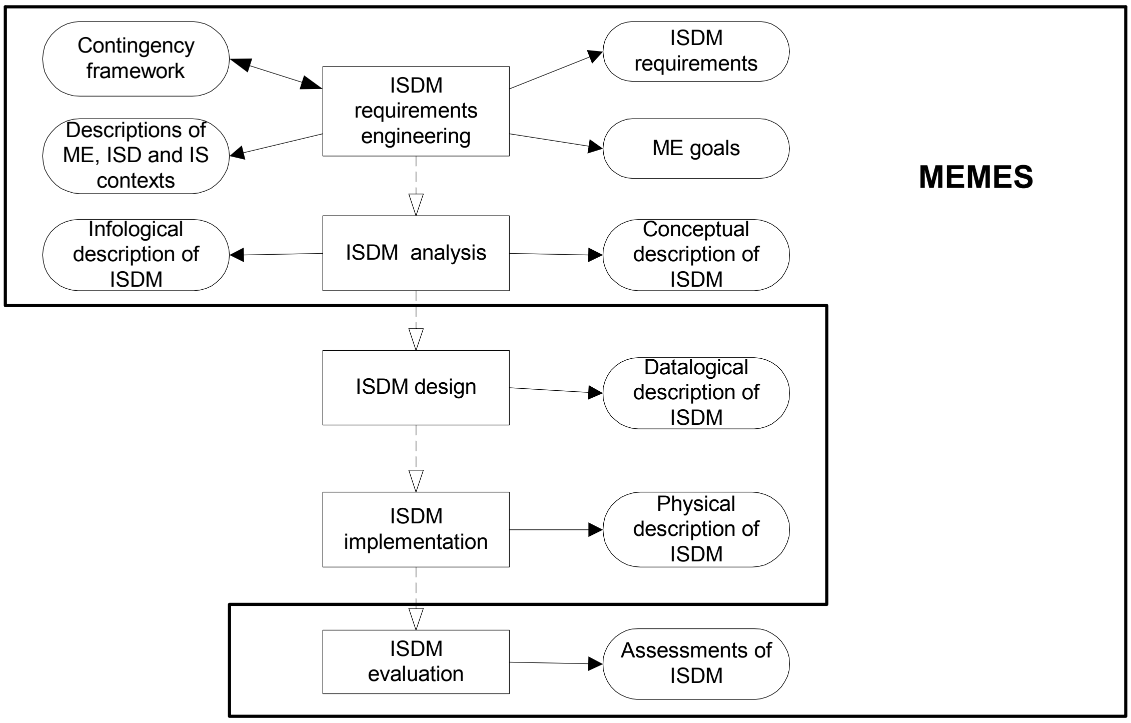

Contingency framework
ISDM requirements engineering
ISDM requirements
Descriptions of ME, ISD and IS contexts
ME goals
MEMES
Infological description of ISDM
ISDM analysis
Conceptual description of ISDM
ISDM design
Datalogical description of ISDM
ISDM implementation
Physical description of ISDM
ISDM evaluation
Assessments of ISDM

FIGURE 122 ME workflows and their main deliverables

The ways in which the ME workflows are executed differ from one another depending on the ME contexts. Figure 123 describes the scopes and emphases of three ME workflows in three types of ME contexts. The development of a generic ISD method or a domain-specific ISD method mainly includes actions of ISDM analysis workflow, which consider the ISD method from the ISD systelogical perspective, the ISD infological perspective and the ISD conceptual perspective. The development context may also contain some ISDM design actions if the method is to support, for instance, the establishment of ISD roles and ISD positions. In the customization context the actions of the ISDM design workflow dominate. In addition, some actions of the ISD analysis workflow may be carried out. For instance, to decide on dropping some ISD actions from a generic ISD method under customization, it may be necessary first to check,

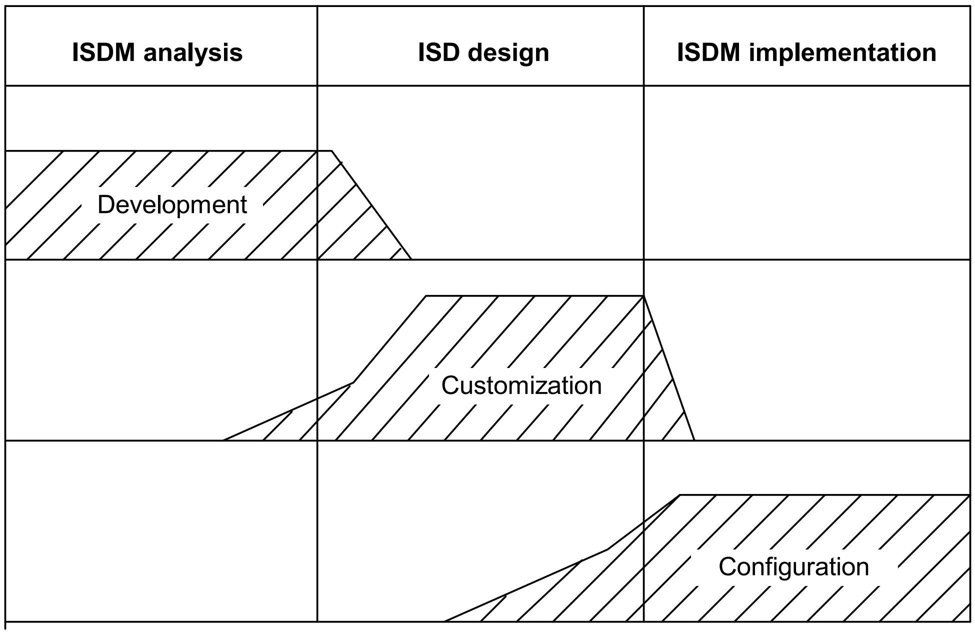


FIGURE 123 ME workflows in the development, customization and configuration contexts

what consequences this deletion may have to the other ISD actions and to carry out necessary re-customization actions of the ISDM design workflow. Likewise, in engineering an organization-specific method the need may arise to describe an organizational structure including individual units and persons in charge of specific ISD actions. Consequently, the customization context also includes some actions of the ISDM implementation workflow. The configuration context primarily contains actions of the ISDM implementation workflow through which an ISD project organization with specific responsibilities and authorities is established, the schedule with baselines is decided, and the like. In this context some needs may also emerge to reconsider ISD datalogical aspects of the organization-specific method.

As shown in Figure 122, MEMES contains three ME workflows. In the next sections we will describe them in more detail, including the underlying approaches and steps.

11.7 ISDM Requirements Engineering

ISDM requirements engineering (ISDM RE) means an ME workflow which aims to identify and elicit ME stakeholders' requirements on the nature, contents and structure of the ISD method. ISDM requirements can be brought out nearly at any stage of an ME effort. In the first stages ISDM requirements concern features related to the use and contents of the ISD method, later ISDM requirements pertain to e.g. the presentation and technical support of the ISD method. Because MEMES does not cover the whole range of actions of the workflow, we concentrate on describing ME actions at the first stages of the ISDM requirements engineering.

Figure 124 describes the ME tasks of the ISDM RE workflow concerned here. The arrows between the ME tasks present the sequence relationships. The ME tasks are: (a) decide on the feasibility of a contingency framework, (b) analyze the ME context at hand, (c) characterize the target contexts, (d) analyze prior contexts, (e) specify ISDM requirements, and (f) determine ME goals. The purpose of the first task is to consider whether it is possible and feasible to use some contingency framework to elicit and structure descriptions of ISD methods and concerned contexts. The second task aims to find out what kind of context the ME context at hand is. In the third task the target ISD contexts and the target IS contexts are characterized[171]. In the fourth task the prior contexts at the ME layer, the ISD layer and the IS layer are analyzed to learn from the gained experience. The purpose of the fifth task is to derive requirements for a new ISD method, or for an improved ISD method. The last task in the ISDM RE workflow is to determine goals for the ME context at hand. The seventh task in Figure 124 is called 'Analyze the current method(s)'. This task is actually a part of the ISDM evaluation workflow but it is included in the figure to show how it relates to the ISDM requirements engineering. It is presented by dotted line in Figure 124.

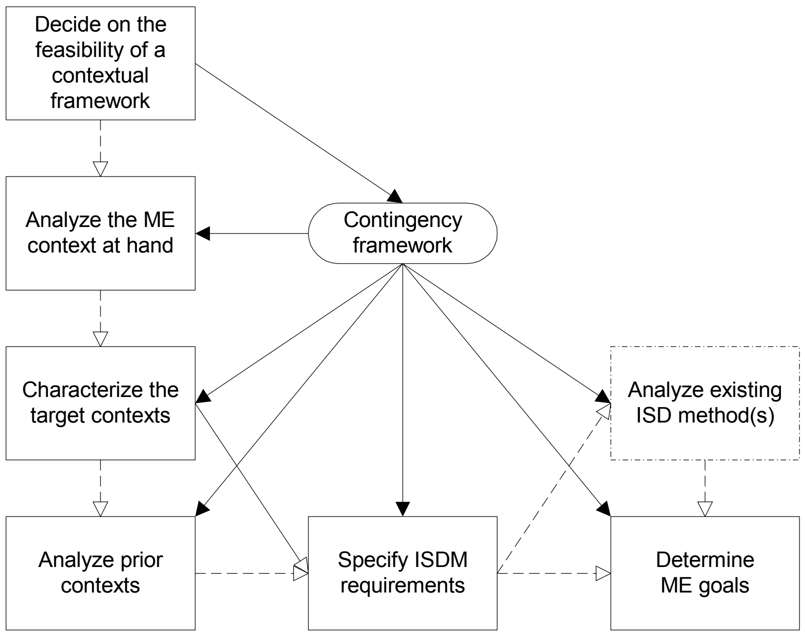

Decide on the feasibility of a contextual framework
Analyze the ME context at hand
Contingency framework
Characterize the target contexts
Analyze existing ISD method(s)
Analyze prior contexts
Specify ISDM requirements
Determine ME goals

FIGURE 124 Tasks of the ISDM RE workflow

The tasks of the ISDM RE workflow are connected to one another by the sequence relationships in the figure. It should, however, be noted that the tasks are highly iterative and also the order in which tasks are carried out can vary a

171 Note that we ignore the characterization of the target ME contexts.

lot depending on the ME context. In the following we describe each task in more detail.

11.7.1 Decide on the Feasibility of a Contingency Framework

The purpose of this ME task is to consider whether it is possible and feasible to use some contingency framework to elicit and structure descriptions of ISD methods and the concerned contexts. The concerned contexts mean here the ME context at hand, the prior contexts and the target contexts. If the answer is 'yes', a contingency framework is selected from those available and adapted, if necessary.

A contingency framework can help to focus on and elicit those features that are the most essential to the concerned things. Contingency frameworks are commonly applied in the selection and construction of methods, techniques, models or tools for the needs of particular ISD contexts. For instance, in selecting the ISD method it is examined how well characterizations of the ISD method and of the target ISD context, structured according to a contingency framework, match with one another. There are a large number of studies on contingency approaches (e.g. Naumann *et al.* 1980; Davis 1982; Burns *et al.* 1985; Iivari 1989b; Saarinen 1990; Louadi *et al.* 1991; Cockburn 2000; Iivari *et al.* 2001; Kettinger *et al.* 1997; Roberts *et al.* 1998; van Swede *et al.* 1993; Odell 1996; van Slooten *et al.* 1994; van Slooten *et al.* 1996; Lin *et al.* 1999; Zhu 2002). Based on the relevant literature, we propose the meta model of a contingency framework in Figure 125 and define the concepts in the framework.

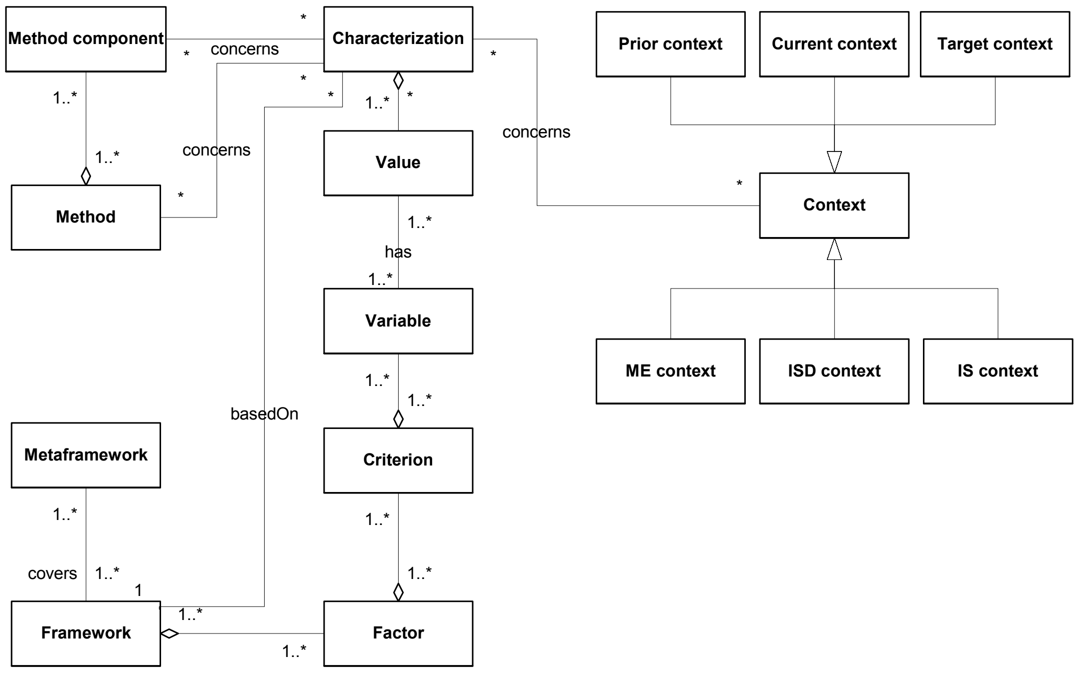


FIGURE 125 Meta model of a contingency framework

A *contingency framework* is composed of one or more contingency factors. A *contingency factor* stands for a certain type of feature in the concerned thing that is considered essential. An example of the contingency framework is the two-dimensional framework by Louadi *et al.* (1991) for the selection of a method for an ISD project. This contingency framework is composed of two factors: uncertainty and complexity. Each contingency factor can be concretized by one or more *criteria.* For example, complexity of the ISD context can be considered in terms of project size, number of users, volume of new information, and complexity of new information production (cf. Davis 1982; Davis *et al.* 1985). Each criterion can be measured through one or more *variables,* whether single-valued or multi-valued. Project size, for instance, can be measured in terms of man-hours, money, months etc.

A contingency framework can be used in making characterizations of ME contexts, ISD contexts, or IS contexts. Characterizations of a (information processing) context, whether past, existing or designed, may also concern its object system (OS) and utilization system (US). A contingency framework can also be applied to characterize methods, or a part thereof, expressing whether a method or its component is feasible or not feasible in certain kinds of contexts. Characterizations are built up from values of variables.

In the ISD field many kinds of contingency frameworks and approaches have been suggested. These contingency frameworks are based on a large variety of categorizations of contingency factors (e.g. Davis 1982; van Swede *et al.* 1993; van Slooten *et al.* 1993; van Slooten *et al.* 1996; Harmsen 1997; Punter *et al.* 1996; Kettinger *et al.* 1997; Roberts *et al.* 1998; Lin *et al.* 1999; Kraiem *et al.* 2000). Zhu (2002) distinguishes between three contingency approaches for the method selection: contingency at the outset, contingency with a fixed pattern, and contingency along development dynamics. The first approach aims to select a single method or a fixed combination of methods for the whole lifecycle of an ISD project. According to the second approach, suitable parts of the method should be selected at each individual stage of a project. The third contingency approach states that various issues that shape ISD should be appreciated and tackled at each unique development moment. This corresponds to We are not able here to consider the contingency approaches, nor the contingency frameworks, in more detail. Instead, we bring out a need for a *meta-contingency framework,* which helps us select a contingency framework that is the most suitable to the situation at hand. One of the variables of such a meta contingency framework is rigor (cf. Zhu 2002, 353).

In the literature there are also techniques for the use of contingency frameworks in method engineering. Punter *et al.* (1996) present a multi-dimensional model, called the spider's web portfolio, to organize basic modeling strategies according to the contingency factors based on the notions of complexity and uncertainty. They also propose a process of determining an appropriate modeling strategy using the spider's web portfolio. Harmsen *et al.* (1995) and Harmsen (1997) suggest the so-called 3 S's model (success, situation, and scenario) and steps to use the model in the selection of method fragments for a certain situation. For a situation a large set of situation factors are defined with associations to performance indicators. For example, low "management commitment" contributes negatively to performance indicators such as "organizational management" and "system acceptance" (Harmsen 1997, 204). For each method fragment, a large set of scenario aspects is made available to characterize method fragments. To each scenario aspect a set of performance indicators is associated. Hence, having knowledge of situational factors of a given project it is, at least in principle, possible to select a method fragment that compensates the expected negative effects of the situational factors with positive effects of the scenario aspects of the selected method fragments.

Many kinds of critics against the contingency approach have been brought out (e.g. Kumar *et al.* 1992; Avison 1996; Tolvanen 1998; Zhu 2002). For instance, the use of the contingency approach may be costly, and its effective use may appear to be difficult. Nevertheless, we suggest that in the ME context it is reasonable to consider the feasibility of a contingency framework. This can be carried out with the following steps:

- *Consider on which level of detail it is necessary to characterize the ME context at hand.*
- *Consider on which level of detail it is necessary to characterize prior ME, ISD and IS contexts.*
- *Consider on which level of detail it is possible to characterize target ME, ISD and IS contexts.*
- *Consider on which level of detail it is necessary to characterize ISD methods, or parts thereof.*
- *Consider by which features it is feasible to express characterizations of each of the aforementioned things.*
- *Consider whether there are contingency frameworks that are suitable to the needs above. If not, make the necessary modifications and enhancements in the framework that comes closest.*
- *Make a detailed description of the contingency framework selected/produced.*

There are three remarks worth making. First, the steps above have been organized with the principle according to which some conceptions about the relevant ME / ISD contexts are first obtained and after that the search for a suitable contingency framework begins. It depends on the situation at hand how detailed information about the contexts is gathered in this "pre-phase". Second, it goes beyond our scope to give guidelines of how to engineer a contextual framework that suit the situation at hand. Third, after having engineered the framework it can be applied in MEMES according to any of three approaches distinguished by Zhu (2002). In this process some of the techniques suggested in the ME literature (e.g. Harmsen *et al.* 1995; Punter *et al.* 1996; Harmsen 1997) can also be deployed.

11.7.2 Analyze the ME Context at hand

The purpose of this ME task is to analyze the ME context at hand in order to decide which of the following ME tasks should be accomplished, in which way and in which order. The ME context can be characterized in many ways. On a general level, the ME context can be merely classified to be a development context, a customization context, or a configuration context. Also ME strategies can be used to generally describe the nature of the ME context. On a more detailed level one pays attention, for instance, to the following issues: What is the level of knowledge of, experience from, and skills in method engineering in the organization? Is there any support available for ME in the environment? What kinds of technology are available for the ME effort? How clear and steady are the conceptions about what it is really that is wanted from the ME context? How is the ME context connected to other efforts? Is there any customary way of working in these kinds of situations in the organization? What is the level of management commitment to the ME effort?

The ME context can be described with numerous aspects based, for instance, on seven contextual domains. Let the following be a simple example of considerations needed when more multifaceted factors are used in ME. Here we use one factor that is especially significant to the way in which ME work is carried out in the ISDM RE workflow. This factor is the *initiating condition,* the main reason for why the ME effort is initially launched. We distinguish between two types of reasons, problem-driven and policy-driven, and specify two ISDM RE approaches based on them. In the specification of the approaches we utilize the work by Sutcliffe (1996) who proposes four tentative models of the IS requirements engineering process for possible pathways. The models are called policy-driven requirements, problem-initiated requirements, requirements by example, and requirements imposed by the external environment. Our ISDM RE approaches correspond to the first two in the above list. Next, we define the approaches and clarify how the ME process proceeds in each of the cases.

The *problem-driven approach* is applied when problems in prior ISD contexts are experienced to be severe, thus calling for changes in the current ISD method. Problems manifest themselves as inefficiency in ISD processes, difficulties in project planning and control (e.g. uncontrolled process iteration), problems in cooperation and communication (cf. Tollow 1996), or inflexibility in the customization of the ISD method. Problems may also come out as errors in designs and implementations, difficulties in the maintenance of IS's, and the dissatisfaction in, or even the resistance to, the use of the method. In all these cases it is necessary to first collect and carefully analyze experience from the prior ISD contexts, and then decide how to proceed.

The analysis may lead to the conclusion that the main reasons for the problems are not deficiencies in the method, but shortages, for instance, in technical skills of designers, in cooperation between the ISD stakeholders, or in incompetence in the use of software tools. In these cases, ME would not bring any solution. In contrast, if problems are traced to deficiencies in the ISD method, the decision should be made on whether the problems are so severe and the desire to solve them so unanimous that the ME work should continue. The decision is also needed on whether the existing method, in spite of its deficiencies, can serve as a basis for engineering an improved method. If the answers are yes, the scope of engineering requirements on the ISD method is determined and the ME process goes on.

The use of the problem-driven approach to ME is widely applied in practice (e.g. Jaaksi 1997; Tollow 1996). It is more focused and concrete than the other ME approaches. On the other hand, starting from the problems may limit the considerations too much to contemporary issues and ignore potentials of larger improvements (Tolvanen 1998).

The *policy-driven approach* means an effort that is triggered by the need to deploy a novel technology, to build for a new application area, and/or to apply a new approach to ISD. For instance, to develop a ubiquitous information system based on the microchip technology may require such a novel approach that it is seen reasonable to develop a new method, or at least a set of new techniques. Examples of other new approaches are client-led approach, agent-based approach, agile approach, generative programming approach, aspect-oriented approach, and soft computing approach. In this kind of situation, first steps in the ISDM RE workflow are, instead of considering problems in the prior ISD contexts, to collect and analyze the expectations of stakeholders and to derive requirements from them. Also new technological and organizational solutions should be considered. High-level statements are decomposed and refined to obtain more concrete visions of the ISD and requirements for the ISD method. In this work, contingency frameworks (e.g. van Slooten *et al.* 1996; Harmsen *et al.* 1994; Punter *et al.* 1996) may be found feasible.

The categorization of ME approaches into the problem-driven approach and the policy driven approach is not complete. There are many other reasons, although not so common, for initiating an ME effort. One of these reasons is the need to make the ISD process more disciplined. The motivation for this may be a desire to acquire the ISO certification for the organization (van der Pijl 1997). That requires not only documenting the followed conventions in a rigorous manner, but also careful reconsiderations of routines.

Contextual features of the ME have substantial influence upon which tasks, and in which order, are carried out in the ISDM RE workflow. Therefore, in each ME effort it is important to analyze which kind the ME context at hand is and to plan the next courses of action accordingly. Below we present steps with which the analysis of the ME context can be done:

- *Collect the data already available about the ME context at hand.*
  The purpose is to find out, among else, who suggested the initiation of the ME effort and what reasons were used to justify it. In addition, memos of negotiations and even e-mails concerning the ME effort can be valuable to make the picture of the situation more clear.

- *Characterize and analyze the ME context to find out the essentials of the context.* Make a structured description of the ME context at hand which highlights the most essential features of the context. Some contingency framework can be utilized to elicit, structure and inter-relate the features.
- *Decide on the next actions.* Based on the above analysis, decide which ME tasks, with which emphasis and in which order should be carried out in the following.

11.7.3 Characterize the Target Contexts

The purpose of this ME task is to characterize contexts where the concerned method is to be engineered and deployed. These contexts comprise target ME contexts, target ISD contexts and target IS contexts. The target ME contexts should be characterized because it is necessary to know the degree to which it should be possible or desirable to make refinements, customizations, and/or configurations in the ISD method before and/or during the method deployment. This is affected by the kind of the ISD method under engineering (cf. generic, domain-specific, organization-specific vs. project-specific method). The target ISD context means a context for which the method is to be engineered. The more specific the context is, the more detailed its characterization should be. Closely related to the target ISD context is the application and the IS contexts in which the application is to be run. IS contexts can be viewed from various perspectives. The IS systelogical perspective recognizes the business system as a utilization system. For instance, the ISD method could be intended for the use in ISD contexts which develop transaction processing systems, office information systems or inter-organizational systems.

The level of detail applied in the characterizations of the target contexts depends on the type of the ME context. In the method development one is interested in ISD approaches, application areas, ISD process models, IS architectures, etc. In the ME context aiming to engineer an organization-specific method, characterizations refer to the organizational culture, politics and conventions, as well as to preferred ISD approaches and principles expressed in more detail. In the ME context aiming to engineer a project-specific method the target ISD context and the target IS contexts are characterized on the most detailed level. If the method is not intended to be applied as such, it should be mentioned in which kinds of situations the method is assumed to be elaborated, customized and/or configured.

Inputs to the task are more or less vague conceptions of the ISD method and of the target contexts drafted in the preceding task. The purpose here is to elaborate these conceptions. Contingency frameworks can play a vital role in the task. In Table 33 we present examples of contingency factors related to the IS context and the application, on one hand, and to the ISD context, on the other hand (van Slooten *et al.* (1996, 32-33); Harmsen (1997, 206-208); Kruchten (2000, 50-51); Firesmith 2004).

TABLE 33 Examples of contingency factors

IS context & Application	ISD context
Business criticality	Management commitment
Organizational impact	Stakeholder involvement
Formality of information and business processes	Size of the project
Size and complexity	Knowledge, experience and skills
Stability of information and business processes	Corporate culture
User knowledge and experience	Degree of resistance and conflict potential
	Time pressure
	Availability of human resources
	Availability of facilities
	Process breadth
	Clarity and stability of requirements and goals
	Novelty of technology and methods (e.g. tool compatibility)
	Contractual issues (e.g. subcontracting)
	Legal and regulatory issues (e.g. standards and certificates)
	Degree of geographical distribution of the project
	Dependency of other projects

Characterizing the target contexts can be seldom carried out in a straightforward manner. Below we present a preliminary set of steps to structure the process:

- *Characterize the most essential features of the target ISD context and the IS contexts.*
  Figure out what is the most essential to the method under construction: e.g. What workflows should it cover? What are the most typical features of applications and target IS contexts? Are there any preferences or requirements for specific ISD paradigms, ISD approaches, and ISD principles?
- *Characterize the target ME context.*
  Draft circumstances in which it should be possible or necessary to elaborate, customize and/or configure the method to make it suitable for the needs of the target ISD contexts.
- *Complete characterizations of the target ISD contexts and the target IS contexts.*
  Elaborate the characterizations of the ISD target contexts and the IS contexts in more detail, possibly with contingency frameworks, in order to cover all important aspects of the situations in which the method is to be deployed.

11.7.4 Analyze Prior Contexts

The purpose of this ME task is to describe and analyze the contexts in which the concerned ISD method(s) have been engineered and/or used, in order to learn from experience. The concerned method means the method that is considered suitable to provide a basis for integration and/or adaptation in the ME context at hand. The analysis involves contexts on the ME layer, ISD layer, and IS layer. Knowledge about contexts on the ME layer is relevant to forming a solid view of the background and nature of the method (cf. the context of creation in Jayaratna 1994, 228). The ISD method represents organizational knowledge about ISD, externalized as an artifact. To ensure that the ISD method represents the "best" practices in the ISD, it is necessary to assess how the ISD efforts accomplished with the support of the method have succeeded. This necessitates the analysis of the prior ISD contexts. Because success or failure ultimately comes to light in IS practice, the IS contexts concerned should also be involved in the analysis. It should, however, be noticed that it is not always possible to show cause – effect relationships between IS problems and courses of action in ISD work.

A need for, and a possibility to, the analysis of the prior contexts depends on the type of the ME context at hand. If the ME context aims to develop a generic method primarily "from scratch", there are usually no prior contexts to be analyzed. If a generic method is to be engineered by adaptation or integration from the existing methods, prior ME contexts exist. Also if a generic method is to be engineered by decustomization and/or deconfiguration, there may be prior ISD contexts that should be analyzed. The prior IS contexts are left unconsidered in all the aforementioned cases.

The more concrete the ISD method to be engineered is, the more necessary it is to consider the prior contexts. Customizing an organization-specific method is grounded on the needs and features of the specific organization. In this case, the contexts relevant to the analysis embrace ME contexts in the near past, as well as related ISD efforts. Also those IS's that are "run" according to the prescriptions and with the support of software developed in the concerned ISD contexts can be included in the analysis. The analysis is especially important if the problem-driven ME approach is applied.

Instantiating a project-specific method is highly dependent on the features of that particular project. For this reason, the emphasis in the analysis moves from the ME contexts to contexts on the lower layers. If the instantiation is based on a generic method, the analysis resembles that which is applied in the customization. Although ME is mainly focused on the needs of a particular project, it is reasonable also to take into account previous conventions and future visions of the organization. If the instantiation is based on an organization-specific method, the purpose of the analysis is to find out what kinds of project-specific methods have been used in the organization, and with which kinds of steps and results. It is also necessary to consider how successful the developed IS's have been and why.

The analysis of the prior contexts proceeds with the following steps:

- *Consider needs to analyze prior contexts.*
- *Identify prior ME contexts, prior ISD contexts and prior IS contexts to be analyzed.*
- *Define the level of detail for the analysis of each context.*
- *Collect the data that has already been produced about the prior contexts.*
  The organization may have a policy to document problems encountered and innovations made in the ISD projects. Also some IS's have computer-based systems (e.g. Help Desk systems) that record and disseminate information about problems encountered in IS practice, as well as about change requests. This data is collected for further consideration. If no data is available or if it is inadequate, more data is acquired by studying documentation about ISD and ME efforts, interviewing stakeholders involved in the contexts, etc.
- *Make a general description of each relevant prior context.*
  Descriptions should be presented as scenarios (Harmsen 1997, 202, Vlasblom *et al.* 1995, 602) that characterize the contexts in a concise and condensed manner. One way of structuring the descriptions is to apply contingency frameworks.
- *Carry out the analysis.*
  The aim of the analysis is to find out failures and successes, as well as reasons behind them in the prior contexts.

11.7.5 Specify ISDM Requirements

The purpose of this ME task is to derive requirements for a new ISD method, or an improved ISD method, from the characterizations of the target contexts and from the analysis of the prior contexts. If the ME context follows the problem-driven approach, the problems collected and analyzed from the prior contexts provide a baseline for requirements specification. If the ME context applies the policy-driven approach, the focus is on deriving requirements from the characterizations of the target contexts.

Deriving requirements is not an easy task for many reasons. First, as mentioned above, problems encountered in prior ISD contexts may not be due to deficiencies in the applied method but result from some human, organizational or technological reasons. Second, although the problems would be consequences of the method use, it is difficult to conclude how differently one should have behaved to be more successful, and how this behavior should be expressed in the method. Third, it is important to figure out how requirements should be expressed so that it is possible, based on them, to start engineering a new, or improved method. Also when following the policy-driven approach it is difficult to name features that are essential to the target context.

Regardless of how ISDM requirements are derived they have to be associated to those stakeholder(s) who have presented and/or agreed on the requirements (cf. the expressedBy relationship in the ME ontology, Section 10.3.5). Requirements should also be classified according to how necessary they are deemed (e.g. obligatory, favourable, optional). In this task the use of a proper contingency framework may help.

In conclusion, the steps to specify requirements for an ISD method are:

- *Decide on what issues are taken into account in specifying requirements for the ISD method (cf. problem-driven approach vs. policy-driven approach).*
- *Specify requirements for the ISD method and present them in a structured form.*
- *Identify stakeholders and attach the requirements to them with priorities.*

11.7.6 Analyze Existing Method(s)

The purpose of this ME task is to find out whether there already exists one or more ISD methods that satisfy, at least to some extent, the specified requirements. This task actually belongs to the ISDM evaluation workflow, but we discuss it here to provide a proper context for the understanding of its intention and content when used in this stage. We have applied the generic principles and steps, presented for the task in Section 11.9, to tailor the description of the task below.

A way of carrying out this task depends on the type of the method the ME context is aiming at. In engineering a generic method, or a domain-specific method, the analysis is mainly targeted at those generic/domain-specific methods that seem to contain desirable features. In some cases, it may also be worth of looking for organization-specific methods that could be suitable for decustomization. In the customization context the analysis is targeted at the current method of the organization, and if not available or if the method is too far from the specified requirements, it is focused on more promising generic/domain-specific methods. Also successfully deployed project-specific methods in the organization could be considered. In the configuration context other project-specific methods as well as the organization-specific method(s), if available, are analyzed. Sometimes the analysis is extended, also in this case, to address generic methods in order to obtain fresh ideas of ways of modeling, working and controlling in the ISD.

The task of analyzing ISD methods proceeds with the following steps:

- *Specify the criteria for the analysis.*
  Specification of the criteria is based on the ISDM requirements specified in the preceding task. It can also utilize evaluation criteria presented in the ISD literature (e.g. Olle *et al.* 1983; Olle *et al.* 1986; Law *et al.* 1984; Law 1988; Karam *et al.* 1993; Flynn *et al.* 1993; Jayaratna 1994; Iivari 1994; Kitchenham 1996a; Tran *et al.* 2003; Dam *et al.* 2004).
- *Select ISD methods for the analysis.*
  Selection is based on the availability of methods, preconceptions about their match with the requirements, and resources available for the analysis.

- *For each selected method, carry out an analysis according to the specified criteria.* If the set of the methods is large, the analysis is divided into two parts. In the first part an investigation is conducted according to the most important features in order to decrease the number of the analyzed methods. In the second part an in-depth analysis is carried out only for the most potential methods.
- *Make a summary of, and conclusions from, the analysis.* The purpose of this step is to find out what method(s) or method component(s), if any, can serve as a basis for the next tasks. One possible conclusion from the analysis could be that there already exists a method that satisfies, to a reasonable extent, the specified ISDM requirements and therefore no need for further ME tasks prevails.

11.7.7 Determine ME Goals

The purpose of this task is to decide on goals for the ME context at hand. ME goals concern the ME context as a whole, as well as each of its constituents. ME goals concerning ME deliverables are primarily related to the goals of the ISD method. ME goals are determined on the basis of the specified ISDM requirements and knowledge obtained from the analysis of existing ISD methods. Also resources available for the ME context are taken into account in determining ME goals. The last issue is highly important in practice. As Jaaksi (1997) points out, ME efforts frequently take place under financial pressure, and in exchange of the resources used in ME, significantly better solutions from ISD are expected. The goal can never be to find the best method, but a satisfactory method instead.

Determining goals is a value-based function: what the ME stakeholders consider ″better″ or ″desirable″, and how much they want to invest on achieving that “better” (Kumar 1984, Kumar *et al.* 1992, 264). For eliciting stakeholder value profiles, a survey instrument called ISD-PVQ (information Systems Development – Personal Value Questionnaire) (Kumar 1984) can be used.

To structure the ME goals, a proper contingency framework can be used. Here, we apply the contextual framework to show how ME goals can be structured according to the target contexts with seven contextual domains and on two layers. First, we can distinguish the goals that concern the target ME context as a whole or some of its contextual parts, e.g. an ME organization, ME actions, ME deliverables, ME resources, ME tools, etc. The main ME deliverable is the ISD method that is seen at the ME layer as a representational and physical artifact. Second, through the contents of the ISD method the ME goals involve the target ISD contexts. Actually it would also be possible, through the contents of ISD deliverables (i.e. IS models), to state goals that are related to the target IS context (e.g. a goal to engineer an ISD method for geographic information systems). We do not go into such detail here. In the following we give examples of pertinent issues for which the ME goals can be stated, on two layers.

Examples of issues in the ME context for which the ME goals can be stated are:

- *Purpose*. What are the general objectives of the ME (e.g. high quality, minimizing resources)? What are the ME strategies and the ME approaches to be applied?
- *Actor*. Who should take part in the ME and in which role?
- *Action* & *Time*. What workflows should the ME cover and in which phase structure? What are the time limits/schedule for the ME?
- *Object*. In which form and physical appearance should the method be presented (e.g. in a manual, in the web, in a CAME tool)?
- *Facility*. What are the resources available for the ME and which ME tools are to be used?
- *Location*. Where should the ME be accomplished?

The ME goals concerning the contents of the ISD method can be categorized according to the ISD perspectives. ISD systelogical goals concern ISD application areas, ISD paradigms, ISD approaches, and ISD principles (cf. the application view and the generic view in the ISD method ontology) (cf. route maps in van Slooten *et al.* (1996)). An example of the ISD systelogical goal is: "to engineer a method for developing data-intensive web-based applications with the object-oriented approach and active customer involvement". Applying the ISD infological, ISD data logical and ISD physical perspectives means that ME goals are expressed in more detail, commonly structured according to the ISD contextual domains. Examples of issues pertaining to the domains of the ISD context for which the ME goals can be stated are:

- *Purpose*. What are the most important issues the ISD method should support in the ISD effort (e.g. high quality of user requirements, easy maintenance of the system, minimum costs of development)?
- *Actor*. Who should participate in the ISD and in which role (e.g. the client-led approach)?
- *Action*. What workflows should the ISD cover and with which phase structure (e.g. analysis and design)?
- *Object*. What are the most typical aspects of the IS's to be developed with the support of the ISD method (e.g. data-intensive, web-based)?
- *Facility*. What kinds of tools are supposed to be used in the ISD (e.g. CASE tools)?
- *Location*. Where is the ISD to be accomplished (e.g. in a software house, in an industrial organization)?

The ME goals should be attached to ME stakeholders who have expressed/ agreed on them (cf. Nuseibeh *et al.* 1996, 171), as well as to reasons due to which the goals are seen to be important. As specified in the ME ontology (see Section 10.3.1), reasons can be brought out in terms of requirements, problems, opportunities/threats, and strengths/weaknesses. The ME goals are inter-related with one another in many ways. Before making final decisions on ME goals one should make a goal analysis rooted on the established goal hierarchies (cf. the refinement relationship) and consider the costs of and benefits from various alternatives. Decisions on ME goals are made, not only at the beginning of the ME, but across all the ME phases. Goals, stated first on a general level, are later refined and made more concrete to cover specific issues.

To summarize the discussion above and to structure a way of working in this task, we present a set of steps for determining ME goals:

- *Select issues on which it is necessary to make goal statements.*
- *Collect requirements specified for the ISD method.*
- *Collect constraints concerning the ME.*
- *Formulate alternative ME goals based on the requirements and constraints.*
- *Evaluate alternatives based on the predefined set of criteria (incl. costs and benefits).*
- *Make decisions on alternative ME goals.*

11.8 ISDM Analysis

ISDM analysis workflow comprises ME actions, which aim to produce high-level descriptions of the ISD method. The ISD method is considered from the ISD infological perspective and the ISD conceptual perspective. Consequently, in this ME workflow the concepts and constructs of the ISD purpose domain, the ISD action domain, and the ISD object domain are used to produce prescriptions of what is to be done, for which, and why in the target ISD context. Also the conceptual contents of the ISD deliverables are specified in this workflow.

The main inputs to the ISDM analysis are: (a) overall descriptions of the ME context and the target contexts, (b) a list of requirements for the ISD method, (c) a summary of the analyses of the current ISD method(s), and (d) a structured and preferably prioritized list of ME goals.

The ISDM analysis is composed of three main tasks: (a) infological ISD modeling, (b) conceptual ISD modeling, and (c) inter-perspective ISD modeling. With the infological ISD modeling, a description of the ISD method is produced to reveal ISD purposes, ISD actions and ISD deliverables. In the conceptual ISD modeling an IS ontology is engineered to be used in describing the conceptual contents of the ISD deliverables. The inter-perspective ISD modeling is needed to integrate and verify the two perspectives of the ISD context.

Before describing the tasks of this ME workflow in more detail, we first define two ME approaches, which affect the order in which the tasks of the ISDM analysis workflow are accomplished.

11.8.1 Approaches

We distinguish between two ME approaches, the functional approach and the conceptual approach, which differ from one another with regard to the order in which ISD perspectives are applied (Figure 126). According to the *functional approach* the ISDM analysis starts with describing the functional structure of the target ISD context, meaning that ISD purposes, possibly drafted in the previous ME workflow, are elaborated and ISD actions and ISD deliverables are sketched out. Within each of the aforementioned ISD domains, decomposition and specialization are applied to establish goal hierarchies, action hierarchies and deliverable hierarchies. The ISD actions and the ISD deliverables are related to one another through the intra-domain and inter-domain relationships. After constituting some view of the functional features of the ISD context, understanding of the contents of the ISD deliverables is captured and deepened through conceptual ISD modeling.

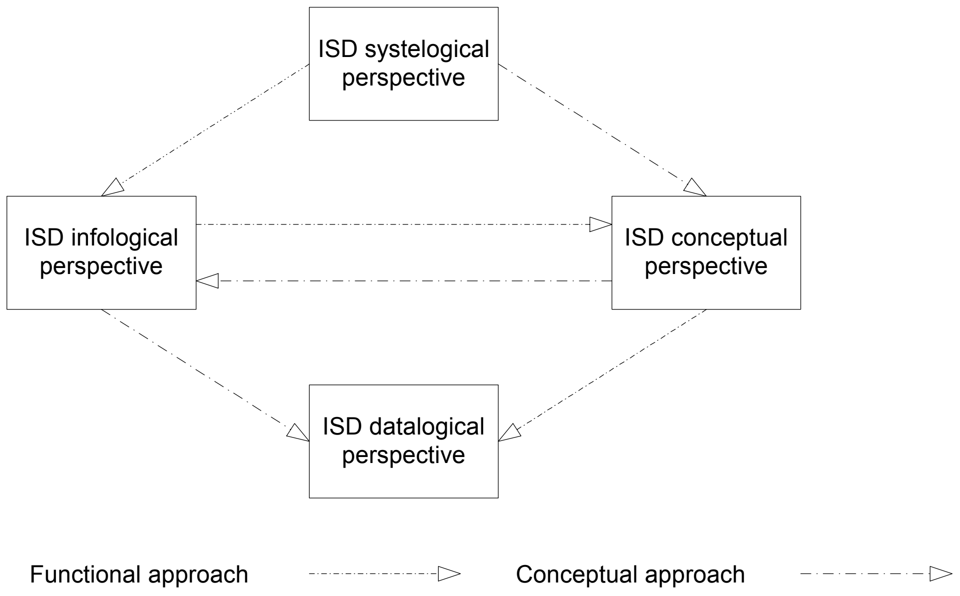

ISD systelogical perspective
ISD infological perspective
ISD conceptual perspective
ISD datalogical perspective
Functional approach
Conceptual approach

FIGURE 126 Two approaches to the ISDM analysis

The purpose of the *conceptual approach* is to first establish an overall view of the object system (OS_{ISD}) which the ISD execution deliverables signify. This means engineering an IS ontology. An IS ontology is composed of concepts and constructs defining the IS from five IS perspectives (see Section 8.4). There are several sub-approaches, which affect the order in which these IS perspectives are applied. They will be discussed in Section 11.9.3. After having defined the essential part of the IS ontology, ME moves to consider ISD deliverables that operate with the concepts of the defined IS ontology, and ISD actions which produce those ISD deliverables. The ISD actions and the ISD deliverables are then further elaborated.

In practice, these two approaches are rarely applied in a pure format. Instead, some kinds of mixed approaches are favored. There are, however, some circumstances, in which an approach closer to the conceptual one suits better than the functional approach, and vice versa. The more novel the application domain is for which the ISD method is to be engineered, the more important the role of the conceptual ISD modeling has in the ME context. This is so because without knowing, at least to some extent, the structure and behavior of an artifact, it is very difficult to specify how to design it. Examples of these kinds of novel application domains are e.g. architecture design, web-application design, geographical information system design, and ubiquitous system design. In contrast, if the ME context aims to engineer the ISD method mainly by adapting the existing method and for a well-known application domain, it is more beneficial to apply an approach that is close to the functional approach, because based on the ISD workflows distinguished in infological modeling it is easier to recognize those parts of the method that should be adapted. Moreover, after re-engineering ISD deliverables it is simpler to make changes in the concerned parts of the IS ontology.

In the ME literature there are only a few considerations of the corresponding approaches. Ralyte *et al.* (2003, 105) suggest the ME strategy according to which "a product model" is constructed before engineering "a process model". This strategy roughly corresponds to our conceptual approach. The process model "can take multiple forms: an informal guideline, a set of ordered actions, a set of process patterns, multi-process guidelines" (ibid p. 105). These are deliverables of the infological ISD modeling.

Another example of applying the conceptual approach to the ISDM analysis is given in Hruby (2000b). Hruby presents a methodological framework, which regards the "software development artifacts" as the most essential constructs. These artifacts are viewed as conceptual, not representational (ibid p. 23). In engineering a method, artifacts are first selected. Artifacts (artifact types) have two kinds of "methods"[172] that rule how to create, interrelate and check the artifacts (instances). For instance, the "methods" specify preconditions that require that certain artifact (instance) must exist before some other artifact (instance) can be created. Thus, preconditions indirectly impact on the order in which the ISD actions creating the artifacts should be performed (e.g. to create a class life cycle, the artifact 'class' must be first created (Hruby 2000b, 29)). When selecting and including artifact (types) to the body of the method, the "methods" attached to the artifacts also indicate the kinds of ISD actions there should be in the ISD method. Hruby (2000b) calls his framework a "product-focused" framework, as compared to the OPEN framework (Graham *et al.* 1997), which he calls the "process-focused" framework.

172 Method here corresponds to the notion of a method in the object-oriented paradigm. To differentiate it from our term, we present it in quotation marks.

11.8.2 Infological ISD Modeling

The purpose of *infological ISD modelling* is to describe and gradually elaborate specifications of ISD purposes, ISD actions and ISD deliverables to include them in the body of the ISD method. The modeling actions deploy concepts and constructs defined in the ISD ontology (see Section 8.4.2). From the ISD action structures, the generic ISD action structures (i.e. the decomposition structure and the control structures), the IS modeling structure, the ISD problem solving structure and the ISD workflow structure are applied. The ISD management - execution structure and the ISD phase structure are excluded in this task. Also the ISD management deliverables are ignored.

The results from the infological IS modeling are commonly presented in ISD action models, ISD deliverable models, and/or some hybrid models. If the ISD purposes have an essential role in the ME effort, they are brought out in some ISD goal models (i.e. in goal/means graphs, Loucopoulos *et al.* 1998; Katzenstein *et al.* 2000; Castro *et al.* 2001). It is, however, more common to describe ISD goals as being associated with ISD actions and/or ISD deliverables. To give concrete examples of ISD models suitable for presenting the results from ISD infological modeling, we give a categorization of ISD models with references to the literature[173]:

- *Action control model (ACM)* describes ISD actions and their control structures (i.e. sequence, selection, iteration); e.g. action diagrams (Martin *et al.* 1985).
- *Action decomposition model (ADM)* describes hierarchical decomposition structures of ISD actions; e.g. structure charts (Yourdon 1989).
- *Information flow model (IFM)* describes ISD actions and information flows (i.e. ISD deliverables) between them; e.g. data flow diagrams (Gane *et al.* 1979).
- *Deliverable decomposition model (DDM)* describes hierarchical decomposition structures of ISD deliverables; e.g. data structure diagrams (Jackson 1983).
- *Deliverable supply model (DSM)* describes ISD deliverables and supply relationships between them; e.g. I-graphs (Lundeberg 1982).

The conceptual contents of, and the relationships between, the ISD models mentioned above are illustrated in Figure 127. In the following we use the setting in the figure to illustrate the approaches and steps of infological ISD modeling.

The basis for infological ISD modeling is obtained from the goal statements made in the ISDM RE workflow. For proceeding, there are three alternative approaches. The approaches differ from one another in regards to which ISD domains the modeling process starts with. The approaches are: (a)

173 Note that the models mentioned in the parentheses have originally been developed for the IS. Here they are applied to model the ISD.

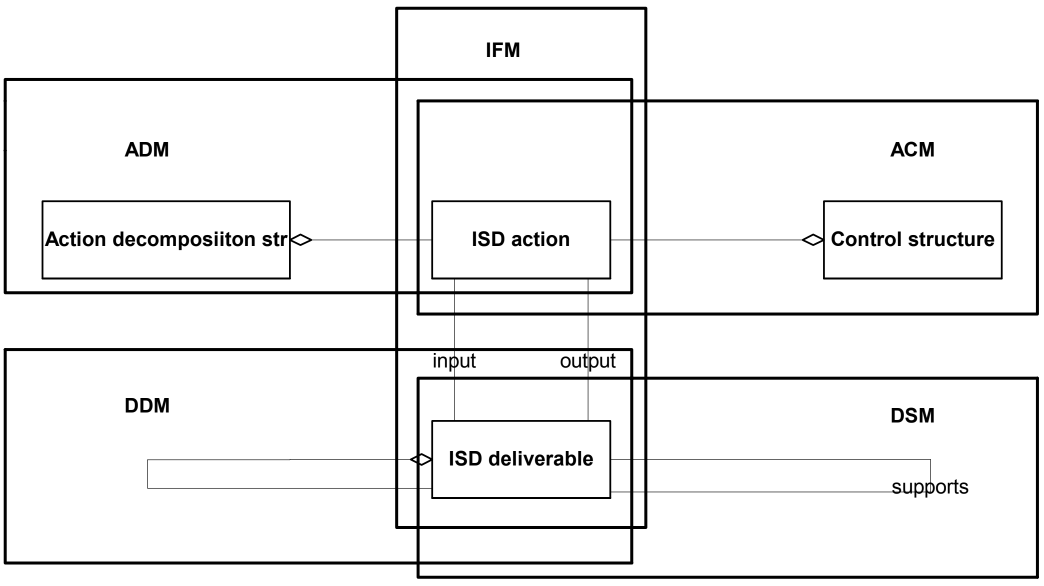


FIGURE 127 Models applicable to the infological ISD modeling

the ISD action-driven approach, (b) the ISD deliverable-driven approach, and (c) the mixed approach. Figure 128 describes the steps of infological ISD modeling and their order in the first two approaches. The steps are called by the names of the artifacts that are the targets of the steps (e.g. 'ISD actions' means infological ISD modeling for specifying ISD actions). The arrows between the steps denote sequence relationships.

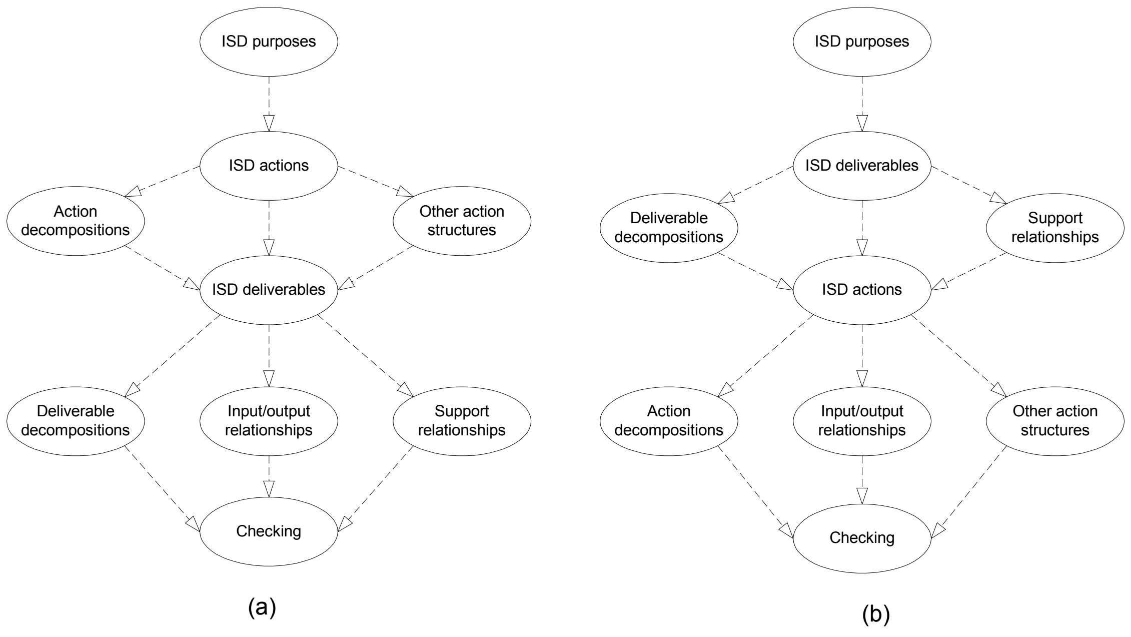


FIGURE 128 Processes (a) in the ISD action-driven approach and (b) in the ISD deliverable-driven approach

According to the *ISD action-driven approach,* infological ISD modeling starts with elaborating ISD purposes and proceeds to identify and specify ISD actions and their decomposition and control structures. After that ISD deliverables are distinguished, structured and specified. In contrast, in the *ISD deliverable-approach* the identification and specification of ISD actions comes after modeling ISD deliverables and their inter-relationships. In the *mixed approach* modeling is carried out more or less in parallel with ISD actions and ISD deliverables. A good example of applying the mixed approach is the ME effort to "build a software maintenance methodology" (Polo *et al.* 2002) which proceeded from first identifying tasks and then to specifying "input and output products", and techniques. After pilot testing, divergent paths for corrective, perfective, preventive and adaptive maintenance types were established. Each path was composed of specialized tasks and products.

A choice between the three approaches depends, to some degree, on the approach applied in the ISDM analysis workflow. If the functional approach has been applied, the mixed approach to the infological ISD modeling is more suitable. If the conceptual approach has been applied, it is more natural here to start the process of infological ISD modeling with considering what are the ISD deliverables signifying those OS_{ISD} constructs that have been identified in conceptual ISD modeling. This means that the ISD deliverable-driven approach is more appropriate. Because it is not possible here to describe for all three approaches, how infological ISD modeling proceeds, we describe the steps in the order they appear in the ISD action-driven approach.

As mentioned in Section 11.2, MEMES has been firmly built upon OntoFrame. This means that ISD modelling, here done from the infological perspective, is guided to utilize the ISD meta models included in the ISD ontology. There are two basic approaches to the utilization of the ISD ontology: the bottom-up approach and the top-down approach. According to the bottom-up approach, ISD modeling starts with specifying instance-level and specialized concepts and constructs and proceeds to abstract generic concepts (e.g. workflow, activity, task, operation) generalized from them. In the top-down approach, generic concepts and constructs are first specified and later made more concrete by instantiation and specialization. We favor the top-down approach, because this way it is easier to ensure that the concepts and constructs constitute a unified and coherent whole. Our ISD ontology offers a basis for adaptations when specifying an ISD context-specific ontology. Actually, also the bottom-up approach can benefit from our ISD ontology as it provides concepts and constructs toward which the process of classification and specialization orientates when searching for a coherent set of generic concepts. It is not possible for us to show how the utilization of the ISD ontology takes place in all the cases. Instead, we give some examples of the utilization when following the ISD action-driven approach to the infological ISD modeling with the following steps:

- *Elaborate and categorize ISD goals.*

  Based on the ME goals stated in the ISDM RE workflow, reconsider, elaborate and categorize those goals that pertain to the target ISD context.

  Consider which concepts and constructs, defined in the ISD purpose domain in the ISD ontology, as such or adapted, apply to this case. In order to help infological ISD modeling in the following steps, categorizing the goals according to the contextual domains is recommended.

- *Identify and specify ISD actions.*
  Deriving from the ISD goals stated above identify and specify ISD actions that are needed to satisfy the goals.
- *Decompose ISD actions.*
  To manage the complexity related to the ISD actions and to get a more detailed view of them, decompose the ISD actions into more elementary parts (e.g. ISD tasks and ISD steps).
- *Establish other ISD action structures.*
  There are several other action structures in the ISD ontology by which ISD actions can be organized, e.g. the ISD workflow structure, the ISD problem solving structure, the IS modeling structure, and the control structures. A decision on which of these is used as the primary ISD action structure depends on the ISD approach(es) (of the category B, see Section 8.1.2) selected in the ISDM RE workflow. In the transformation approach, for instance, the IS modeling structure may be favored, while in the problem solving approach the ISD actions are structured according to the process by which ISD problems are solved and decision are made. To have an overall structure for the whole ISD effort, ISD workflows are usually used[174]. The control structures (i.e. sequence, selection and iteration) are specified to express monotonic, alternative, and cyclic processes.
- *Identify and specify ISD deliverables.*
  Deriving from the ISD goals and the ISD actions identified above, decide what ISD deliverables are needed. At the beginning conceptions about the ISD deliverables are quite general. The generic notions of ISD deliverables in the ISD ontology can be used to specialize ISD deliverables into more concrete concepts. One way of recognizing ISD deliverables is to consider, for each identified ISD action, what are the outputs from, and what are the inputs to it.
- *Decompose ISD deliverables.*
  To get a more detailed view of the ISD deliverables, decompose them into smaller informational objects. Decomposition should be in compliance with the structures of the corresponding ISD actions.
- *Specify support relationships.*
  To show what ISD deliverables are needed to produce other ISD deliverables, associate ISD deliverables to one another with the support relationships (e.g. I-graphs in Lundeberg (1982)).
- *Specify input / output relationships.*
  Associate the ISD actions and the ISD deliverables to one another with the input and output relationships.

[174] In the early days it was common to first establish a phase structure.

- *Check the specifications of ISD actions and ISD deliverables.*
  Reconsider the specifications of the ISD actions and the ISD deliverables in order to check their compliance with the stated ISD goals, comprehensiveness, and internal and external consistency. Rename concepts and reorganize structures and relationships, if necessary.

The steps above are presented in the order in which they are commonly executed when applying the ISD action-driven approach to infological ISD modeling. An actual ME process may, naturally, deviate to a large degree from this order and several steps are carried out concurrently. Also moving up and down in the hierarchies of ISD actions and ISD deliverables is common.

11.8.3 Conceptual ISD Modeling

The purpose of *conceptual ISD modeling* is to specify the conceptual contents of ISD deliverables. ISD deliverables are informational objects referring to the IS contexts, as well as to the ISD context. The latter objects primarily correspond to ISD management deliverables. Because we do not address the management part of the ISD context in this stage, we concentrate on ISD execution deliverables. The object system of the ISD context, OS_{ISD}, embraces three parts, the IS, the OS_{IS} and the US_{IS}. These together constitute an IS ontology. Consequently, conceptual ISD modeling means IS ontology engineering.

We have defined the IS ontology to be composed of the concepts and constructs in seven IS domains, from five IS perspectives. The IS systelogical perspective concerns the business system (US_{IS}) and the support the IS provides for it. The IS infological, IS datalogical and IS physical perspectives uncover various aspects of the IS. From the IS conceptual perspective the concepts and constructs of the object system (OS_{IS}) are recognized.

IS ontology engineering is neither an easy nor straightforward task for several reasons. First, a way of engineering an IS ontology depends on whether the ISD addresses the CIS, the HIS, or both of them. In the former case, the IS ontology is technology-oriented, whereas in the latter cases the IS ontology also contains social and organizational concepts. Second, IS ontology engineering differs in the extent to which the ISD method is to support also re-engineering of business processes. For instance, if the IS is considered just a 'tool' to be taken into use without too much interest in consequences to organizational or functional issues of the business system, the role of systelogical IS ontology engineering remains insignificant. Third, selecting and defining the concepts and constructs of the IS are affected by how the IS is actually seen. As mentioned in Section 8.1, there are different view-based ISD approaches. For example, the IS can be seen as a context for data processing, enabling the collection, recording, processing, and dissemination of information to the end-users. Alternatively, the IS can be considered as a context of cooperation, collective decision making, and knowledge sharing. It is clear that these views lead to the selection and definition of different concepts and constructs.

IS ontology engineering involves the IS domains on different stages. Normally, engineering starts with defining concepts in the so-called core domains: in the IS purpose domain, the IS action domain, and the IS object domain (cf. Section 4.3). After that, engineering proceeds to the consideration of the related IS domains. Expressed in terms of IS perspectives we can say that it is more common to start with the IS systelogical perspective and proceed through the IS infological and IS conceptual perspectives to the IS datalogical and IS physical perspectives. To have an overall view of different approaches to IS ontology engineering we categorize ME actions according to the IS perspectives (see Figure 129). We distinguish between four approaches to the IS ontology engineering depending on where the process starts from. The approaches are: the US-driven approach, the IS-driven approach, the OS-driven approach, and the CIS-driven approach. In the following we shortly characterize these approaches.

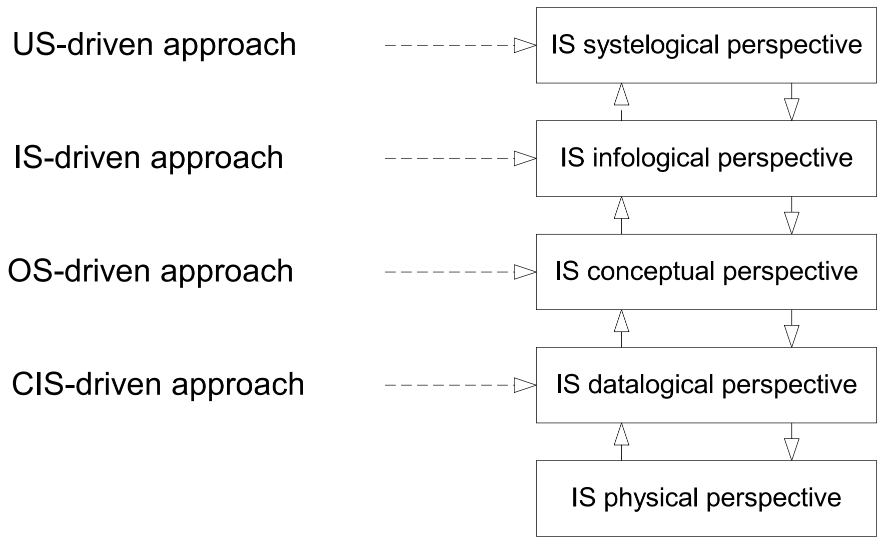

US-driven approach
IS-driven approach
OS-driven approach
CIS-driven approach
IS systelogical perspective
IS infological perspective
IS conceptual perspective
IS datalogical perspective
IS physical perspective

FIGURE 129 IS ontology engineering approaches and IS perspectives

In the *US-driven approach* the alignment of the IS to the business system utilizing the services of the IS is seen vital in IS ontology engineering. Therefore, the process starts with applying the IS systelogical perspective. This is the case, for instance, in developing methods for strategic information systems and decision support systems. According to the *IS-driven approach,* IS ontology engineering starts with identifying the main concepts of IS actions and IS objects, as well as with defining input and output relationships between them. This approach can be applied in engineering methods for information systems, which contain complex and abnormal information processing or information structures. In the *OS-driven approach* IS ontology engineering starts with defining the concepts that are used to model the conceptual contents of the informational objects of the IS (cf. the IS conceptual perspective). Later, it proceeds with defining informational objects representing the conceptual constructs and IS actions processing them. This approach is applicable in situations where informational objects in the IS refer to unusual conceptual constructs. We can assume that in engineering methods for GIS development, it is beneficial first to carry out OS metamodeling to specify complex conceptual constructs underlying spatial data. The *CIS-driven approach* is focused on architectural and technical features of the computerized information system. This approach is common in engineering the ISD method for technical system design (e.g. for architecture design).

Due to the plurality of approaches, each with different orientation and emphasis on the IS perspectives, we describe the tasks of conceptual ISD modeling in four parts. In the first part, instructions are given for the selection of approach(es) and making decisions on the order in which steps in the other parts are executed. The next five parts provide steps for IS ontology engineering. The parts are: systelogical IS ontology engineering (or business system metamodeling), infological IS ontology engineering, conceptual IS ontology engineering (or object system metamodeling), datalogical IS ontology engineering, and physical IS ontology engineering.

I. Generic decisions:

- *Select a view through which the IS is primarily considered (e.g. transformation view, problem solving view, decision making view (see Section 8.1)).*
- *Deriving from the view selected, decide on the IS perspectives to be applied in IS ontology engineering.*
- *Define the aim and scope of the IS perspectives.*
- *Decide in which order the perspectives are to be applied in the IS ontology engineering (cf. US-driven, IS-driven, OS-driven, and CIS-driven approaches).*

II. Systelogical IS ontology engineering (cf. Section 6.3.1):

- *Define the most essential concepts by which the utilizing system of the IS can be modeled.*
  The concepts and constructs to understand and represent the structural, functional and behavioral features of the utilizing system are defined. They refer to e.g. US goals and US requirements, US roles, US positions and organizational units, US actions and US objects. Also intra-domain and inter-domain relationships are defined.
- *Define the essential US rules.*
  US rules are defined and structured in the ECAA (Event, Condition, thenAction, elseAction) form, or in some variant of that form.
- *Define US concepts that elaborate the view of the US as a physical, locational and temporal context.*
  Elaboration is done with the concepts of the US facility domain, the US location domain and the US time domain. Through the concepts of the US facility domain, for instance, it is possible to define the functional role in which the IS is deployed in the US.
- *Complete the definitions of the intra-domain and inter-domain relationships between the US concepts.*

III. Infological IS ontology engineering (cf. Section 6.3.2):

- *Define the essential concepts which are used to refer to IS actions* (e.g. function, activity, task, step, operation).
- *Define the essential concepts which are used to refer to IS objects* (e.g. outcome, deliverable, information set, message, decision).
- *Define the essential concepts which are used to refer to IS purposes* (e.g. problem, goal, intention, motive).
- *Define the main IS action structures.*
  There is a large set of IS action structures available (see Section 4.4.3): e.g. the generic structures (the decomposition structure, the control structures, and the temporal structures), the IS problem solving structure, and the IS management – execution structure. Select the appropriate structures and elaborate them to be included in the method.
- *Define the main structures of information processing rules.*
- *Define the main specializations and structures of the IS objects.*
  In congruence with the defined IS actions structures, decide on which sub-concepts of the IS objects and relationships between them (e.g. partOf, supports, copyOf, versionOf) are needed. Define the concepts and the relationships.

IV. Conceptual IS ontology engineering (cf. Section 6.3.3):

- *Find out whether there exists a suitable IS meta data model.*
  In the literature there is a large set of meta data models (e.g. ER model (Chen 1976), EER (Elmasri *et al.* 2000), NIAM (Nijssen *et al.* 1989), OPRR (Welke 1988, Smolander 1991), GOPRR (Kelly *et al.* 1996), UML class diagram (Booch *et al.* 1999)), which provide either a ready solution to the needs of the method, or a basis for adaptations. If no such meta model is found, the following steps should be accomplished:
  (a) Define the concept(s) with which independent and distinguishable things in reality are modeled (e.g. entity (type), concept (type), thing (type), phenomenon (type)).
  (b) Define the concepts with which relationships between the things are modeled (e.g. relationship (type), connection (type), association (type), role).
  (c) Define the concepts with which abstraction structures are modeled (i.e. classification, generalization, composition, and grouping).
  (d) Define the concepts with which characteristics of the things and relationships are modeled (e.g. attribute, property, feature).
  (e) Define the main structures with which static OS_{IS} constraints can be modeled (e.g. cardinality constraint, role constraint, attribute constraint).
- *Find out whether there exists a suitable dynamic OS model.*
  If the ISD method should also comprise concepts for modeling the dynamic features of the OS_{IS}, a variety of dynamic OS models (e.g. UML state diagram, Booch *et al.* 1999) are evaluated. If no suitable model is found, the following steps are carried out:

(a) Define the main concepts with which the states of things are modeled.
(b) Define the main concepts with which the shifts between the states are modeled.
(c) Define the main concepts with which triggering the shifts between the states is modeled.

V. Datalogical IS ontology engineering (cf. Section 6.3.4):

- *Define the main concepts, with which humans acting in the IS contexts can be modeled* (e.g. agent, actor, stakeholder, user, end-user, role).
- *Define the main constructs with which organizational structures can be modeled* (e.g. organization, unit, team, group).
- *Define the main structures with which IS actions can be decomposed into more elementary things* (e.g. position, task, operation, step, work procedure, process, event).
- *Define the main concepts with which interaction between the human beings and the CIS can be modeled* (e.g. dialog, window, UI component, UI data, UI action, UI state, UI transition).
- *Define the main concepts with which the structure and behavior of the CIS can be modeled on a general level* (e.g. CIS action, CIS rule, transaction, algorithm).
- *Specify the languages (incl. notations) in which IS models can be presented.*

VI. Physical IS ontology engineering (cf. Section 6.3.5):

- *Define the concepts with which processes enacted by persons in a certain spatio-temporal space are modeled.*
- *Define the concepts with which persons, groups and their relationships are modeled.*
- *Define the concepts with which data and data structures in different forms and on different granularity levels are modeled* (e.g. data file, data base, record, data field, data message etc.).
- *Define the concepts with which the physical structure and behavior of the CIS is modeled* (e.g. node, processor, memory device, SW component, HW component, protocol).

The lists of the steps of the IS ontology engineering given above are not to be exhaustive, but rather to provide a concrete view of the wide range of issues that might be considered in the accomplishment of this task. On the other hand, it is not the purpose that in every ME context all the steps should be carried out. In contrast, whenever a part of the IS ontology is found in the literature that suits the needs of the ISD method, one should be encouraged to fully utilize it.

11.8.4 Inter-Perspective ISD Modeling

The ME tasks described above are related to specific ISD perspectives. To have an integrated view of the ISD context, it is necessary to ensure that the perspective-based ISD models are inter-related in a consistent manner. That is the aim of *inter-perspective ISD modeling.*

The ISDM analysis workflow comprises ME tasks of infological ISD modeling and conceptual ISD modeling. The ISD domains primarily concerned are the ISD action domain and the ISD object domain. Things in the ISD object domain are seen as linguistic artifacts and conceptual constructs. To ensure the consistency between the perspective-based ISD models, it is necessary to check that the defined concepts and constructs of those ISD domains are appropriately inter-related.

In Section 11.8.1 we distinguished between the two main approaches to the ISDM analysis, the functional approach and the conceptual approach. Here we show how, with these approaches, it is possible to carry out inter-perspective ISD modeling. In Figure 130 the ME processes following the approaches are illustrated with two settings composed of eight squares (cf. Leppänen 2000). A square stands for a specific sub-area of the ISD context. With the symbols in the squares we express the concepts and constructs in these sub-areas. The meanings of the symbols are[175]:

- $A \times A_A$. ISD actions and their abstraction relationships (i.e. decomposition, and specialization)
- $A \times A_R$. ISD actions and their relationships based on the control structures, the ISD problem solving structure, the ISD workflow structure, and the IS modeling structure.
- $A \times D$. ISD actions, ISD deliverables and their inter-relationships.
- $D \times D_A$. ISD deliverables and their abstraction relationships (i.e. decomposition and specialization)
- $D \times D_S$. ISD deliverables and their supports relationships.
- $D \times C$. ISD deliverables, OS_{ISD} constructs and the signifies relationships between them
- $A \times C$. ISD actions, OS_{ISD} constructs and the involvedBy relationships between them.

The setting on the left side (Figure 130a) illustrates the process of the ISDM analysis carried out according to the functional approach. The first tasks produce descriptions / prescriptions about ISD actions (A) and ISD deliverables (D). At this stage abstraction by decomposition and specialization can be applied to the ISD actions ($A \times A_A$) and the ISD deliverables ($D \times D_A$). In addition, a variety of action structures can be defined among the ISD actions ($A \times A_R$), and supports relationships are defined between the ISD deliverables ($D \times D_S$). After having defined the main concepts and relationships within the ISD

[175] We have used Cartesian product in the symbols to highlight that the symbols stand for conceptual constructs within and between the concerned domains. We interpret Cartesian product freely so that it covers the n-ary relationships ($n \geq 2$) between the contextual concepts. The abbreviations used in the subscripts are: A = Abstraction (relationship), S = Supports (relationship), R = Rest (of the intra-domain relationships).

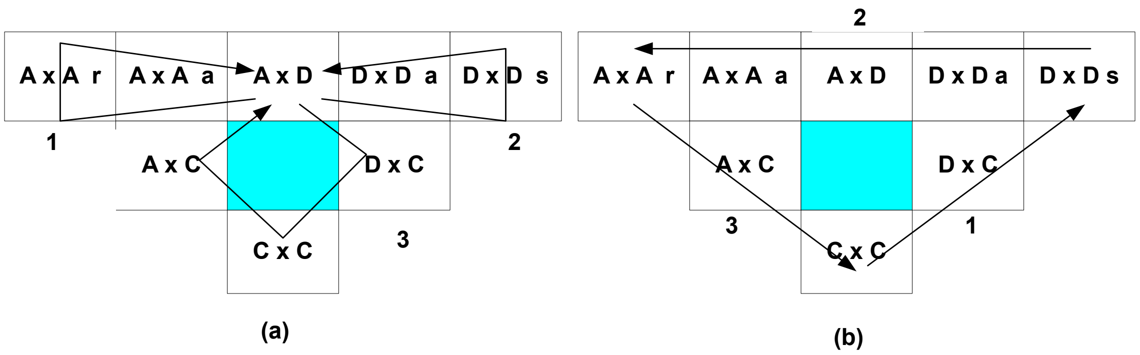


FIGURE 130 (a) The functional approach and (b) the conceptual approach

action domain and the ISD object domain, the ISD analysis proceeds to increase the understanding of the meaning of the ISD deliverables through conceptual IS ontology engineering. The results from the task comprise OS_{ISD} constructs (C x C).

The setting on the right side (Figure 130b) illustrates a process of the ISDM analysis carried out according to the conceptual approach. The process starts with recognizing phenomena (C x C) in which we are interested during the ISD effort. With this better understanding of the structure and behavior of the IS it is easier to decide which ISD deliverables should be established to signify those phenomena (D x C). Having established the ISD deliverables with relationships to the object system of the ISD, it is then possible to define the supports relationships (D x D_S) and the abstraction relationships (D x D_A) between the ISD deliverables. To complete the picture, ISD actions are defined based on what their inputs and outputs (A x D) are. To verify the total view from the ISD infological and ISD conceptual perspectives, the involvedBy relationships between the ISD actions and the OS_{ISD} constructs (A x C) are defined.

ME tasks along the path of applying one or the other of the aforementioned ME approaches are accomplished with the steps of the infological ISD modeling and the conceptual ISD modeling, respectively. In addition, it is necessary to verify the consistency between the ISD infological and ISD conceptual views. This is carried out with the following steps:

- *Ensure that for each ISD deliverable there is a non-empty set of OS_{ISD} constructs which the ISD deliverable signifies.*
- *Ensure that for each OS_{ISD} construct there is a non-empty set of ISD deliverables that signify it.*
- *Ensure that each OS_{ISD} construct is involved by at least one ISD action.*
- *Ensure that each ISD action involves at least one of the OS_{ISD} constructs.*

There are several models that can support carrying out the aforementioned verification. The data class/business entity matrix (IBM 1984), for instance, can be used to model and analyze the signifies relationships between the ISD deliverables and the OS_{ISD} constructs. Likewise, the process/data class matrix (IBM 1984) can be used to describe the involvedBy relationships between the ISD actions and the OS_{ISD} constructs. In the latter matrix, special markings (R, C, U, D) can be used to denote whether a certain ISD action requests, creates, changes or deletes information about a certain OS_{ISD} construct.

11.9 ISDM Evaluation

ISDM evaluation means ME actions, which aim to assess one or more ISD methods, or parts thereof, according to the defined criteria. ISD methods can be evaluated in various contexts for different purposes. Evaluation can be conducted, for instance, on the bases of drafts made from the method being engineered. Evaluation plays an important role also in the selection of the method for the use of the specific ISD project. Furthermore, it is common, during the method use, to make assessments of it for future acts of improvements. There is a large variety of ways and techniques that can be applied in the evaluation (Sol 1983; NCC 1987; Law 1988; Avison *et al.* 1995a; Kitchenham 1996a; Kitchenham 1996b; Kitchenham 1996c). To clarify a variety of contexts and targets of the evaluation, we present a meta model in Figure 131. In what follows we define the concepts in the meta model, refer to the relevant literature and present a set of generic steps to be applied in ISDM evaluation.

An *evaluation context* is a situation, the purpose of which is to assess and/or compare one or more ISD methods, or parts thereof. There are two main reasons for evaluation: academic reasons and practical reasons (Avison *et al.* 1995a, 434). In the former case, the purpose of evaluation is to obtain a better understanding of the nature of the ISD method(s) in order to improve future methods. In the latter case, evaluation aims to choose a method and/or to provide the basis for making decisions on necessary improvements in the evaluated method. Whereas the categorization above is based on the different aspects in the ISD purpose domain, differences between the evaluation contexts can be based on the other contextual domains as well. For instance, evaluation can be performed in an academic organization, a method vendor organization, a software house, or a client organization (cf. Kitchenham 1996b). Evaluators in the context can be academic people, method developers, IS developers, and/or IS users.

The main target of evaluation is a method or a method component. Let us call them *methodical things.* There is a myriad of literature on evaluation of methods in general (e.g. Olle *et al.* 1983; Olle *et al.* 1986; Jayaratna 1994; Blum 1994; Hughes *et al.* 1996; Bielkowics 2002; Moody 2003a), or methods of certain types (e.g. object-oriented methods (Arnold *et al.* 1991; de Champeaux *et al.* 1992; Hong *et al.* 1993; Iivari 1994; Liang 2000; Henderson-Sellers *et al.* 2001), component-based methods (Forsell *et al.* 2000; Boertien *et al.* 2001), agent-based methods (Tran *et al.* 2003; Dam *et al.* 2004; Sturm *et al.* 2004). The most common method components in evaluations are models (e.g. conceptual models (see

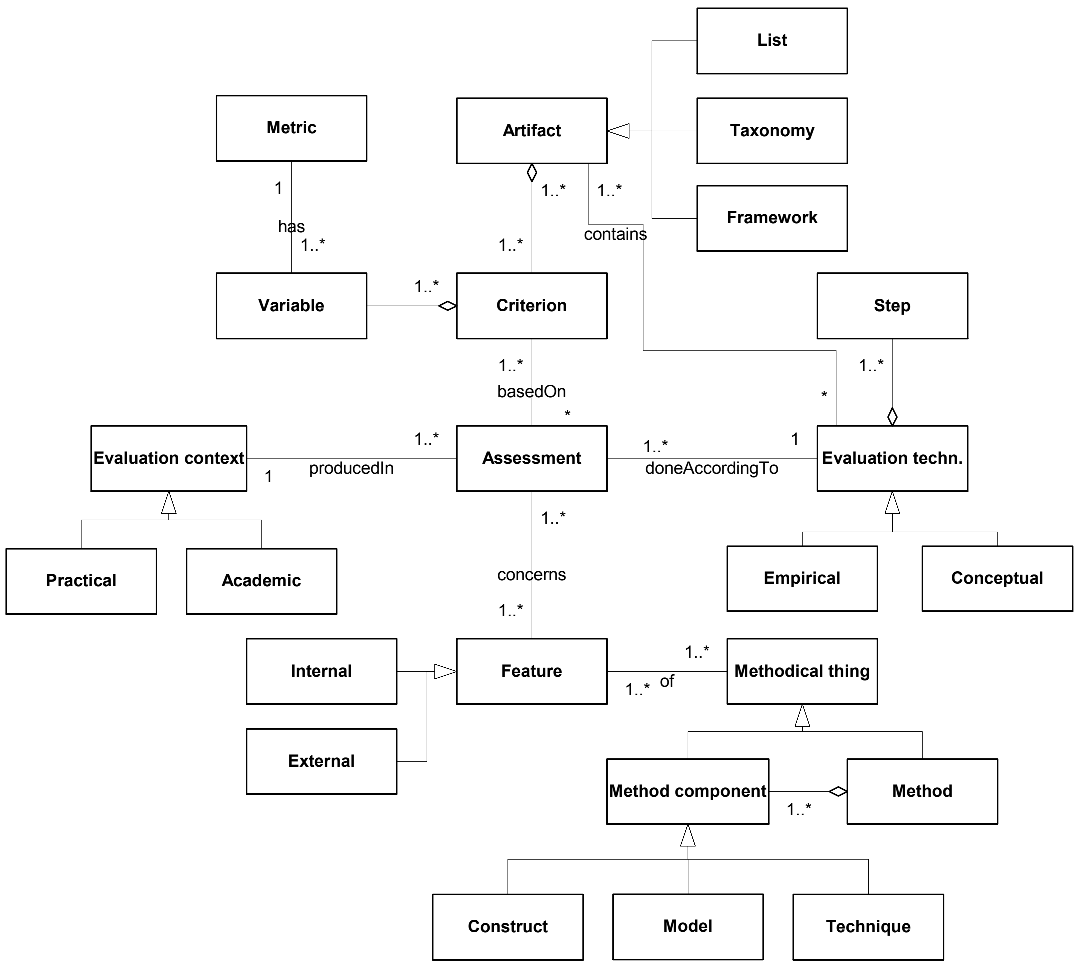


FIGURE 131 Meta model of issues related to the evaluation of an ISD method

references in Gemino *et al.* 2002; Moody 2003b; Moody 2000c), process models (e.g. Green *et al.* 2000), enterprise / business models (e.g. Hommes *et al.* 1999; Hommes *et al.* 2000; Arnesen *et al.* 2002), and techniques (data requirements specification techniques (e.g. Bielkowicz *et al.* 2001), user interface analysis techniques (e.g. Jefferies *et al.* 1991). A methodical thing can exist at different points in its life cycle and appear in different forms. It may be a draft on paper or a heavily used method, perhaps embedded in a CASE tool. Evaluation can be made on the basis of primary descriptions (i.e. manual), trials of a CASE tool implementing the methodical thing, method specifications recorded in a CAME tool (Harmsen 1997; van Slooten *et al.* 1993), or based on some secondary sources, e.g. characterizations presented in articles and books.

Of a methodical thing one or more features can be selected for evaluation. A *feature* means any property of a method, or of a method component, that is considered relevant in the evaluation context. Features can be categorized along several dimensions, partly depending on the kind of methodical thing: structural, functional vs. behavioral features; systelogical, infological, conceptual, datalogical vs. physical features; internal vs. external features; etc. *Internal features* mean properties that can be evaluated without associating the methodical thing with any other thing or empirics (e.g. consistency and coherence of a conceptual model). *External features* mean properties that can be evaluated only by comparing them with others or deploying them (e.g. applicability and understandability).

Evaluation of features should be based on well-defined evaluation criteria. A *criterion* means an explicitly specified standard of evaluation (cf. Webster 1989). One or more criteria can be used to evaluate a certain feature, and one or more features can be evaluated with the same criterion. Examples of criteria are readability (Bajaj 2002), expressiveness (Chaves *et al.* 1996), effectiveness (Gemino *et al.* 2002), efficiency (Moody 2003a) and correctness (Moody 2003b). For each criterion there are one or more *variables* with which "measurement" is to be performed (e.g. for measuring correctness of a data model there are variables, such as a number of violations to data modeling standards, a number of instances of entity redundancy, and a number of instances of relationship redundancy (Moody 2003b)). For each variable there is a *metric*, which specifies the data type, range, etc. of the variable. Variables can be qualitative or quantitative.

Individual criteria are commonly structured into the form of an *evaluation artifact.* In the simplest form an artifact is a list of criteria (e.g. Rzevski 1983; Ang 1993; Brodie 1983; Karam *et al.* 1993; Flynn *et al.* 1993). More structured forms of the artifacts are a taxonomy (e.g. Brandt 1983; Blum 1994), a hierarchy (e.g. Law 1988) and a framework (e.g. Iivari *et al.* 1983; Lindland *et al.* 1994; Jayaratna 1994; Krogstie 1995; Krogstie *et al.* 2000; Wieringa 1999; Bielkowicz 2002). A structure and contents of criteria are sometimes grounded on some theory (e.g. sociocybernetics in Iivari *et al.* (1983), semiotic ladder in Krogstie (1995)).

In an evaluation context one or more evaluation techniques can be deployed. An *evaluation technique* provides criteria, principles, guidelines and steps to carry out an evaluation. Examples of evaluation techniques are SDSS (Law *et al.* 1984), STARTS (NCC 1987) and DESMET (Kitchenham 1996a). The use of an evaluation technique can be undertaken as an empirical or conceptual study. Examples of *empirical studies* are formal experiments, case studies, action research, and surveys. In a formal experiment participants are asked to perform a task or a set of tasks using the methodical thing under investigation. In a case research and in an action research (e.g. Grant *et al.* 2003; Tolvanen 1995) the method under evaluation is used and assessed in a real project where the method developer participates. In a survey ISD stakeholders having experience in specific methodical thing(s) are asked to provide information about it. Empirical studies aim to uncover method appropriateness (i.e. the fit with the needs and culture of the organization) and measurable effects of using a methodical thing. In a *conceptual study* an evaluator focuses on those features (i.e. internal features) in a methodical thing that do not necessitate empirical evidence for assessments.

A variety of evaluation contexts is so huge that it is quite impossible here to provide detailed guidelines for the accomplishment of the ISDM evaluation workflow in all kinds of contexts. Instead, we present a set of steps for ISDM evaluation on a general level with the purpose that the steps can be, in a case-by-case manner, instantiated to suit a particular evaluation context. The steps are:

- *Analyze the evaluation context at hand.*
  Due to a large variety of evaluation contexts it is necessary to find out what kind of context is currently being dealt with. For this purpose, goals, target and constraints of evaluation are specified. These issues have a great impact on what actions are necessary to carry out in the following.
- *Decide what features in general are important to the evaluation.*
  Having specified the target of the evaluation one decides what features of the methodical thing(s) are to be evaluated. These features can be specified and structured according to the ISD method ontology (Section 9.5).
- *Select an evaluation technique.*
  Based on the goals, target, and constraints of evaluation, as well as on the specified features of the methodical thing(s), it is decided which kind of evaluation technique is needed. First, a decision is made between the empirical techniques and the conceptual techniques. Second, search for the literature is carried out to find an appropriate evaluation technique. If not found, the one(s) that come(s) closest can perhaps be adapted.
- *Specify evaluation criteria.*
  A search is conducted to find an appropriate evaluation artifact (e.g. a list, a taxonomy, a framework, etc), which provides suitable evaluation criteria. This artifact is perhaps contained by the evaluation technique selected, or found from the literature. If not found, necessary adaptations are applied to some other artifact(s). For each criterion, the variables, metrics and usage guidelines are specified.
- *Fix the methodical thing(s) for the evaluation.*
  The methodical thing can be a method, or some part of it. Depending on the nature of the evaluation context and the goals of the evaluation, there may be one or more methodical things that should be evaluated. If several methodical things are to be evaluated, the selection of them is based on the availability, preconception of their match with the needs, and resources available for the evaluation.
- *Carry out the evaluation.*
  For each selected methodical thing the evaluation is made according to the defined criteria. If the set of the things is large, the evaluation is decomposed into two parts. In the first part an investigation is conducted on the basis of the most important features in order to decrease the number of the methodical things. In the second part an in-depth evaluation is carried out only for the most potential things.
- *Make a summary of, and conclusions from, the evaluation.*
  To help the utilization of results from the evaluation the assessments should be documented in a structured form. Documentation should also contain arguments by which the selections and assessments have been made (cf. method engineering rationale, Rossi *et al.* 2004).

11.10 Summary

In this chapter we have described the methodical skeleton for method engineering, called MEMES. It has been built upon OntoFrame, and in particular, upon the ISD ontology, the ISD method ontology, the ME ontology, and the ME method ontology. The purpose of MEMES is to provide support to the engineering of generic and domain-specific methods with any ME strategy. MEMES views the ME context from three ME perspectives and from three ISD perspectives. It covers three ME workflows that are the ISDM requirements engineering, the ISDM analysis and the ISDM evaluation.

The ISDM requirements engineering starts with making decisions on the feasibility of a contingency framework to the ME context at hand. One or more frameworks can be selected to aid the elicitation and structuring of the characterizations of the concerned contexts (i.e. the prior contexts, the ME context at hand, and the target contexts) and the concerned methods. Next, the ME context at hand is analyzed to figure out, for instance, what is the level of knowledge of, experience from, and skills in method engineering in the organization. Also a selection between the problem-driven approach and the policy-driven approach is made. In the next tasks the target contexts and the prior contexts on three layers are characterized and analyzed. Based on the information about the aforementioned contexts, requirements for the ISD method are specified and classified. Next, existing ISD methods are analyzed to find out to what extent they can be used as the basis for integration and/or adaptation in the ME context. Finally, ME goals are stated and structured according to the processing layers and the contextual domains.

The ISDM analysis aims to produce high-level descriptions about the ISD method. This workflow is composed of three main tasks, called the infological ISD modelling, the conceptual ISD modelling and the inter-perspective ISD modelling. For the workflow, two competing approaches, the conceptual approach and the functional approach, have been specified. Each of the tasks is supported by discussions of relevant issues and step-by-step procedures. In the descriptions it is also shown how OntoFrame can be utilized in engineering different parts of ISD methods.

The ISDM evaluation in practice is composed of a large variety of tasks. MEMES provides the meta model of the main issues related to the evaluation of the ISD method, as well as a set of generic steps for the evaluation. The meta model covers concepts such as an evaluation context, an evaluation criterion, an evaluation technique, a methodical thing, and a feature. Each concept has been defined. The steps of the ISDM evaluation are to be instantiated to suit a particular evaluation context.

MEMES was described in the structure of the methodical views defined in Chapter 9. Hence, we first characterized the background (cf. the historical view), application area (cf. the application view), and basic assumptions and approaches (cf. the generic view) of the artifact. After that we detailed the conceptual contents (cf. the contents view) and functional structure (cf. the structural view) of MEMES. By this we demonstrated that the concepts and constructs specified in OntoFrame are fully applicable to the construction and presentation of new artifacts. More about the ways in which we utilized OntoFrame and MEMES themselves in the process of engineering MEMES will be said in the next chapter.

To our knowledge, MEMES is the only ME artifact which provides comprehensive support, in terms of contextual issues, for method engineering. It is also the only one, which has been constructed in the deductive fashion, based on the sound theoretical foundation. In the next chapter we will present the results from a large comparative analysis of existing artifacts, which show that MEMES compares favorably with the other artifacts also in the other respects.

12 EVALUATION OF MEMES

In the preceding chapter we have defined MEMES (**M**ethod **E**ngineering **ME**thodical **S**keleton) as a normative prescription for the ME context and described it in terms of ME workflows, which structure and guide the ME process. In this chapter our purpose is to evaluate MEMES, in particular its applicability. In Chapter 1 we defined the goals for the applicability of MEMES in terms of framing, analytical and constructive intentions. Our evaluation in this chapter covers all of these three intentions.

The chapter is organized as follows. In Section 12.1 we apply one of the ME workflows in MEMES, known as ISDM evaluation, to make sense of and structure the evaluation context at hand. In Section 12.2 we use MEMES as a frame to describe and analyze the processes of, and the deliverables from, the OSSAD project. In Section 12.3 we describe and evaluate how MEMES performed, in the constructive sense, in the MEMES effort. In Section 12.4 we use MEMES and OntoFrame as the analytical framework to make a comparative analysis of those existing artifacts in the ME literature, which are aimed to provide methodical support for ME. The chapter ends with a summary and discussions.

12.1 Evaluation Context

In Section 11.10 we presented the meta model of issues related to the evaluation of an ISD method and the set of generic steps for the evaluation. The steps to perform are: analyze the evaluation context at hand, decide what features in general are important to evaluation, select an evaluation technique, specify evaluation criteria, fix the methodical thing(s) for the evaluation, carry out the evaluation, and make a summary of and draw conclusions from, the evaluation. We consider this situation as an evaluation context and apply these steps to make sense of and carry out the evaluation of MEMES.

The evaluation context at hand is a research context (RW context), the purpose of which is to assess the ME methodical skeleton developed in this work. This is a one-person effort, in which the engineer of MEMES also acts as the evaluator. In Section 11.6 we stated the goals for MEMES in terms of internal and external properties. The goals that are based on the internal properties are: MEMES should be based on a solid and sound view of the relevant sub-domains, and MEMES should be modular and flexible. These goals are satisfied by the use of OntoFrame as the conceptual foundation. The goal concerning the external properties is: MEMES should be applicable. In this chapter we concentrate on the evaluation of the applicability of MEMES for three intentions of use. The intentions are: framing intention, constructive intention, and analytical intention. The applicability for the *framing intention* means that MEMES should provide concepts and constructs, which help us make sense of and structure the phenomena of ME in reality. The applicability from the viewpoint of the *constructive intention* means that MEMES should support the engineering of a methodical artifact. The applicability for the *analytical intention* means that MEMES should provide concepts and constructs for the analysis and comparison of existing ME artifacts. Next, we tell how we will evaluate the applicability of MEMES from these viewpoints.

To evaluate how MEMES serves as a frame for conceiving the phenomena of ME, we make a retrospective analysis of one of our prior ME contexts. As mentioned in Chapter 11, the author has been involved in four prior ME efforts. In the first context the aim was to develop a language and a "design model" for a conceptual schema design (Leppänen 1984a). The second context dealt with an international project, which pursued a methodology for the analysis and design of office support systems (Conrath *et al.* 1989). The third context was a project which engineered a method for strategic planning of information technology and services (Leppänen *et al.* 1991). In the fourth context a method for database application design for teaching purposes was constructed (Leppänen 1993; Leppänen 2001). We select the OSSAD project as the target of our retrospective analysis for the following reasons. The OSSAD project engineered a domain-specific method without considering too much how to customize or configure it. This matches with the purpose of MEMES. Second, the other contexts either concentrated on specifying a language, not a method (cf. the first context), or on engineering methods mainly by the integration strategy (cf. the third and fourth ME contexts). Because MEMES does not offer specific support for method integration, we do not consider them here. The selected prior ME context is analyzed in Section 12.2.

To evaluate how MEMES performs in the constructive sense, we make a retrospective analysis of the effort, which yielded MEMES. As said in Chapter 11, we have applied MEMES in the engineering MEMES itself. Hence, in this effort there were actually two processes, parallel to each another, the RW process and the reflection process. In the RW process we constructed, step by step, the ME skeleton, and in the reflection process we tried to learn about this RW process and its outcome in order to elaborate the skeleton. The whole effort was accomplished iteratively following the reflection-in-action approach (Schön 1983). The MEMES effort is analyzed in Section 12.3.

To evaluate the applicability of MEMES in the analytical sense, we carry out a comparative analysis of existing ME artifacts. Although in the ME literature there is no complete ME method, nor anything that would come even close to it, we select the most advanced suggestions for ME artifacts for our analysis. In this conceptual evaluation we compare MEMES with existing ME artifacts for two purposes: to examine the usefulness of MEMES as a frame, and to find out how MEMES compares with the existing ME artifacts. We describe the selected ME artifacts as well as the criteria, process and results of the comparative analysis in Section 12.4.

12.2 Evaluation through the OSSAD Project

In this section we make a retrospective analysis of the OSSAD project to evaluate MEMES as a frame. First, we outline the OSSAD project and describe the research setting. Then, we describe the objectives and process of the OSSAD project as well as the OSSAD methodology and analyze them in terms of MEMES. At the end we collect findings of and lessons from the retrospective analysis.

12.2.1 Research Setting

The OSSAD project (1985-1989) was the ESPRIT Project launched with the aim to develop a methodology[176] for **O**ffice **S**upport **S**ystems **A**nalysis and **D**esign. It was funded by the European Union and companies, which participated in the project. The project was implemented as cooperation between academic and industrial partners from four countries, Finland, France, Germany, and Italy. The project engineered the comprehensive methodology that was published in a manual (Conrath *et al.* 1989) and in articles (e.g. Beslmuller *et al.* 1986; Beslmuller *et al.* 1987; Conrath *et al.* 1988; Charbonnel *et al.* 1991; Conrath *et al.* 1992; Vincent *et al.* 1992; Conrath *et al.* 1999; Savolainen 1999). This author's role as a researcher (1986 – 1989) in the project was to comment on and ideate the conceptual foundation of the methodology, to contribute to some specific parts of the methodology, as well as to field test the methodology in a Finnish organization (Leppänen *et al.* 1988; Leppänen *et al.* 1989a).

The project is relevant from the viewpoint of this study for the following reasons. First, it engineered a comprehensive domain-specific method. Second, the engineering was not "just another academic exercise" but it was firmly linked to practice. An indication of this is that the methodology in its various

176 A methodology means "a set of tools and procedures from which one can abstract a sub-set for particular goals and environmental conditions of a project" (Dumas *et al.* 1986, 3).

versions was field tested several times in practice. Third, the engineering occurred at the time when there was available no such body of knowledge of method engineering as we have at present. That gives us the possibility to analyze the context and its outcomes as "pre-methodical" ME instances.

As said above, we apply the retrospective analysis (cf. Fitzgerald 1991) to make sense of the process and outcomes of the OSSAD project. We set up two goals for the analysis. First, we try, using MEMES as the frame, to find out essential features and approaches of the OSSAD project, as well as to discover possible problems and speculate how these could have been avoided with methodical support such as that offered by MEMES. Second, our aim is to investigate how MEMES performs as a frame in this kind of analysis. Assessments of the OSSAD projects are not intended as criticism against the project or its members. It has to be remembered that at the time when the project was seen through the ME field was in its infancy.

Conducting the retrospective analysis in a proper way would necessitate keeping a record or log of the process in the context. This is so because it is often difficult to remember details at the end, which may be several years later (Fitzgerald 1991). The OSSAD project produced no material solely for this kind of research purpose. However, there is a large collection of documents, including the manual of the methodology, field test reports, memos, working papers, etc. that are applicable for the analysis. This material is not sufficient for a comprehensive and in-depth analysis of the process and outcomes, but it suits for our purposes.

12.2.2 OSSAD Process

The primary objective of the OSSAD project[177] was "to develop, implement, and validate a methodology (a set of methods) that can be used to design effective and acceptable computer-based office systems" (Baron *et al.* 1989, 2). This primary objective was decomposed into four sub-goals concerning "the description of an office work and information management systems, the creation, application and evaluation of measures of performance, the development of a model to describe and explain office work, information management systems, etc., and the development, application and evaluation of the means to evaluate the performance of various office support systems" (Baron *et al.* 1989, 2-3).

The ME work in the OSSAD project was divided into four ME stages. In the first of these stages a more detailed plan and schedule for ME work were made, the project organization composed of several working groups was established, and the approaches, fundamental principles and structure of the methodology were delineated. Four research fields were explored: taxonomies of office activities, a language for describing the office, a model for representing office work and organization, and performance measurements of office

[177] Esprit Project No. 285, R & D Area 4.1: Office Systems Science and Human Factors. Esprit = European Strategic Program for Research in Information Technology.

activities (Dumas *et al.* 1986, 5). The work in these fields contributed to e.g. concepts and notations of office models. In the OSSAD approach, the notion of a methodology was defined to mean "a set of tools and procedures from which one can abstract a sub-set suited for particular goals and environmental conditions of a project. This subset is called a 'method'" (Dumas *et al.* 1986, 3). A methodology was drafted around four dimensions (Dumas *et al.* 1986, 6): (a) "subdivision of the project into steps (What has to be done and how to do it?)", (b) "organizing the project (Who will accomplish the tasks?)", (c) "selection of the set of instruments out of the tool box (Which tools are to be used for each step and how?)", and (d) "documentation for project work and management".

In the second stage of the OSSAD project the two first phases (i.e. 'Contracting Phase' and 'Analysis Phase') in the OSSAD methodology were, on a general level, engineered. About the rest of the phases it was only mentioned that "they deal with the design of alternatives, choice, implementation, evaluation and the like" (Dumas *et al.* 1986, 7). The OSSAD methodology for the two phases was field tested in three countries (France, German and Italy). The common application area for field tests was savings banks. Experience from the field tests (Dumas *et al.* 1986; Beslmuller *et al.* 1987) were analyzed and utilized to refine the methodology. In the third stage the OSSAD methodology was enhanced with three more phases, which were 'Design System', 'Implement Changes', and 'Monitor System Performance'. Also in this stage, the OSSAD methodology was field tested, now in four countries (Finland, France, Germany and Italy) (Baron *et al.* 1989).

The purpose of the fourth stage in the OSSAD project was to refine, elaborate and complete the methodical support to the analysis and design of office support systems, covering for the whole life cycle. Work contained a very laborious process of restructuring, streamlining, detailing, and verifying descriptions of functions, activities, operations and procedures of the methodology. In this stage the decision was made to convert the descriptions of the OSSAD methodology to follow the terminology of the OSSAD methodology itself. The manual was finalized in 1989 (Conrath *et al.* 1989).

Next, we describe the process of the OSSAD project in terms of ME workflows of MEMES. First, we make it for the part which corresponds to the ISDM requirements engineering workflow.

In the OSSAD project the ME context as well as the target contexts were characterized, but not with any specific contingency framework. The ME context at hand was outlined at the beginning and later elaborated during the process. By its basic nature, the project was a research project, which was set up to fulfill concrete objectives. Despite its "academic flavor", the project was - through its participants, ways of working, and contributions - closely and firmly related to ISD practice. This premise "colored" all considerations and solutions in the project. The project applied primarily the policy-driven approach (cf. Section 11.7.2), as it aimed to provide a new methodology for the development of office support systems. That 'new' was concretized to mean the specification and deployment of more office-specific concepts and structures, as compared to the other methods of that time, as well as to provide more possibilities for users to participate in the development.

No prior contexts on the ME layer, nor on the ISD layer, were analyzed explicitly or separately. Instead, the participants brought with them their knowledge about and experience from the past contexts of method engineering and deployment they had participated in. The key participants in the OSSAD project were distinguished scientists or practitioners with long experience from the consultancy and software industry. The OSSAD project characterized the target contexts such as the one below:

> While early systems were usually developed for a single purpose, current technologies (e.g. the micro-computer and local area networks) support a variety of tasks, and the trend is to increase the level of integration across not only tasks but people as well. The consequence is that the nature of office work is changing and will continue to change, and to accomplish this work effectively an office […] depends on the appropriate integration of the organizational and technical support systems (Conrath *et al.* 1989, 1)

Requirements for the OSSAD methodology were specified throughout the ME project. The first requirements were very general, concerning generic approaches and principles. Later, they were detailed and made more structured. An example of the generic requirements is: "The resulting need is to ensure that integration of the organizational and technical support system is done in a well-planned, comprehensive and beneficial fashion" (Conrath *et al.* 1989, 1)

Existing ISD methods were not analyzed in an explicit fashion. Participants had good knowledge about the application area (i.e. office systems) and its evolution, as well as about methods available at that time (e.g. AXIAL, MERISE, SADT, X-TOP, OFFIS, OAM, MOBILE-Burotique, etc.). There were also some comparisons of relevant methods available in the literature (e.g. Newman 1980; Bracchi *et al.* 1984). Nevertheless, an in-depth analysis of existing methods would have given a more solid basis for the ME work. The knowledge was summed up with statements such as: "most of the existing system design methods focus on technical aspects of the office systems and pay virtually no attention to problems of organizational structure", "few approaches purport to be comprehensive and detailed enough", and "methods ignore organizational change issues" (Conrath *et al.* 1989, 1).

Based on the perceived problems in ISD practice and shortcomings in the methods available, several goals for the OSSAD methodology were stated in terms of ISD approaches, coverage and ISD actors. Examples of the goals are: the OSSAD methodology should be "concerned with the entire process – from the initial contact with someone interested in changing the existing support systems, to their design, implementation and ex post evaluation". It should be "developed on the assumption that those who use it are not experts in the design and implementation of integrated organizational /technical –support systems". "The target audience […] is expected to be composed of those who work in the areas of information systems, office automation, organization and methods, organisational design and like. (Conrath *et al.* 1989, 1). Furthermore, in the early stage of the project a set of main ISD principles were defined for the OSSAD methodology (Dumas *et al.* 1986, 6). The principles were: contingency, problem-orientation, participation, iterativeness, and experimentation. Later, the set was enhanced with the principle of decomposition/aggregation (Conrath *et al.* 1989, 2).

Next, we describe and analyze the process of the OSSAD project in terms of the ISDM analysis workflow. According to MEMES the ISDM analysis workflow is composed of three main tasks: infological ISD modeling, conceptual ISD modeling and inter-perspective ISD modeling. There are also three approaches (i.e. the functional approach, the conceptual approach, and the mixed approach) which affect how and in which order these tasks are carried out.

The OSSAD project applied the mixed approach to the ISDM analysis. It deployed an information flow view to delineate the functional structure of the analysis and design process of an office support system, and at the same time it made an attempt to establish a new kind of view of an office and office support system. For infological ISD modeling the process was decomposed into phases, later known as functions, and further into activities, operations, etc. Likewise, the notions of a packet and a resource were defined and connected to functional notions (cf. the information flow view). For conceptual ISD modeling a special work stream was established to define the basic notions of office and office support system. It followed the IS-driven approach (cf. Section 11.8.3), resulting in that also an office and an office support system were modeled in terms of information flows. We return to discuss the feasibility of this view later. From the tasks of conceptual ISD modeling (cf. Section 11.8.3) the OSSAD project addressed infological IS ontology engineering, datalogical IS ontology engineering, and physical IS ontology engineering but ignored systelogical IS ontology engineering and conceptual IS ontology engineering. This concretely shows how function oriented view the project had adopted. No special activities for inter-perspective ISD modeling were carried out.

Finally, we describe how the OSSAD methodology was evaluated (cf. the ISDM evaluation workflow in MEMES). Conceptual evaluation was organized in the way in which various versions and parts of the methodology were commented in other groups. By this way most of the deficiencies and inconsistencies in structures, naming and notations were revealed and improved. The decision to apply the terminology and notation of the methodology in describing the OSSAD methodology itself enabled easier and more transparent internal verification. The methodology was also empirically tested several times in practice (Dumas *et al.* 1986; Leppänen *et al.* 1988; Baron *et al.* 1989). Experience from field tests was utilized to make improvements in the methodology.

The OSSAD project carried out also engineering actions, which are not covered in MEMES. For instance, the OSSAD methodology provides some concepts and constructs for the ISD actor domain, such as a role, a unit and an actor. It also enables the modeling of physical and technological support for office work. These issues pertain to the ISD datalogical perspective and the ISD physical perspective, which belong to the ISDM design workflow and the ISDM implementation workflow, respectively. Because they are not covered in MEMES, we ignore them in our consideration.

12.2.3 OSSAD Methodology

In this section we briefly describe the OSSAD methodology and use MEMES as the frame to analyze it. Of numerous features of the methodology we concentrate on the main principles, the phase structure and the office models[178]. The descriptions are based on the OSSAD manual (Conrath *et al.* 1989).

The OSSAD methodology has been built upon a number of fundamental premises and beliefs, known as the OSSAD principles. The following is a summary of these principles: (1) Contingency: It is not realistic to tackle the world of office systems with one unique method. That is why OSSAD proposes a methodical framework, which allows the tailoring of the methodology to a specific project situation. (2) Decomposition/Aggregation: One has to able to examine a system at various levels of detail. (3) Experimentation: No method is likely to yield an ideal solution without any experience in its use. (4) Iteration: The use of feedback during the analysis, design and implementation of a system implies that the procedure is iterative. (5) Participation: The users are invited to analyze the existing situation and to model and suggest effective alternatives.

The methodology is functionally divided into five functions (originally called phases): Define Project, Analyze Situation, Design System, Implement Changes, and Monitor System Performance. Define Project involves establishing the basis upon which a particular study will be undertaken. It results in a contract for the re-organization, outlining the terms of reference and plans for the development. Analyze Situation concerns the collection, processing and presentation of the data needed to describe the organization and its environment for diagnostic purposes. The purpose is to identify the problems which should be resolved, and the opportunities which could be discovered by the introduction of a new and/or revised office support system. In Design System, one searches for alternative organizational and technical systems that will effectively respond to the identified problems and improve performance in general. Implement Changes convert things, which to this point have been conceptual, into something that is concrete, into an operational system. It consists of acquiring hardware and software, making and executing plans for reorganizing, educating and training, and making the system operational. Monitor System Performance is devoted to the analysis of an implemented office support system to identify whether or not it functions as

178 We have also made an in-depth analysis of the abstraction structures included in the OSSAD methodology (Leppänen 1989a). This is not considered here.

intended. It is composed of developing instruments and procedures, collecting and analyzing data, and devising recommendations for modifications.

The methodology provides three basic models for describing office and office support systems: Abstract Model, Descriptive Model, and Specification Model. The Abstract Model is composed of four basic concepts: function, sub-function, activity, and packet. A packet stands for an informational or material object in our terminology. The relationships between the actions (i.e. functions, sub-functions, and activities) and the packets are input/output/visited relationships. In addition, there are the ascendant/descendant relationships (cf. the partOf relationship) between the actions and between the packets. As a conclusion from the above we can say that the Abstract Model is a pure instance of applying the IS infological perspective.

In the Descriptive Model the functions, sub-functions and activities are decomposed into operations, tasks, and procedures. Tasks can be assembled to establish roles and (organizational) units. Corresponding to the concept of a packet in the Abstract Model, the concept of a resource is used in the Descriptive Model to mean "data or objects which are inputs to, or outputs from, operations / tasks / procedures / roles / units" (Conrath *et al.* 1989, 11). A facility is a physical and/or technological support used to perform work. In addition, the concept of an actor is used to mean "an individual who fulfills a role and/or who possesses the capabilities, such as education and experience, to fulfill a role" (Conrath *et al.* 1989, 11). To conclude, the Descriptive Model mainly highlights the features of office support systems from the IS datalogical perspective.

The third basic model, the Specification Model, corresponds mainly to the IS physical perspective. The OSSAD methodology distinguishes between two systems: the organizational support system (cf. HIS) and the technical support system (cf. CIS) (Conrath *et al.* 1989, 7). Both of them require a specification model. The technical specification model is intended to serve as the basis for hardware and software acquisition. More specifically, the technical specifications encompass the following components: user interfaces, software, databases and knowledge bases, hardware, system interconnections, and quality and control features. The organizational specifications are broken down into positions/roles, communication linkages, and decision making systems (incl. job descriptions and organization charts).

At a rather late stage in the OSSAD project, the decision was made to convert the descriptions of the OSSAD methodology itself to apply the terminology of two OSSAD models: each function in the OSSAD methodology is, therefore, presented in an abstract model and in a descriptive model. These two models are deployed to differentiate between situation-independent features and situation-dependent features (cf. the contingency principle). The abstract model of the OSSAD methodology applies to all applications of the OSSAD methodology, whether involving a large or a small organizational unit (Conrath *et al.* 1989, 3). It provides the essentials of office support system analysis and design. A descriptive model of the OSSAD methodology describes how in a certain kind of situation the analysis and design should be accomplished. Because the set of situations is large, the manual provides description models only for one or two situations in each phase.

12.2.4 Findings and Lessons

In this section we collect findings of and lessons from the retrospective analysis of the OSSAD project. First, we describe problems in the OSSAD process and methodology, identify ME approaches applied in the OSSAD project, and characterize the scope, emphasis and nature of the OSSAD methodology. Second, we present assessments of how MEMES performed as the frame in the analysis.

The OSSAD project was a comprehensive ME effort covering, to a large extent, all five ME workflows recognized in the ME ontology. Some tasks of the ME workflows were, however, insufficiently addressed. Methods and models of that time, for instance, should have been more carefully analyzed. With respect to conceptual ISD modeling, the project ignored tasks of systelogical IS ontology engineering as well as tasks of conceptual IS ontology engineering. Ignoring the IS conceptual perspective resulted in that the rich variety of office documents and communication and the semantics of office work were not considered. This led to a rather narrow domain of office models (see below).

ISDM analysis started with balancing the functional approach and the conceptual approach and later proceeded with the emphasis on establishing and refining the functional features of the methodology (cf. ISD infological perspective). In ISD conceptual modeling the project clearly applied the IS-driven approach. The OSSAD methodology was the target of several kinds of evaluation. First and foremost, field tests in four countries provided a noteworthy proof of the applicability of the methodology in practice.

The OSSAD methodology is quite extensive covering ISD phases from contracting to implementation. The methodology is structured according to two major constructs: the phase structure and the 'perspective-based' office modeling structure. Functionally office support system analysis and design is decomposed into five functions, corresponding to our phase structure. Three models (i.e. Abstract Model, Descriptive Model, and Specification Model) provide concepts and constructs for viewing from three different IS perspectives. The Abstract Model reflects the essence of office work from the IS infological perspective. The Descriptive Model applies the IS datalogical perspective, and the Specification Model provides concepts and constructs for the IS physical perspective. In the OSSAD methodology the first three phases and the first two models are described in more detail. This is justified by having several existing methods that can be used to fulfill the possible gaps in the support of later phases. Through the scope and contents of the Specification Model the methodology becomes more oriented towards the development of a technical office system than towards the design of human and social aspects of an office support system. This is actually contrary to the goals stated.

The OSSAD project was running at the time when several office models had been published in the field of office information systems. The best-known office models were SCOOP (Zisman 1977), ICN (Ellis 1979), Form Flow Model (Ladd *et al.* 1980), OFFIS (Konsynski *et al.* 1982), OMEGA (Barber 1983), TAM (Sasso 1984), and SOS (Bracchi *et al.* 1984). Common to all of them is that they view an office in terms of information flows (Auramäki *et al.* 1992a), whether as a set of interconnected operations, activities, or tasks. In the models of the OSSAD methodology the same orientation can be clearly seen. In this sense the methodology can be classified as having a functionalist approach to an office (Hirschheim 1986). Implied from the above, we can say that as regards to the concepts and constructs, the OSSAD Methodology did not bring much that was essentially new. During the OSSAD project human and social aspects were emphasized and communication was seen as vital. Unfortunately, the OSSAD methodology fails to adequately cope with these aspects (cf. the SAMPO model (Auramäki *et al.* 1988; Auramäki *et al.* 1992a)). What was new in the methodology is how it categorizes the features into three models, each of which reflects a particular view on the office and office work. Also its techniques for data collection and analysis as well as explicit guidelines for defining performance measures are innovative (cf. Auramäki *et al.* 1992b). It lacks a model with which the conceptual contents of office documents could be modeled. That means that the methodology applies a kind of document-based approach to office modeling (cf. Ladd *et al.* 1980; Zloof 1981; Ellis *et al.* 1982) in which documents are treated as data objects without explicit knowledge of their contents. The terms used to model an office, and in particular informational objects (i.e. 'packet' and 'resource') are not very illustrative, neither "office-like". Clearly, more effort would have been required to come up with better "essentials" of the office and office work. Any simple categorization of office document types would have been welcome, something similar to genres perhaps (e.g. Yates *et al.* 1992; Orlikowski *et al.* 1994).

Then, how did MEMES perform in the analysis? It provided conceptual constructs and "building blocks" which helped us structure the conceptions about the OSSAD process and the OSSAD methodology. We were able, for instance, to identify the basic ME action structures and to point out limitations in the ME life cycle of the OSSAD project (cf. the analysis of existing methods, tasks of systelogical IS ontology engineering, and tasks of conceptual IS ontology engineering). Second, we could recognize ME approaches followed in the OSSAD project (e.g. the functional approach and the IS-driven approach), based on the ISD perspectives and the IS perspectives adopted, which help us understand the rationale of ME actions. We found ISD constructs and IS constructs embedded in MEMES helpful in analyzing the OSSAD methodology. The collection of the main OSSAD models (i.e. Abstract Model, Descriptive Model, and Specification Model), for instance, was found to follow closely three IS perspectives respectively. All the models were recognized to reflect the view of information flows on an office and an office work.

12.3 Evaluation through the MEMES Effort

In this section we examine how MEMES performed as a prescriptive artifact in the engineering of MEMES itself (cf. the constructive intension). First, we explain why the MEMES effort is interesting from the research viewpoint, and what principles, resulting from the use of the reflection-in-action approach, shaped the MEMES process. Second, we describe the MEMES process in more detail. Third, we present findings of and lessons from the MEMES effort.

12.3.1 Research Setting

The MEMES effort is interesting from the research viewpoint for several reasons. First, evaluative studies of method engineering in practice are fairly rare. Second, as far as we know there are no studies that would address method engineering for method engineering. Third, two parallel processes, induced by the application of the reflection-in-action approach, offer an interesting research domain to make sense of, structure and evaluate. In the following, we describe and evaluate the MEMES effort by means of retrospective analysis.

The MEMES effort was carried out applying the reflection-in-action approach (Schön 1983). The approach makes the distinction between two theories of action (Argyris *et al.* 1978): espoused theories, which are those that an individual claims to follow, and theories-in-use that are those that can be inferred from action. The latter are tacit, cognitive maps by which actions are designed. They can be made explicit by reflecting in action (Heiskanen 1995, 7).

In our case there was only one person, who acted in two roles, as a practitioner and as a researcher. The practitioner was the method engineer and the user of MEMES. The researcher was the one who, based on his reflections, formulated and elaborated prescriptions for method engineering. The practitioner sought to discover the paricular features of a problematic situation, and from the gradual disovery, framed the situation and tried to fomulate a solution. An essential ingredient of situation framing in action design is the notion of a generative metaphor (Heiskanen 1995, 8). This is a vehicle for seeing the phenomena under study as something. In the MEMES effort our metaphor was based on the assumed analogy between ISD contexts and ME contexts, and between ME contexts and RW contexts. In our view, the ME context possesses typical features of the ISD context (e.g. Olle *et al.* 1983; Kumar *et al.* 1992; Tolvanen 1998), and correspondingly the RW context resembles the ME context.

Reflection is the practice of periodically stepping back to ponder one's immediate environment (cf. Raelin 2001, 11). We can categorize kinds of reflection according to (a) the target of reflection, (b) the timing of reflection, and (c) the primary goals of reflection (cf. Raelin 2001; Heiskanen 2005, Baskerville *et al.* 1998). Based on the target, Raelin (2001) and Heiskanen (2005) distinguish between content reflection, process reflection, and premise reflection. Content reflection is about how a practical problem was solved. Process reflection studies the procedures and the sequence of the events. Premise reflection questions the presuppositions underlying the problem. Based on the timing of reflection, Heiskanen (2005) recognizes anticipatory, contemporaneous, and retrospective reflection. Furthermore, we can distinguish between three kinds of goals of reflection: organizational development, system design, and scientific knowledge (Baskerville *et al.* 1998, 95).

Reflection in the MEMES effort concerned contents, process, and premises, with the aim to contribute to method engineering and scientific knowledge. It appeared as contemporaneous reflection in the form of a fluid process structure. A fluid structure "defines activities very loosely, allowing substantial simultaneity or leaving the temporal location of various activities relatively undefined" (Baskerville *et al.* 1998, 95).

Figure 132 illustrates, in a more concrete way, the two processes in the MEMES effort. It expresses how the RW process and the reflection process make up an iterative process. It also shows the role of MEMES in this process. The RW process means actions of engineering the ME skeleton for the construction of an ISD method. The reflection process means actions of learning from the aforementioned RW process and engineering ME guidelines for the construction of the ME methodical skeleton. The practitioner starts the RW process by considering a particular ME problem (e.g. how to decompose the ME analysis workflow into ME tasks). He tries to make sense of the problem space by using his intuition and the selected generative metaphor according to which the ME can be conceived as the ISD context. He produces, for instance, a tentative sub-division of the ME analysis workflow into ME tasks to be included in the body of MEMES (for ME). Next, he changes his role into a researcher and starts the process of reflection on what he just did, with the

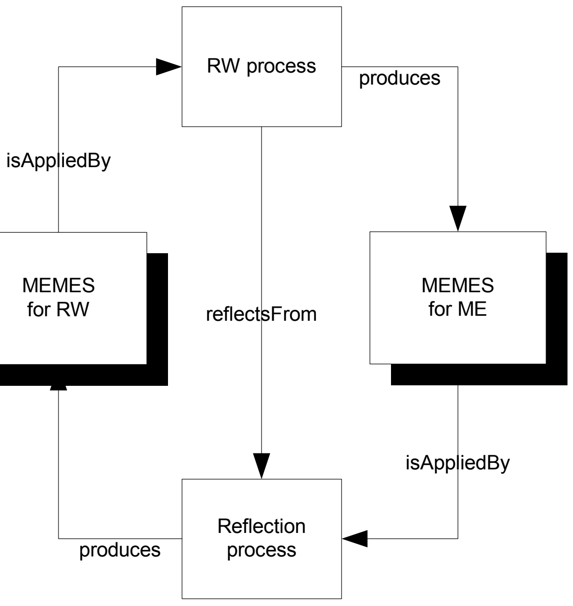


FIGURE 132 Iteration between the RW process and the reflection process

aim to formulate guidelines for how the corresponding problems should be solved in a general case. Using the generative metaphor he tries to reconstruct and restructure the past RW process by engineering the corresponding part of MEMES (for RW). After this, the researcher returns into the role of a practitioner, and repeats the RW process, now with the reconstructed guidelines (MEMES for RW) in order to re-engineer MEMES for ME. This may be followed by still another reflection process with the help of the re-engineered MEMES.

In iteration within and between the RW process and the reflection process MEMES acts in three roles. First, MEMES is an outcome of the RW process. Second, MEMES is applied as the frame through which the RW process is reflected. Third, MEMES acts as a prescription produced by the reflection process and applied in the RW process.

The reflection-in-action approach was implemented in the form of self-reflection, not as collaboration between researchers and pracitioners. Thus, our way of conducting the research clearly differed, for instance, from the one named as 'reflective IS action research' in Baskerville *et al.* (1998, 110). Due to this fact, the findings and lessons reported on the MEMES effort are mainly subjective and only of moderate significance from the viewpoint of validation. We also acknowledge the danger of post-rationalization and one-sidedness (Heiskanen 2005, 8) when making interpretations about problem settings, decisions, and events in the MEMES effort. However, we believe that including the retrospective analysis of the MEMES effort in the thesis increases our understanding of an ME effort in general, and of how MEMES performed as the prescription in the MEMES effort in particular.

12.3.2 MEMES Process

In this section we describe, in more detail, the RW process and how it was related to the reflection process. In particular, we elaborate the role of MEMES in the process. In the description of the RW process we refer to an "engineering space" (Figure 133), which is composed of two dimensions, one for the RW workflows (RW RE, RW analysis, RW evaluation) and the other for the ME workflows (ISDM RE, ISDM analysis, ISDM evaluation). The former stands for the RW process and the latter corresponds to the ME process covered by those parts of MEMES that were engineered in the RW process. RW RE, for instance, means requirements engineering in the RW context, in particular for MEMES. ISDM RE, in turn, means requirements engineering for an ISD method. The analysis workflow in both of the dimensions is sub-divided into two parts standing for the infological modeling and the conceptual modeling, respectively. Because at the first stages of the RW process MEMES was not yet viewed as a composition of the aforementioned parts, the topmost row in Figure 133 corresponds to a general view of MEMES. The numbers between 1 and 20 and the arrows connecting them show how the RW process progressed and iterated.

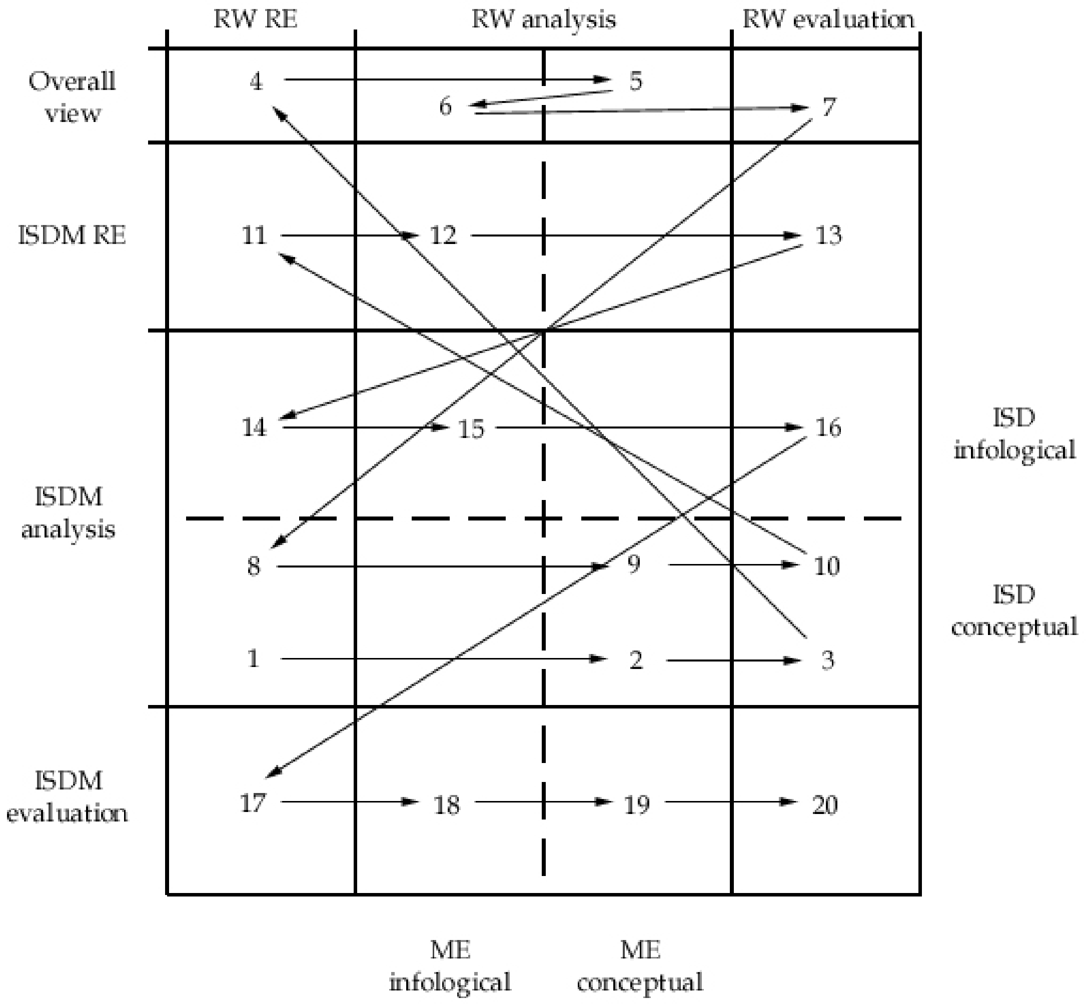


FIGURE 133 A detailed description of the cyclic ME process

The RW process started with considering the requirements engineering for an ISD method from the ISD conceptual perspective (1). The ISD conceptual perspective was applied to obtain a rationale and basis for engineering the core ontology and the context ontology. This was followed by RW analysis (2) and RW evaluation (3) from the same perspective. RW evaluation means checking the internal consistency and coherence of the ontologies, comparing them with existing artifacts in the literature and applying the engineered parts of the core ontology in the analysis of the OSSAD methodology (Leppänen 1989a). These tasks constituted a functional totality within which several iteration cycles were carried out. We refer to this totality as the first RW stage[179]. This stage began in small-scale in the 1980's and strengthened in the 1990's.

The second RW stage was triggered in 2000 when the goal to develop a more normative support for ME was set up. In this stage the historical view, the

[179] For the MEMES effort no phase structure with milestones and baselines were defined. Therefore, we refer to these functional wholes as the RW stages.

generic view and the application view of MEMES was outlined (4). The work done so far in the conceptual IS modeling was reconsidered and plans to enhance it into the ME ontology was made (5). The functional structure of MEMES was also sketched (6). All these results were evaluated in terms of their consistency and coherence (7). The process in this stage contained several iterations. Implied from the above, we can say that our ME approach to this end was clearly conceptual.

In the third stage the RW work moved to enhance the ME ontology. So far only some parts of the core ontology and the context ontology were established. We specified more concrete goals for the missing parts of OntoFrame (8) and started the engineering work to fulfill these goals (9). Engineering the contextual ontologies, the ISD ontology, the ISD method ontology, the ME ontology and the ME method ontology was executed in a highly iterative process including repetitive evaluations (10). During that stage several changes and enlargements were also made in the core ontology and the context ontology. Also the view of methodical support provided by MEMES was clarified and refined.

In the fourth stage we concentrated on refining the functional structure of MEMES based on the ME workflow structure delineated in the second stage. We started with elaborating goals for the ISDM requirements engineering workflow and decomposed the workflow into ME tasks (11-13). In the same way we carried out RW actions for the two other ME workflows, the ISDM analysis (14-16) and ISDM evaluation (17-20). Here we specified, among other things, ME tasks for engineering an ISD ontology and an IS ontology (cf. conceptual ISD modeling in Section 11.8.3). The process was highly iterative including evaluation in terms of internal and external criteria.

After outlining the RW process of engineering MEMES above, we will next describe in which stage and how we could deploy MEMES as a methodical support in this RW process.

The first stage was carried out without MEMES. The process was guided by general knowledge about conceptual modeling (in the 1980's) and metamodeling (in the 1990's), collected from the literature and experienced from practice. In the second stage we made a conscious decision to start applying the reflective approach according to which the problem space and the solution space of the RW process were framed with those parts of MEMES which were already available. This decision was made for two reasons. First, this way we could, with only short delays, obtain individual and concrete knowledge about how parts of MEMES just sketched functioned. Second, the RW process was highly complicated and difficult to manage and MEMES, although a half-ready artifact, could substantially benefit the process.

In the third stage we could not utilize MEMES because it did not yet contain normative prescriptions for ontology engineering. However, major decisions on essential structures of MEMES made in the preceding stage framed the process and its outcomes in this stage. Engineering of the topmost ontologies in OntoFrame shaped some conventions, which we later wrote into prescriptions of how to make conceptual ISD modeling. In the fourth stage we could fully benefit from MEMES. For instance, in specifying tasks for the ME RE workflow (Section 11.7), we reconsidered the specifications made for RW requirements and goals. Likewise, in specifying guidelines for infological ISD modeling (Section 11.8.2) we applied the same guidelines as in modeling ME from the infological perspective.

12.3.3 Findings and Lessons

Our purpose in this section is to describe the MEMES effort in terms of MEMES and uncover the approaches, action structures and motives for the RW actions performed. We also assess the applicability of MEMES in the MEMES effort.

We did not use any special contingency framework in the RW requirements engineering. Instead, we applied OntoFrame whenever it seemed to be applicable. When analyzing the RW context at hand, we recognized it possessing features of both the problem-driven approach and the policy-driven approach. Our experience from the four prior ME efforts had convinced us that ME in practice is too often 'engineering by trials' without any systematic way of thinking and working. Experiences reported from other ME projects (e.g. Vidgen 2002; Polo *et al.* 2002; Serour *et al.* 2002; Fitzgerald *et al.* 2003; Backlund *et al.* 2003, Bajec *et al.* 2004) confirmed our view of the unsatisfying state of the art in ME. On the other hand, we had no detailed problems to be solved. Instead, we recognized the need to develop a new approach by which method engineering could be considered and managed in a more comprehensive and uniform manner, and to implement this approach into the form of generic methodical support.

We analyzed documents reporting on the backgrounds, processes, deliverables and experiences of our prior ME contexts (CSDM (Leppänen 1984a), OSSAD (e.g. Baron *et al.* 1989; Conrath *et al.* 1989), SPITS (Leppänen *et al.* 1991), DBSD (Leppänen 1993; Leppänen 2001)) in order to obtain an overview of how the contexts were accomplished and with which results. Because the material available from these contexts was not originally made for research purposes and it does not cover all the decisions and phases, we do not want to overemphasize its significance to this research. However, it appeared to be useful in summoning up thoughts emerged in the prior contexts. Short descriptions of these prior ME contexts are presented in Section 11.3.

We did not specify ISDM requirements in a separate ME task. Instead, we established an overall conception of the level of detail in which MEMES should be presented and of the application area (i.e. the nature of the target ME context and the ME strategy), which MEMES should support. These are reported in Sections 11.2 and 11.4, respectively.

We made a comprehensive analysis of ME approaches, meta models, ME techniques and ME procedures in the literature. In addition, we reviewed reports of single ME efforts in practice (e.g. Jaaksi 1997; Vidgen 2002; Polo *et al.* 2002; Serour *et al.* 2002; Fitzgerald *et al.* 2003; Backlund *et al.* 2003, Bajec *et al.* 2004). The purpose of these reviews was two-fold. On one hand, we wanted to learn what kind of support for ME there was already available. On the other hand, we were interested in needs and motivations for, processes of, and experience from, engineering ME methods, or parts thereof.

Based on the analysis of the prior contexts and the ME literature, we stated goals for our RW effort. In this task we had to take into consideration the scarce resources we had for the work and to set the goals at a reasonable level. The goals of MEMES are reported in Section 11.5.

In the RW analysis workflow we postponed the engineering of the "procedural" part of MEMES (Kumar *et al.* 1992), thus following the conceptual approach to the RW analysis (cf. Section 11.8.1). The reasons for this are evident. Although there were some conceptual frameworks for analyzing, comparing and assessing ISD methods, they were far from being suitable as an ISD ontology. Without having a profound understanding of what ought to be engineered, it is not possible to engineer actions (i.e. ME actions) of engineering.

The process of engineering OntoFrame followed the top-down approach (Uschold *et al.* 1996; Noy *et al.* 2001). Consequently, we started conceptual ISD modeling with building main parts of the core ontology. Then, we carried out a comprehensive search for theories addressing the notion of a context and specified the fundamental categorization of contextual domains containing specific contextual concepts and constructs. After that, we extended our engineering work to address the ISD sub-domain and the ME sub-domain.

The RW analysis workflow was carried out as a highly iterative process. At each level of detail we decomposed and inter-related ME actions and ME deliverables, always trying to ensure that ME models were properly grounded on OntoFrame. In the inter-perspective ME modeling we followed a variant of the conceptual approach (cf. Figure 120 in Section 11.8.4). We started with the ISD ontology (C x C) but selected the ME workflow structure from the ME ontology to establish a rough decomposition of ME work into ME workflows. After that we derived ME deliverables from the ISD constructs (D x C). In the later cycles of the RW process we made cross-checking to ensure the consistency of MEMES.

In the RW evaluation we applied the generic steps of the ISDM evaluation workflow (cf. Section 11.9). The basis, process and outcomes of this evaluation are reported in this chapter.

To sum up our experience from the RW process and its results, we can say that the generative metaphor in the first stages and MEMES in the later stages of the MEMES effort appeared to be viable and beneficial. Conceiving ME contexts as ISD contexts and correspondingly the RW context as an ME context enabled us to make our first choices and specifications of concepts and constructs of MEMES. Later, MEMES highlighted, in a structured fashion, those issues that should be analyzed and reframed. MEMES povided steps by which we could, for instance, split the workflow of RW requirements engineering into manageable and inter-related tasks and carry out them. By the help of MEMES we could also integrate our earlier work done for the core ontology and the context ontology into the methodical skeleton. MEMES helped us categorize relevant engineering issues on the basis of the ME perspectives and the ISD perspectives in a way which guided us in making decisions on the scope of MEMES (cf. the datalogical and physical perspectives on the ME and ISD layers were excluded) and helped us plan for the next ME steps and complete them. MEMES also offered a useful set of ME approaches from which we selected the mix of the policy-driven approach and the problem-driven approach to the RW requirements engineering and the conceptual approach to the RW analysis.

What could we have done better, or in another way? If the RW process was to start now, it would be better first to outline MEMES on a general level (see steps (4) – (7) in Figure 133). Based on this outline, it would be much easier to decide on the objectives and scopes of OntoFrame. Nevertheless, the next steps would be conducted to engineer OntoFrame, with the top-down approach as we did. Second, in engineering the first component ontologies of Ontoframe we should have been more conscious of what we are doing and how, so that the conventions used could have been devised into structured steps earlier than it occurred in the MEMES effort. This would have saved us from gratuitous iterations and groping. However, the resulting version of OntoFrame itself would hardly be different from the present as to its structure and contents.

How valid are the conceptions presented above? As mentioned in Section 12.3.1, in these kinds of subjective evaluations there is a risk of post-rationalization and one-sidedness. We have been conscious of that risk and tried to avoid it. Some of the conceptions are based on the facts. For instance, the detailed description of the MEMES process in Figure 133, used to illustrate how complicated, multifaceted and iterative the process was, has been made in a precise manner on the basis of written working plans, notices and memos. It is not a result of post-rationalization but a view of what really happened. OntoFrame and MEMES just povided useful means to portray how the process navigated from one array of engineering issues to the others. Assessments of the quality of the support MEMES provided for the MEMES effort are naturally subjective but to increase their credibility we have provided plenty of arguments.

12.4 Comparative Analysis of ME Artifacts

In this section we make a comparative analysis of those artifacts in the ME literature which are aimed to provide methodical support to ME efforts. We call them the ME artifacts. The analysis is divided into two parts. First, we examine the backgrounds, application areas, ME strategies and ME approaches of the ME artifacts. Second, we analyze the coverage and emphases of the ME artifacts in terms of ME workflows, perspectives and contextual domains. Our aim here is to find out how MEMES compares with the existing ME artifacts, and how useful MEMES is as a frame of reference in this kind of comparative analysis.

The ME literature is quite large. We can distinguish between three groups of ME artifacts there. The first group comprises a large variety of meta models and metamodeling languages developed and used to model methods, or parts thereof. This group contains two kinds of meta models, that are meta data models and meta process models. Meta data models are used to metamodel the conceptual contents and notations of data models. Examples of the meta data models and metamodeling languages are ER (Chen 1976), eERM (Rosemann *et al.* 2002, Scheer 1998), NIAM (Nijssen *et al.* 1989), OPRR (Smolander 1991), ASDM (Heym *et al.* 1992), CoCoA (Venable 1993), GOPRR (Kelly *et al.* 1996), Telos (Jarke *et al.* 1995), and MEL/MDM (Harmsen 1997)). More about meta data modeling languages and differences between them can be found in Venable (1993), Saeki *et al.* (1994), Harmsen *et al.* (1996) and Tolvanen (1998, 155). Meta process models and modeling languages have been developed for modeling SE/ISD process models. Meta process models and process modeling languages are presented in Bandinelli *et al.* (1993), Deiters *et al.* (1994), Christie (1993), Shepard *et al.* (1992), Dutton (1993), and Kaiser *et al.* (1993). Evaluations of and comparisons between process modeling languages are reported e.g. in Söderström *et al.* (2002).

The second group of the ME literature comprises ME strategies, ME approaches and ME techniques. In what follows we shortly discuss some of them. Kumar *et al.* (1992) propose a methodology for developing a situation-specific methodology. They distinguish between four ME strategies: modular construction, stakeholder-value based composition, the use of automated computer-based support, and a supporting organizational structure for ME. The proposal does not provide any concrete guidelines for courses of action in ME. van Slooten *et al.* (1993) present a framework and a procedure to configure development scenarios from project characterizations defined by project contingency factors. Oei (1995) suggests the MMT approach (the Meta Model Transformation approach) to relate meta models of languages in an open ordering and transformation scheme by means of a set of basic meta model transformations. The MMT approach can serve for comparison, integration, and evolution of modeling languages. Kinnunen *et al.* (1996) suggest an O/A (Object / Activity) matrix –based technique for describing and analyzing the interoperability of method components. van Slooten *et al.* (1996) provide a wide contingency model for ME and discuss its use in the choice of route map fragments and method fragments. Grundy *et al.* (1996) sketch an integrated method engineering approach based on the MViews framework. Saeki (1998) presents an approach to integrate multiple methods through the use of a meta model and a CASE tool to demonstrate that the approach is beneficial. Leppänen (2000) defines the concept of consistency from several perspectives and provides an ME technique and a set of ME approaches to ensure the conceptual consistency of an ISD method. Hruby (2000b) describes a process framework for the specification of development processes that considers management and software development artifacts as objects and evolution as collaborations between them. None of the aforementioned ME artifacts can be considered to come even close to the notion of an ME method. They either remain on a very general level (cf. ME strategies and ME approaches) or they cover only a small part of the ME process (cf. ME techniques).

The third group of the ME literature contains artifacts that aim to offer more comprehensive support for ME. No one in this group is, however, a complete ME method if weighed with the criteria given in Section 10.5. This group contains the dissertation works of Harmsen (1997) and Tolvanen (1998). Both of them suggest specific approaches to ME (a situational ME in Harmsen (1997) and an incremental ME in Tolvanen (1998)) and offer general-level procedures to implement the approaches. In addition, this group contains the ME artifacts of Gupta *et al.* (2001), Song (1997), Vlasblom *et al.* (1995), Nuseibeh *et al.* (1996) and Ralyte *et al.* (2003). Gupta *et al.* (2001) define a representation system for a method requirements specification and describe an automated process for instantiating a technical meta model. Song (1997) defines a framework for the integration of design methods and gives principles of applying it. Vlasblom *et al.* (1995) propose the three-level description of a method and present a "protocol" for the construction of a development model. Nuseibeh *et al.* (1996) outline a multi-perspective ME approach based on the notion of Viewpoint and describe, on a general level, a process of method design and construction. Ralyte *et al.* (2003) present a generic process model supporting the integration of different approaches to situational method engineering. Though some of the artifacts in this group are not described in detail (e.g. Nuseibeh *et al.* 1996; Ralyte *et al.* 2003), they are included here because of their special features of support for ME.

For our comparative analysis we have selected the ME artifacts in the third group. In the following, we first describe results from the overall analysis and then deepen the view with considerations of the coverage and emphases of the ME artifacts. In both of these parts we make comparisons to MEMES.

12.4.1 Overall Analysis

The purpose of the overall analysis is to disclose the backgrounds, application areas, ME strategies and ME approaches of the selected ME artifacts. These issues concern the historical view, the application view, and the generic view, respectively (see the ME method ontology in Section 10.5). To put it more precisely, the issues considered are:

- *Historical view*. What are the theoretical foundations and research methodologies used to engineer the ME artifact (cf. the prior RW contexts)? Has the ME artifact been applied and with which experience (cf. the prior ME contexts)? Which arguments are given to justify the applicability and validity of the ME artifact?
- *Application view*. For which kind of ME contexts is the ME artifact intended (cf. the target ME context)? The application area can be characterized with types of ISD methods (i.e. generic, domain-specific, organization-specific,

vs. project-specific) and ME strategies (i.e. creation, integration, adaptation).

- *Generic view*. Which kinds of ME strategies and ME approaches does the ME artifact apply to?

The summary of the results from the overall analysis is presented in Table 34. In what follows, we briefly describe the selected ME artifacts and analyze them in terms of the defined issues.

Harmsen (1997) presents (a) an ontology for products of information systems development, called the Methodology Data Model, (b) a method engineering language, called MEL, and (c) a process of situational method engineering with heuristics and formalized method assembly rules. MEL can be used to represent and administrate method fragments. The situational approach to method engineering is based on the principle of controlled flexibility according to which for each situation, whether a project or an organization, a specific method is built. The notion of a situational method corresponds to an organization-specific method or a project-specific method in our terminology.

The process of method engineering has been rooted on the theory of situational method engineering. The approach to develop the theory is said (ibid p. 16) to follow the so-called Lockean inquiry system (Churchman 1971), and three stages are distinguished in it: theory building, theory testing and theory expanding. At the first stage a number of methods were metamodeled and analysed to develop the ontology of the IS and MEL. At the second stage a CAME (Computer Aided Method Engineering) tool was designed and implemented to test the theory. At the third stage experiences got from tests resulted in corrections and enlargements into the ontology and MEL, as well as in establishing and formalizing method assembly rules. Some studies are also mentioned in which the theory has been empirically tested.

Harmsen (1997) advocates the contingency approach, which provides a set of factors to characterize fragments and IS engineering situations. Second, Harmsen (1997) applies the integration strategy in constructing a method from method fragments. Method configuration process is seen as a part of project management, meaning that a method evolves during the project.

Tolvanen (1998) presents a set of constructs of method modeling languages, defines guidelines and mechanisms for collecting and analyzing modeling-related experiences, and explains their implications for method improvements. Related to the latter, he brings out principles of incremental method engineering and studies method development through experience-based method refinement. The principles are aimed at supporting organizations to develop their own methods, known as local methods.

Tolvanen advocates a view according to which local method engineering is a learning process "in which experience of successful (or unsuccessful) ISD efforts needs to be incorporated into future ME efforts: every use situation of methods should evaluate and analyze methods with a view to improving them"

TABLE 34 Overall analysis of the ME artifacts

Reference	**RW context**	**ME context**	**ME strategies and ME approaches**
Harmsen (1997)	Lockean inquiry system: theory building – theory testing – theory expanding	A situational method is engineered based on organizational contingencies and the use of a method base	Situational method engineering by assembly of method fragments (cf. integration strategy)
Tolvanen (1998)	Iteration of conceptual and empirical research (action research) to produce metamodelling constructs and principles for incremental ME	An ISD method is incrementally constructed for a given project based on method specifications in a method base	Incremental approach to local method engineering Meta-data modelling approach
Gupta *et al.* (2001)	--	Method engineering for a specific project	Decision making approach to ISD Implementation-independent approach to ME
Song (1997)	Analysis of existing methods and practical experiments	Integration of useful ideas and notations from other methods	Method integration on two levels of detail
Vlasblom *et al.* (1995)	--	A project-specific method is engineered based on situation profiles	Flexibilization approach to ISD method Contingency approach
Nuseibeh *et al.* (1996)	Practical experiences	Method engineering in the context of multi-perspective software development	ViewPoint approach Integration strategy
Ralyte *et al.* (2003)	--	Situational engineering of a project-specific method	Situational method engineering Approach to support the integration of different ME strategies

(ibid p. 170-171). This experience-based learning is realized through an incremental process. Tolvanen divides the process into two kinds of steps: a priori steps and a posteriori steps. A priori ME steps are method selection, method construction and tool adaptation. A posteriori ME steps are collection of experiences, analysis of method use, and method refinement. His view on meta modelling and method engineering is limited to meta data models only.

Research in Tolvanen (1998) applied conceptual and empirical methods. Seventeen ISD methods were modeled and their meta data models were validated through implementation by a CASE tool. The method specifications were used to analyze method knowledge to extend languages for method modeling. Second, an action research strategy was followed in two case studies in which methods were developed and adapted to local needs. Experiences got from these cases were used to refine the principles and mechanisms for collection and analysis of experiences from ISD efforts (Tolvanen 1998, 29-30).

Gupta *et al.* (2001) define a representation system for a method requirements specification (MRS) and describe an automated process for instantiating a technical meta model with an MRS. This instantiation is used to produce the actual method, which is then given to a metaCASE tool to produce a CASE tool. Interesting in the ME approach is that a method is first specified on a ”relatively abstract” level with statements of method requirements, and then it is elaborated and instantiated. The approach is based on a belief that method engineers are experts of the domain of methods, but not necessarily experts of meta models and how to instantiate them. The approach applies the static and dynamic views on methods (Prakash 1997; Prakash 1999) reflecting the ISD as a decision making process.

Proposals contained in the paper have been used to construct the CAME tool part of a CASE shell, called MERU (Method Engineering Using Rules). Nothing is said about research work resulting in the proposals, neither about validation of results, although through MERU some proofs by implementation are provided.

Song (1997) defines a framework for the integration of design methods and gives general principles of applying it. The framework views integration mainly as an effort to enhance the existing method with properties or components of some other method(s). In the framework four kinds of integration are distinguished: model integration, principle integration, process integration, and representation integration. Further, two approaches to integration are recognized: function-driven integration and quality-driven integration. Song does not provide any method of integration, neither any uniform procedure. Instead, he gives an outline of steps for function-driven integration and quality-driven integration on two levels of detail. The framework and the steps have been derived from the analysis of existing methods and practical experiments. No validation has been made in a rigorous sense. It is mentioned that "feedback from these experiments has been positive; designers appreciate the common forum for generalizing, communicating and applying ideas” (ibid p. 108). No generic philosophy can be found underlying the ideas. The framework and the steps are suggested for situations in which it is seen beneficial "to borrow ideas and notations from other methods" (ibid p. 107).

Vlasblom *et al.* (1995) propose the three-level description of the method. The levels are the generic level, the model level, and the specific level. The generic level is composed of building blocks for various elements of the method. The model level contains development models for specific application domains. The lowest level corresponds to project-specific development methods. With these levels a more flexible method architecture and method engineering process are pursued. Vlasblom *et al.* (1995) present the so-called "seven-point protocol" to be followed in establishing a development method that "is optimally tailored to a project". The protocol is composed of seven questions, like "Which products are to be deliverables?" and "For which target group are they intended"? Vlasblom *et al.* (1995) also suggest steps of how to utilize a development model in a specific situation, as well as steps to construct a development model. Nothing is said about the research process, neither about validation. Several examples of development models with situation profiles, taken from practice, are provided to show "that the concepts described are of practical significance" (ibid p. 604).

Nuseibeh *et al.* (1996) outline an ME approach based on multi-perspective development. The key concept of the approach is Viewpoint that is "a loosely coupled, locally managed, distributable object that encapsulates representation knowledge, development process knowledge and specification knowledge about an ISD" (ibid. p. 268). An ISD method is a collection of method fragments, each of which describes how to develop a single ViewPoint specification. A Viewpoint is internally divided into five 'slots': style (notation), work plan (development process), specification (one described in the notation, produced by the development process), domain (label identifying the area of concern of the ViewpPoint), and work record (specification development status, history and rationale). A ViewPoint template contains the first two 'slots'. A method is a configuration of ViewPoint templates. This implies that an ISD process is not a sequential series of procedures but dynamically created as the development proceeds.

Nuseibeh *et al.* (1996, 270) outline a process of method design and construction. They also present considerations of method integration. Integration is seen useful in three types of cases (cf. Kronlöf 1993): (1) integrating common features of several methods, (2) extending a main method with some new features of other methods, and (3) restricting a main method by replacing or overriding some of its features. To advance the integration they advocate the use of pairwise inter-ViewPoint relationships or rules. The ViewPoint framework and the Viewer (CASE-tool) have been deployed in a number of case studies and research environments (ibid p. 272). A set of standard methods have been modelled. These experiments are said to provide feedback on improving the structure and organisation of the ViewPoint framework.

Ralyte *et al.* (2003) present a generic process model supporting the integration of different approaches to situational method engineering. The generic model contains three ME techniques: assembling method chunks, extending an existing method, and generating a method by abstraction/instantiation of a model/meta-model. The paper also shows how other ME techniques could be integrated in the generic model. Nothing is said about the theoretical basis underlying the work, or about the applied research methodology. Validation of the generic model is suggested to be part of the future project.

To summarize, only in three of the ME artifacts analyzed (Harmsen 1997; Tolvanen 1998; Song 1997) the research context is considered. Harmsen (1997) compares the research process to the Lockean inquiry system but does not describe the process in more detail. Tolvanen (1998) makes very clear what research methods have been used and how. Song (1997) mentions the practical background of his artifact. The absence of discussions about the research context may partly be due to the limited space of the articles but it is more probable that there have not been any profound theoretical grounds, nor the use of any rigorous research methods.

Among the analyzed artifacts the target ME context is most commonly a situation in which a project-specific method is configured (Tolvanen 1998; Gupta *et al.* 2001; Vlasblom *et al.* 1995; Ralyte *et al.* 2003). In Harmsen (1997) a constructed method is either an organization-specific method or a project-specific method. Principles presented in Song (1997) and Nuseibeh *et al.* (1996) apply also to other kinds of ME contexts. As regards with the ME strategies, the artifacts of Harmsen (1997), Song (1997) and Nuseibeh *et al.* (1996) are clearly based on integration. Tolvanen (1998) favors the adaptation strategy. Gupta *et al.* (2001) and Vlasblom *et al.* (1995) give no preference to ME strategies. Ralyte *et al.* (2003) recognize a number of ME strategies and pursue to integrate them with the generic model.

In this study we have clearly and firmly grounded MEMES on the conceptual and theory-based foundation, OntoFrame. We have also, in a comprehensive manner, brought out the research methodology (i.e. research process, research methods) by which MEMES has been produced (cf. Section 1.4). We have discussed the verification and validation of MEMES in Section 1.5 and reserved Chapter 12 solely for the evaluation of MEMES. MEMES is aimed at the engineering of a generic or domain-specific method. It can be used in conjunction with any ME strategy, although with some elaborations.

12.4.2 Coverage and Emphases of the ME Artifacts

The purpose of this section is to analyze the coverage and emphases of the selected ME artifacts in terms of ME workflows, perspectives and contextual domains.

In the analyzed ME artifacts the process of method engineering is decomposed in different terms: e.g. into phases (Gupta *et al.* 2001), steps (Harmsen 1997; Tolvanen 1998; Song 1997; Vlasblom *et al.* 1995; Nuseibeh *et al.*

1996), and strategies (cf. Ralyte *et al.* 2003). Because the notion of a phase has the connotation of a temporally ordered entity and ME actions, such as 'assembly of method fragments' (Harmsen 1997) and 'method construction" (Tolvanen 1998) are too large-scale to be considered as steps, we prefer to use the ME workflow structure in this analysis. Based on our ME ontology, the process of ME is composed of five workflows: ISDM requirements engineering (RE), ISDM analysis, ISDM design, ISDM implementation, and ISDM evaluation. The summary of the analysis in terms of these ME workflows is presented in Table 35.

To reveal which features in the ME context are emphasized in the selected ME artifacts, we deploy the perspectives and the contextual domains defined in OntoFrame. For the ME context we use the ME perspectives to categorize the support the ME artifacts offer to conceive, understand, structure and represent contextual phenomena of method engineering. Since method engineering also addresses, through the conceptual contents of the ISD method, the ISD context, we include the ISD context in the scope of our analysis as well. For the ISD context we apply the ISD perspectives and the ISD domains. The summary of the analysis of the coverage and emphases of the ME artifacts in terms of perspectives and contextual domains is presented in Table 36. We use the following abbreviations in the table: for the perspectives: S = systelogical, I = infological, C = conceptual, D = datalogical, P = physical, and for the domains: P = purpose, Ar = Actor, An = Action, O = object, F = facility, L = location, T = time. To distinguish whether the ME artifacts consider the ISD objects as representational deliverables (D) or conceptual constructs (C) we use the markings O/D and O/C, respectively. To express the emphasis the ME artifacts give to the contextual domains, we use the following markings: X = concerned to a large extent, x = concerned to a small extent, - = not concerned. Respectively, we use capital letters (e.g C) and lower letters (e.g. c) to indicate how extensively the perspectives are addressed in the artifacts. In the following we shortly describe the processes and deliverables of the selected ME artifacts and analyze them in terms of the defined issues.

In Harmsen (1997) the process of situational method engineering is decomposed into three main steps. The first step is the characterization of the situation, meaning that ISD project goals are determined and a preliminary scenario is generated and adapted with situational factors (or contingency factors), and possibly with performance indicators. The second step is the selection of method fragments, which is induced by the produced characterizations as the project scenario. Method fragments are characterized by a number of properties, many of which are directly related to scenario aspects. Using these relevant properties the method fragments supporting the project scenario can be selected. The third step is the assembly of method fragments in which the selected fragments are integrated to form a situational method. To avoid defects and inconsistencies in and between the fragments, a number of method assembly quality assurance rules are defined and applied. These rules concern completeness, consistency, efficiency, soundness, and applicability.

TABLE 35 Analysis of the ME artifacts in terms of ME workflows

Reference	**ISDM RE**	**ISDM Analysis**	**ISDM design**	**ISDM implementation**	**ISDM evaluation**
Harmsen (1997)	Characterization of situation	Selection of method fragments Assembly of method fragments	Selection of method fragments Assembly of method fragments	--	Assembly rules enforcement
Tolvanen (1998)	Analysis of ISD environment	Method selection Method construction Method refinement	--	Method construction Tool adaptation Method refinement	Evaluation of existing methods Collection of experiences Analysis of method use
Gupta *et al.* (2001)	Method requirements engineering (Method nature part)	Method requirements engineering (Simple method part & Mapping method part)	Method design	Method construction and implementation	--
Song (1997)	--	Property integration Principle integration Artifact integration Process integration	Process integration Representation integration	--	--
Vlasblom *et al.* (1995)	Analysis of a project situation (specific situation profile)	Modification of the best-matched development model	Modification of the best-matched development model	Modification of the best-matched development model	--
Nuseibeh *et al.* (1996)	--	Method design and construction	Method design and construction	Method design and construction	--
Ralyte *et al.* (2003)	Set method engineering goal	Construct a method	Construct a method	Construct a method	--

TABLE 36 ....Analysis of the ME artifacts in terms of perspectives and contextual domains

	ME								ISD								
Reference	**Perspectives**	**P**	**Ar**	**An**	**O**	**F**	**L**	**T**	**Perspectives**	**P**	**Ar**	**An**	**O/ D**	**O/ C**	**F**	**L**	**T**
MEMES	S & I & C	X	X	X	X	-	-	-	S & I & C	X	X	X	X	X	-	-	-
Harmsen (1997)	I & C & d	X	x	X	X	X	-	-	I & C	-	-	X	X	X	X	-	-
Tolvanen (1998)	I & C	-	-	X	X	x	-	-	C	-	-	-	-	X	-	-	-
Gupta *et al.*(2001)	i & C	-	-	x	X	-	-	-	I & C	X	-	X	X	X	-	-	-
Song (1997)	i & C	-	-	x	x	-	-	-	I & c	-	-	X	X	x	-	-	-
Vlasblom *et al.* (1995)	s & i	x	-	x	x	-	-	-		-	-	-	-	-	-	-	-
Nuseibeh *et al.* (1996)	i & c	-	-	x	x	-	-	-	I	-	-	X	X	-	-	-	-
Ralyte *et al.* (2003)	s & i	X	-	x	x	-	-	-		-	-	-	-	-	-	-	-

Legend:

Perspectives:
S = Systelogical
I = Infological
C = Conceptual
D = Datalogical

Contextual domains:
P = Purpose
Ar = Actor
An = Action
O = Object
O/D = Objects as representational deliverables
O/C = Objects as conceptual constructs
F = Facility
L = Location
T = Time

Emphasis:
X = concerned to a large extent
x = concerned to a small extent
- = not concerned

The steps in the situational ME process cover, at least to some degree, all the ME workflows except ISDM implementation (e.g. there are no rules for customization nor instantiation). The ME context is described from the ME infological and ME conceptual perspectives, and, to a smaller extent, from the ME datalogical perspective. Thus, ME purposes, ME actions (steps), ME deliverables (incl. method fragments) and ME facilities (CAME, Decamerone) are described. Some references to ME actors are also given. The conceptual contents of the ME deliverables are described from the ISD infological and ISD conceptual perspectives, including descriptions of ISD actions (process types), ISD deliverables (products) and ISD facilities (CASE tools). The conceptual contents of the ISD deliverables are also structured. Some descriptions of Decamerone are so technical that they belong to the physical ME perspective.

The process of incremental method engineering by Tolvanen (1998) is divided into two kinds of steps: a priori steps and a posteriori steps. The a priori ME steps are method selection, method construction, and tool adaptation. The a posteriori ME steps are collection of experiences, analysis of method use, and method refinement. In the first step the ISD environment is analyzed according to situation-independent and situation-dependent ISD method criteria. In the second step the selected methods are constructed. This means integration and adaptation of one or more methods, or parts thereof. In the third step the methods constructed are adapted into a CASE tool. This means customizing or building a tool for the method or selecting a set of tools which cover all the method knowledge (ibid p. 70).

In the a posteriori part of ME, experiences from the use of the constructed ISD method are first collected in terms of models produced during the ISD, meta models, reports on stakeholder interviews, etc. The experiences are analyzed according to the defined analysis mechanisms. Evaluation of method use can lead to modifications in the method and tool support. These modifications are done in the final step. The iterative nature of the ME process implies that the refined method can be taken as such or as a re-refined method in the next ISD project (Tolvanen 1998, 190-192).

A part of method selection, the analysis of ISD environment, belongs to the ISDM RE workflow. The rest of the method selection as well as method construction and method refinement correspond to the ISDM analysis workflow because the process concerns neither ISD actors nor ISD facilities. Part of method construction and tool adaptation belong to the ISDM implementation because there the method is particularly tied to technical infrastructure of the project. A part of the first step as well as the fourth and fifth steps address issues of the ISDM evaluation workflow. For these steps Tolvanen (1998) provides descriptions of ME actions. ME deliverables are, to a large degree, addressed from the ME conceptual perspective only. Because the incremental ME process has been established on the considerations of the meta data model (i.e. the GOPRR model (Kelly *et al.* 1996)), the only concerned domain in the ISD context is the ISD object domain, considered from the ISD conceptual perspective.

In Gupta *et al.* (2001) method engineering is composed of three main phases: (1) method requirements engineering (MRE), (2) method design (MD), and (3) method construction and implementation (MCI). In the MRE phase the method requirements definition is first produced. Method requirements are "high-level abstraction of services, that a method will provide, and constraints under which it functions" (ibid p. 136). They are a part of project characteristics and are obtained from the project context in which the method is to be applied. Second, the MRE yields a Method Requirements Specification (MRS), which is a technical document describing what a method that meets the MRS has to offer. Being independent from implementational issues, an MRS details the nature of the method but not the method itself. An abstract language, called MRSL, has been developed to express a MRS. The MD phase translates an MRS into an instantiation of the technical meta model. The method construction phase generates the method and builds the CASE tool. A CAME tool, referred to as MERU, has been built to provide assistance in the tasks of three ME phases (Gupta *et al.* 136-137).

Gupta *et al.* (2001) apply a three-layer architecture to specify an ISD method. At the highest level, the generic view of a method offers a basis for building the abstract model and MRSL. It is used in the MRE phase. The second layer, the metamodel layer, defines the decisional metamodel that is used in the MD phase to make an instantiation. At the third layer, the method is finally produced from the instantiation performed in the second phase. This layer is used in the method construction phase.

The suggestion of Gupta *et al.* (2001) is the only one among the analyzed ME artifacts that clearly distinguishes between different levels of abstraction on which a method can be conceived (cf. the generic view, the metamodel view and the construction view). It also differentiates between three "phases" of the engineering of the method, based on those abstraction levels. These phases stand for our four ME workflows in such a way that the tasks of MRE phase belong partly to the ISDM RE (cf. the method nature) and partly to the ISDM analysis (cf. the simple method and mapping method). The MD phase and the MCI phase correspond to the ISDM design workflow and the ISDM implementation workflow, respectively. Issues of the ISDM evaluation workflow are not addressed in Gupta *et al.* (2001). The ME context is mainly considered from the ME conceptual perspective, because the emphasis in the article is on the MRS language and the decisional metamodel. The ME actions are outlined only on a very general level (cf. the ME infological perspective). The article also contains a short description of MERU. The ISD context is perceived from the ISD infological and ISD conceptual perspectives. Gupta *et al.* (2001) define the notion of a generic work procedure that is composed of procedure elements, which in turn can contain several work elements. A procedure element is an objectified relationship between a product part and a method block. Product parts are contained in the product under development. A method block is a pair consisting of an objective and an approach. Conceptual structures are used to represent the architecture of the product.

Song (1997) gives an outline of the steps for function-driven and quality-driven integration. The steps are divided into two parts, corresponding to high-level integration and low-level integration. The former concerns artifact models, properties, principles, representations, and processes. The latter involves model components (e.g. classes), criteria, guidelines, measures and notations.

The function-driven integration is applied when new functionalities are searched for the existing method from other methods, or parts thereof. In the high-level integration, the process starts with consideration of property integration with the aim of revealing properties that are not effectively supported by the existing method. Next, new and promising ISD principles are searched for and integrated into the existing method. Some of these steps correspond to the ISDM RE workflow but because Song (1997) does not provide any guidance to requirements engineering of the method we locate these steps in the ISDM analysis workflow in Table 35. In the artifact model integration the existing method is enhanced with new artifact models (IS meta models), and in the process integration the processes for specifying the new artifact models are associated into the existing method. These steps are related to the ISDM analysis workflow. Finally, in representation integration the notations of design artifacts are made uniform. This is concerned in the ISDM design workflow in our ontology. The low-level integration considers issues that are related to the ISD datalogical perspective in the ME design workflow.

The quality-driven integration does not aim to enhance the functionalities of the ISD method but to improve its quality in terms of more general criteria (e.g. more maintainable software, more efficient ISD process, etc.). At the high-level, the quality-driven integration consists of at most process and representation integration, and at the low-level it concerns guidelines, measures, actions, and notations. The steps can be located into the ME workflows as above.

The integration process is described in terms of ME deliverables, not by detailing steps. In the ISD context the ISD actions (processes) and the ISD deliverables (artifacts) are distinguished.

Based on the three-level architecture of the method, Vlasblom *et al.* (1995) suggest steps for deploying development models in specific situations. The process starts with the project initiator analyzing the project situation and consulting the library of available development models. After choosing the model the profile of which best matches the specific situation profile, it is modified until a scenario for project direction and operation is achieved (ibid p. 602). In addition, Vlasblom *et al.* (1995) provides steps to the construction of a development model from existing approaches / projects. The steps are: familiarization with the established practice (i.e. completed projects), formalization of the development model, establishing the model situation profile, and feedback to the library of building blocks. The steps cover, to some extent, first four of our ME workflows. They are, however, described on a highly general level, viewing from the ME systelogical and ME infological perspectives.

Nuseibeh *et al.* (1996) generally outline a ME process of method design and construction. The process is composed of the following steps: (1) identify development techniques: derive from the requirements of a method the techniques that the target method will deploy, (2) identify ViewPoint templates that need to be constructed to describe these techniques, (3) describe templates, (4) reuse templates, (5) identify and describe inter-ViewPoint relationships, and (6) construct local (ViewPoint) development process models. Nuseibeh *et al.* (1996) also consider method integration, and to advance the integration they advocate the use of pairwise inter-ViewPoint relationships or rules.

In Nuseibeh *et al.* (1996) the descriptions of the ME context and the target ISD context are given on a very general level. The ME steps concern the ISDM analysis, ISDM design, and ISDM implementation workflows, the emphasis being on the ISDM analysis. The ISD context is perceived through ViewPoint patterns, meaning that ISD actions ("work plan") and ISD deliverables ("specification") are mainly considered from the ISD infological perspective.

The generic process model by Ralyte *et al.* (2003) is based on a strategic process meta-model, called Map (Rolland *et al.* 1999), and describes ME process in terms of intentions and strategies. An intention is a goal that can be achieved by the performance of the process (expressed like "Set Method Engineering Goal", Ralyte *et al.* 2003, 97). A strategy represents the manner in which the intention can be achieved. Ralyte *et al.* (2003) present separate models for assembly-based method engineering, extension-based method engineering, and paradigm-based method engineering. The process models are expressed in the form of a graph in which nodes stand for intentions and edges correspond to strategies. The model applies the ME systelogical perspective and, partly, the ME infological perspective. It does not explicitly recognize the ME deliverables.

To summarize from the analysis of the coverage and emphases of the ME artifacts, we can state that the only ME artifact which applies the ME workflow-like action structure is the one of Gupta *et al.* (2001). But because it considers the ME context mainly from the ME conceptual perspective, ME actions are only generally outlined. ME actions of the ISDM evaluation are ignored. Harmsen (1997) and Tolvanen (1998) apply an ME action structure that consists of the collection of knowledge, selection of methods (fragments in Harmsen (1997)) and integration (Harmsen 1997) and/or adaptation (Tolvanen 1998). In the ME steps Harmsen (1997) includes also project performance and method base administration, whereas Tolvanen (1998), based on his incremental approach, specifies three steps succeeding the method use. In Song (1997) ME work is decomposed into steps based on the target of integration. The steps take a rather narrow scope to the ME workflows. In Vlasblom *et al.* (1995) and Nuseibeh *et al.* (1996) ME work is only partially, and on a general level, covered.

In all the ME artifacts support for the ME context is mainly focused on ME actions and ME deliverables. ME purposes are considered in Harmsen (1997), Ralyte *et al.* (2003) and partly in Vlasblom *et al.* (1995). In some ME artifacts (Harmsen 1997; Tolvanen 1998) ME facilities are also discussed. Harmsen (1997) provides the most extensive treatment of issues within the ME domains. The

ME infological and ME conceptual perspectives are most commonly applied among the ME artifacts. In the ISD context the ISD infological perspective and the ISD conceptual perspective are considered in Harmsen (1997), Gupta *et al.* (2001) and Song (1997). Tolvanen (1998) addresses the ISD conceptual perspective, and Nuseibeh *et al.* (1996) consider the ISD infological perspective.

To help the comparison of the selected ME artifacts with MEMES, we have included MEMES in the same table (Table 36) with the others. As specified in Chapter 11, MEMES applies three perspectives (systelogical, infological, and conceptual), on the ME layer as well as on the ISD layer. Implied from the perspectives applied, MEMES addresses four contextual domains. On the ISD layer MEMES addresses ISD objects as representational artifacts and conceptual constructs. Compared to the analyzed ME artifacts MEMES is more comprehensive in terms of ME workflows, perspectives and contextual domains. In MEMES the ME workflows are decomposed into well-structured tasks and steps. In the other ME artifacts ME actions are mostly expressed in short outlines. They are very far from being considered even as an ME method skeleton. In this sense, our proposal for a method skeleton means a considerable enlargement and elaboration of the body of methodical knowledge about ME.

12.4.3 Conclusions from the Comparative Analysis

The purpose of our comparative analysis was to find out how MEMES compares with the existing ME artifacts and how MEMES performs as a frame in the comparative analysis. The following conclusions can be drawn from the analysis of seven ME artifacts.

There are only some ME artifacts (i.e. Harmsen 1997; Tolvanen 1998; Song 1997) for which information is provided about the processes and methods by which the artifacts have been produced. In contrast, we have explicitly described our research process and research methods. We have also brought out how MEMES has been validated and verified. Most of the analyzed artifacts (Tolvanen 1998; Gupta *et al.* 2001; Vlasblom *et al.* 1995; Ralyte *et al.* 2003) are targeted at engineering a project-specific method. As regards the ME strategies, there are ME artifacts that apply the integration strategy (e.g. Harmsen 1997; Song 1997), and those which follow the adaptation strategy (e.g. Tolvanen 1998; Gupta *et al.* 2001). Ralyte *et al.* (2003) recognize a number of ME strategies and pursue to integrate them with the generic model. MEMES is aimed at the engineering of a generic or domain-specific method. It can be used in conjunction with any ME strategy, although with some elaborations.

Gupta *et al.* (2001) is the only ME artifact in the ME literature which decomposes the ME process into ME workflows based on ISD perspectives. However, it considers the ME mainly from the ME conceptual perspective, and the ME actions are only generally outlined. In most of the ME artifacts (e.g. Harmsen 1997; Tolvanen 1998) the ME process is decomposed into the steps such as the collection of knowledge, selection of methods, and integration and / or adaptation. Tolvanen (1998) specifies also three steps succeeding the method use. Some artifacts provide a narrower and/or more general view of the ME process. Harmsen (1997) provides the most extensive treatment of issues within the ME domains. The other ME artifacts mostly focus on ME actions and ME deliverables from the ME infological and ME conceptual perspectives. From the ISD perspectives some of the ME artifacts (e.g. Harmsen 1997; Gupta *et al.* 2001; Song 1997) apply the ISD infological and ISD conceptual perspectives. The other artifacts are more limited in their scope. MEMES suggests approaches and steps for three ME workflows based on the perspectives. It applies three perspectives (i.e. systelogical, infological, and conceptual) on the ME layer as well as on the ISD layer. Thus, compared to the analyzed ME artifacts MEMES is more comprehensive. Since we have used the notion of comprehesiveness in the contextual sense (cf. ME workflows, perspectives and contextual domains), what has been said above means that with MEMES, compared to the other ME artifacts, it is much better possibilities to recognize, understand, represent and engineer contextual features of ME contexts, ISD contexts and ISD methods.

MEMES was found to be a feasible frame of reference in the comparative analysis. It provided a useful categorization of methodical views with which the background and orientation of the ME artifacts could be revealed, analyzed and compared. The ME workflow structure of MEMES also assisted in the analysis and comparison of functional aspects of the ME artifacts on a general level. The perspectives and the contextual domains, on the ME layer as well as on the ISD layer, helped us factorize and assess the coverage and emphases of the objects systems of the ME artifacts. MEMES offers still more means to elaborate the analysis. The IS perspectives and the IS domains can be used to examine which kinds of phenomena in the IS and the OS_{IS} are recognized in the ME artifacts. Futhermore, it is possible to analyze and compare, on a more detailed level, which concepts and constructs the ME artifacts suggest to use for viewing reality.

12.5 Summary and Discussions

The purpose of this section was to evaluate the applicability of MEMES from the viewpoints of framing, constructive and analytical intentions. We applied one of the ME workflows in MEMES, namely the ISDM evaluation, to make sense of and structure this evaluation context at hand. Following the steps contained in the workflow we specified the evaluation criteria, selected the evaluation methods, carried out the process of evaluation and reported on findings and lessons. Here, we present the summary of the evaluation process and its results.

We evaluated MEMES in three ways. First, we made the retrospective analysis of one of the prior ME contexts, namely the OSSAD project, to examine how MEMES serves as a frame. The analysis showed that the process of the OSSAD project could be structured and analyzed with a large variety of ME action structures provided by MEMES. The ME strategies and the ME approaches specified in MEMES served as a concrete basis with which the orientation of the the OSSAD project and the OSSAD methodology could be analyzed. The IS ontology, embedded in MEMES, was viable in the evaluation of the universes of discourse of the main office models. In conclusion, we argue that MEMES provides structured and feasible means to the retrospective analysis of ME contexts from multiple viewpoints. The information got from the analysis is useful to making improvements in the applied conventions. The retrospective analysis also yielded a structured view of main features (e.g. coverage and emphasis) and approaches of the OSSAD project and revealed some problems. The OSSAD project was found quite comprehensive although it paid insufficient attention to some ME tasks (e.g. analysis of existing methods and models) and perspectives (e.g. IS conceptual perspective). The essence and main characteristics of the project seem to be induced by the functional approach applied in the ISDM analysis workflow and the IS-driven approach in ISD modeling. The OSSAD methodology itself is rather extensive covering several phases, ranging from contracting to implementation of an office support system. However, it totally ignores conceptual modeling. The first three phases are emphasized in the methodology. The methodology is structured upon two major constructs: phase structure and office modeling structure. Three basic models provide concepts and constructs for viewing an office and office work from three different IS perspectives, which appeared to be comparable to the IS infological, IS datalogical, and IS physical perspectives in MEMES. In all these models the functionalist approach (Hirschheim 1986) is applied, meaning that an office is seen as a set of information flows in which the semantics of documents remains uncovered.

Second, we made a retrospective analysis of the MEMES effort to evaluate how MEMES performed in the constructive sense. We presented a detailed description of how MEMES was deployed, according to the reflection-in-action approach, in the MEMES effort and tried to find out how useful it was. We presented a structured view of the MEMES effort decomposed into two parallel and iterative sub-processes, the RW process and the reflection process. We showed that MEMES provided constructs with which the RW process and its deliverables could be framed and described in a structured and comprehensive fashion. MEMES helped us categorize ME actions and issues according to well-defined action structures and perspectives. The target of the effort was so abstract and fuzzy that it would have been very difficult, without the support of gradually enlarging MEMES, to make sense of and shape it. Although the RW process was highly iterative, the use of MEMES, even in its early versions, made iterations manageable and thus the RW process more efficient. To conclude, based on the analysis of, and the experience from, the MEMES effort, we argue that MEMES offered a feasible support for the engineering of MEMES. We have, however, to remember that the retrospective analysis was based on subjective assessments. It is also noteworthy that the MEMES effort was not an ordinary ME endeavor but a research project, which engineered the ME skeleton. Hence, from the evidence obtained here we cannot conclude that

MEMES would be applicable to all kinds of ME efforts. Regardless of what has been said above, we believe that since MEMES, even as a half-ready artifact, appeared to be useful in the MEMES effort, it has much to offer also for other kinds of ME efforts, once it is first elaborated into a complete ME method.

Third, we made a comparative analysis of existing ME artifacts to evaluate the applicability of MEMES in the analytical sense and to find out how MEMES compares with those artifacts. For the comparative analysis we selected the most advanced artifacts in the ME literature. The first part of the analysis uncovered the backgrounds, application areas and ME approaches of the artifacts. In the second part we considered the coverage and emphases of the artifacts in terms of ME workflows, perspectives and contextual domains. The analysis showed that only few ME artifacts have been constructed upon a sound theoretical basis and with a proper research methodology. Most of the artifacts have been engineered for the purposes of method customization or configuration, not to engineer a generic or domain specific method. The integration strategy is most commonly applied. There are some artifacts that cover the ME workflows more extensively than MEMES. However, they do not provide as detailed guidelines for the workflows as MEMES does. Nor do they cover as comprehensively as MEMES does the perspectives and contextual domains on the ME and ISD layers. All but one (i.e. Gupta *et al.* 2001) deploy "technical" meta models which do not allow the recognition of all that diversity of contextual aspects which is typical of the ISD methods. To conclude from the use of MEMES as a framework in the comparative analysis, we can say the following. First, MEMES provided a useful categorization of methodical views with which the background and orientation of the ME artifacts could be analyzed and compared. Second, the ME workflow structure included in MEMES was found feasible in the analysis and comparison of functional aspects of the ME artifacts on a general level. Third, the perspectives and the contextual domains, both on the ME layer and on the ISD layer, helped us factorize and assess the coverage and emphases of the objects systems of the ME artifacts. Fourth, due to the fact that MEMES is strongly anchored on the contextual approach and OntoFrame, it provides relevant concepts and constructs for the recognition, understanding, representing and engineering of contextual aspects and structures of ISD methods. This gives us a general basis to argue for the suitability of MEMES as a "yardstick" to the analysis and comparison of the ME artifacts.

13 CONTRIBUTIONS AND FURTHER RESEARCH

Organizations are nowadays required to act more effectively, to face shorter time-frames and respond in environments where they are confronted by an accelerating pace of change. Rapid and pervasive transformations in business, technology and application environments increase pressures to develop new and better information systems with higher productivity. This intensifies demands to renew and customize current ISD methods, as well as to engineer new kinds of ISD methods. Method engineering (ME) is related, in an intrinsic and pervasive fashion, to every ISD effort. Those days are over when ME was regarded as an "unnecessary nuisance" which could be accomplished with minimum effort. At present ME is an endeavor which has to be performed effectively with proper methodical support. Unfortunately, there is a paucity of ME artifacts that could provide adequate support for ME efforts.

Our objective in this thesis has been to develop intellectual and methodical support for method engineering. The research domain consists of the following sub-domains: IS, ISD, ISD method, ME, ME method. The research problem stated in Chapter 1 is: How to conceive and methodically support the engineering of an ISD method? We have responded to this research problem by crafting two design artifacts (in terms of Hevner *et al.* 2004), the ontological framework and the method skeleton for ME. The ontological framework, called OntoFrame, aims to provide a coherent and comprehensive groundwork for conceiving, understanding, structuring and presenting phenomena in the research domain. The method skeleton for ME, called MEMES, contributes to the support of ME process.

In this chapter we first present an overview of the two design artifacts, considering how they respond to the research questions derived from the research problem mentioned above. Second, we apply the research framework of Hevner *et al.* (2004) to describe each part of OntoFrame and MEMES, highlighting their motivation, importance, novelty and significance. Furthermore, we point out some limitations in our contributions. Third, we bring out several directions for further research.

13.1 Contributions

13.1.1 Overview

We have decomposed the research problem into three research questions in Chapter 1. The questions are: (1) What is the conceptual foundation with which phenomena in the research sub-domains can be conceived, understood, structured and presented? (2) What is the nature, contents and structure of an ISD method? (3) How to structure and support the process of method engineering? In what follows, we describe our contributions in terms of how they are able to answer these questions. In Table 37 we show which parts of our work contribute to each of the research questions.

TABLE 37 Research problems and contributions

Research problems	Contributions	Chapters
RQ1: *What is the conceptual foundation with which phenomena in the research sub-domains can be conceived, understood, structured and presented?*	OntoFrame	Chapters 2-10
RQ2: *What is the nature, contents and structure of an ISD method?*	ISD ontology ISD method ontology ME ontology	Chapter 8 Chapter 9 Chapter 10
RQ3: *How to structure and support the process of method engineering?*	ME ontology ME method ontology MEMES	Chapter 10 Chapter 10 Chapters 11-12

To answer the *first research question* we have engineered the extensive ontological framework (OntoFrame) that covers the five research sub-domains. OntoFrame has been anchored upon several theories. The selection of theories has been guided by the pursuit of generality and the contextual approach. Implied from the former we gave the preference to the theories that help us form a holistic view of the issues in the research sub-domains. These theories are philosophy, semiotics and systems theory. On the other hand, we saw it of vital importance to recognize, understand, structure and represent meanings of things in reality. This is enabled through viewing things as parts in their contexts. Resulting from the exhaustive search for theories that recognize the importance of meaning we selected semantics, pragmatics and theories of human and social action. These theories constitute the underpinning upon which the contextual approach has been defined. OntoFrame is a five-dimensional framework. The first dimension stands for generality. At the one extreme of this dimension there is one concept, a thing, from which all the other concepts have been derived. At the other extreme there is a very large set of concepts intended for the recognition of features of ISD contexts and ME contexts. The second dimension is formed by the categorization of features into seven contextual domains (i.e. the purpose domain, the actor domain, the action domain, the object domain, the facility domain, the location domain, and the time domain). The third dimension stands for the strictly defined perspectives (i.e. the systelogical perspective, the infological perspective, the conceptual perspective, the datalogical perspective, and the physical perspective) through which particular views of the features in reality can be established and applied. The fourth dimension contains four processing layers (i.e. IS, ISD, ME, RW). Finally, the fifth dimension involves levels on which models can be specified (i.e. instance models, type models, meta models, meta meta models, etc.).

The most important research subject of this thesis is an ISD method. Therefore it was taken as the target of our *second research question*: What is the nature, contents and structure of an ISD method? As regards the *nature* of an ISD method, we have given, based on the literature survey, an overview of roles and extent in which ISD methods are used in practice. We have also provided a structured description of benefits from, and problems in, the method use in practice. Second, we have presented a holistic definition of the ISD method and defined the ISD method ontology, which highlights, in a more structured and comprehensive fashion than any other presentation, various facets of the notion. We have also defined a framework which, based on the view of the ISD method as a "carrier" of ISD knowledge, helps classify ISD methods. As regards the *contents* of an ISD method, we have defined, applying once again the contextual approach, the comprehensive ISD ontology, which specifies in a structured manner the features that an ISD method should describe and/or prescribe. The ISD ontology covers four ISD domains (i.e. the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain) and four ISD perspectives (the ISD systelogical perspective, the ISD infological perspective, the ISD conceptual perspective, and the ISD datalogical perspective). To elaborate the *structure* of an ISD method, we have defined seven methodical views based on the semiotic ladder. According to these views, an ISD method can be seen as being composed of the issues related to prior ME contexts and prior ISD contexts (the historical view) and to target ME contexts and target ISD contexts (the application view), of conceptual constructs (the contents view), of linguistic expressions (the presentation view), of physical things in electronic or paper form (the physical view), and of method components (the structural view). To elaborate the structural view, we have also defined the contextual interface of a method component and a five-dimensional scheme for the classification of components.

To answer the *third research question,* we have crafted MEMES, the method skeleton for method engineering, by deploying the "building blocks" offered by the ISD ontology, the ME ontology, and the ME method ontology. The purpose of MEMES is to provide support for the engineering of generic and domain-specific methods. MEMES covers three ME perspectives (i.e. the ME systelogical perspective, the ME infological perspective, and the ME conceptual perspective) and three ISD perspectives (i.e. the ISD systelogical perspective, the ISD infological perspective, and the ISD conceptual perspective). It is composed of three ME workflows: ISD method requirements engineering, ISD method analysis, and ISD method evaluation. MEMES is not a complete ME method. To make MEMES more complete, it has to be enhanced with the issues related to at least the ME datalogical perspective and the ISD datalogical perspective. The applicability of MEMES has been evaluated by conceptual and empirical methods.

13.1.2 OntoFrame

OntoFrame is composed of four main parts: the core ontology, the contextual ontologies, the layer-based ontologies, and the method ontologies. These are further divided into several component ontologies. We will briefly discuss each of them in the next sub-sections. Here we consider OntoFrame as a whole, summarizing its engineering process, assessing it with quality criteria defined in Chapter 1 and discussing its implications to research and practice.

OntoFrame has been built upon two disciplines, metamodeling and ontology engineering, and with two approaches, the inductive approach and the deductive approach. In building the core ontology we made a thorough analysis of existing generic frameworks and ontologies and derived our ontology from them by selection, integration, and customization. In engineering the contextual ontologies we first searched for disciplines and theories that address social and organizational contexts and derived a basic categorization of concepts into contextual domains from them. After that we enriched the contents and structure of each contextual domain by analyzing existing artifacts. For the other parts of OntoFrame we applied the deductive approach by specializing concepts and constructs into lower-level ontologies from higher-level ontologies. In this process we utilized heavily the literature of existing artifacts in order to elaborate and customize the derived concepts and constructs and make them consistent with specific sub-domains. We applied the integration strategy of ontology engineering whenever it was possible. In this way we could import existing knowledge from other sub-domains in cases where these views and concepts were sufficiently clear and stable and they matched with our general premises.

The engineering of OntoFrame was based on the comprehensive analysis of the literature, accomplished in several phases. First we conducted an extensive analysis of fifteen of the most advanced frameworks, meta models, frames of reference and ontologies related to the IS, ISD, ISD methods, and ME (Section 2.5). In addition, comparative analyses of existing artifacts were carried out on more detailed level in each of the sub-domains. These analyses are related to fundamental concepts (cf. the core ontology) in Section 3.10, the notions of information system, object system, and utilizing system in Section 5.1.6, IS concepts and IS perspectives (cf. the context ontology and the perspective ontology) in Section 6.4, and meta levels (cf. the model level ontology) in Section 7.3. Furthermore, we analyzed and compared frameworks of ISD (cf. the ISD ontology) in Section 8.5, frameworks, architectures and reference models of ISD methods (cf. the ISD method ontology) in Section 9.7, and classifications of ISD method components in Section 9.8.6. Finally, we analyzed and compared existing ME artifacts in Section 12.4. We have also presented a plethora of references to the pertinent literature in each subject matter area in order to describe and compare different conceptions and concepts. These analyses and references clearly revealed the great divergence which prevails in the concerned fields at present. Comparisons with OntoFrame showed that our ontological framework is much more comprehensive than any of the existing artifacts. Recalling that OntoFrame has been engineered according to the contextual approach, OntoFrame recognizes in the most extensive way the contextual features of the research sub-domains.

OntoFrame is presented in informal and semi-formal fashions. The concepts and constructs of the component ontologies are defined in English and collected into the unified vocabulary in Appendix 1. In addition, the concepts and constructs in each of the component ontologies are presented in meta models in the UML-based ontology representation language (Appendix 2).

Next, we briefly consider how OntoFrame meets the goals stated in Section 1.3. The goals were expressed in terms of comprehensiveness, contextuality, consistency, coherence, generality, clarity, naturalness, generativeness, extensibility, modularity, theory basis, and applicability. The evidence obtained from the comparative analyses, together with the use of "universal" theories as the basis of the most component ontologies, strengthens our confidence that OntoFrame addresses, in a comprehensive fashion, relevant contextual phenomena in all five sub-domains. To advance the consistence and coherence of OntoFrame we metamodeled it in a disciplined fashion and formed the unified vocabulary. The meta models enforced the specificity of the concepts, relationships and constraints. The definitions in the vocabulary explicitly show how the concepts are inter-related through their intensions. To balance between specificity and generality OntoFrame has been composed of ontologies at several levels. The most generic component ontologies in the core ontology provide concepts that refer to the fundamentals of reality (e.g. thing, relationship, point of view). Clarity and naturalness are qualities about which we have not carried out empirical studies. Nevertheless, several features in OntoFrame have positive impacts on these qualities. For instance, rooting the concepts of OntoFrame upon proper theories (cf. semiotics and linguistics) contributes to naturalness, and the modular structure of the framework enhances its clarity. In addition, we have extensively utilized the established terminology that is commonly used in current ISD methods.

The generative structure is inherent to OntoFrame due to the top-down approach by which the concepts and constructs of lower level ontologies have been derived from those of higher level ontologies. OntoFrame is composed of clearly structured parts and components. The framework with the generative nature and modular structure is easier to extend, if necessary. Extensions can be carried out by specializing some of the concepts in OntoFrame, or integrating new concepts into the existing ones. OntoFrame has been rooted in several theories, ranging from philosophy, semiotics and linguistics to systems theory, activity theory, and ISD theories.

The last goal in the list concerns the applicability of OntoFrame for framing, analytical and constructive intentions. We have deployed OntoFrame as a frame in multiple comparative analyses of existing artifacts. We found it to be a feasible framework with which the backgrounds, comprehensiveness, coverage, emphasis, etc. of models, meta models, frameworks, reference models, architectures, etc. could be recognized and analyzed. OntoFrame has been used mainly as a general-purpose framework, and it should be elaborated and customized when deployed with more specific aims. OntoFrame has also been used as a groundwork for the construction of MEMES. The extensive vocabulary of OntoFrame was found feasible in specifying the scope, structure and contents of MEMES. In particular, the main structures of the ISD ontology, the ME ontology and the ME method ontology offered usable building blocks from which the structure of MEMES could be crafted in a straightforward manner. The use of OntoFrame to frame the universe of discourse has not been investigated in this work. Hence, to find out how OntoFrame performs in this respect requires empirical studies in future.

Next, we consider rigor from the viewpoint of OntoFrame. Rigor "addresses the way in which research is conducted (Hevner *et al.* 2004, 87)). It "is derived from the effective use of the knowledge base - theoretical foundations and research methodologies" (ibid p. 88). This study has been rooted in two disciplines: metamodeling and ontology engineering. Metamodeling is a young discipline without established theoretical and methodological traditions. In ontology engineering there are also divergent conceptions about how to engineer and validate an ontology (Gomez-Perez 1995; Gruninger *et al.* 1995; Guarino 1997; Shank *et al.* 2003). Some researchers use competence questions to ascertain the comprehensibility and usefulness of an ontology (e.g. Gruninger *et al.* 1995). Others suggest reviews, problem solving and transaction testing (e.g. Shank *et al.* 2003). OntoFrame is too large to be derived from and validated with competence questions. We have aspired at rigor with the following means. We have strongly utilized relevant theories on each level of OntoFrame (i.e. the deductive approach). At the same time we have imported views and concepts from the existing literature about fundamental elements of reality, information systems, IS development, ISD methods, etc. We have applied the integration strategy to combine these into a consistent and coherent whole. In integration, contradictory views have been reconciled by principles of metamodeling.

Implied from the above, we argue that OntoFrame is of benefit to both research and practice. With the component ontologies contained in OntoFrame it is possible to achieve a clear understanding of the contextual phenomena in information systems, information system development and method engineering. OntoFrame provides a reference background for scientists and professionals, thus enabling them to express themselves about matters in the concerned sub-domains in a structured and well-defined way. The second thrust of the thesis is to provide a comprehensive and unified framework allowing to relate different approaches to each other. It provides bridges between various approaches, disciplines, and decades. OntoFrame also provides an extensive, consistent and coherent groundwork for teachers and students, who can benefit from its large collection of meta models and comprehensive vocabulary with clear definitions.

In the following sub-sections we describe in more detail the contents of the main parts of OntoFrame, the principles with which they have been engineered, and how they compare with the existing artifacts.

13.1.3 Core Ontology

The core ontology (Chapter 3) provides key concepts and constructs for conceiving, understanding, structuring and representing fundamental phenomena in reality. These fundamentals serve as a basis upon which all other parts of OntoFrame are anchored, thus furthering the consistency and uniformity of the large ontological framework. The core ontology is composed of seven component ontologies: the generic ontology, the semiotic ontology, the intension/extension ontology, the language ontology, the state transition ontology, the UoD ontology, and the abstraction ontology.

We have engineered the core ontology with the top-down approach (cf. Noy *et al.* 2001; Uschold *et al.* 1996), with the adherence to the constructivist position, and with strictly defined viewpoints. The generic ontology is a top ontology (Guarino 1998), which defines the most fundamental concepts and constructs (e.g. thing, relationship, point of view). Applying specific points of view we have specialized lower-level ontologies from the core ontology. For instance, things have been specialized into concepts, referents, and signs, and concepts have been further specialized into abstract and concrete concepts, individual and generic concepts, type and instance concepts, and basic and derived concepts. Correspondingly, conceiving the signs from the language viewpoint we have recognized expressions, symbols, labels, proper names and common nouns. With the viewpoint distinguishing between static things and dynamic things we have defined the notions of state, event, and state transition.

Special attention in the core ontology has been given to the abstraction structures, thus emphasizing the importance of abstraction to human perception and thinking. We have specified the basic concepts, structural rules, and constraints for four basic principles of abstraction (i.e. classification, aggregation, generalization, and grouping). Also the intensional and extensional derivations of predicate concepts have been defined. In addition, we have distinguished between the first-order abstraction and the second-order abstraction. While the first-order abstraction concerns things, the second-order abstraction (or the predicate abstraction) pertains to predicates characterizing things. The predicate abstraction is useful in defining perspectives through which the complexity of reality can be managed.

We have briefly reviewed well-known top ontologies and made a comparative analysis of two of the most acknowledged approaches to ontology engineering at the core level in the IS field: the BWW model (Wand 1988a; Wand *et al.* 1989; Wand *et al.* 1990a; Wand *et al.* 1990b) and the Frisco Framework (Falkenberg *et al.* 1998). The BWW model is based on the philosophy of Bunge (1977) according to which the world is made of things that possess properties. Wand (1988a) and Wand *et al.* (1989, 1990a, 1990b) apply Bunge's ontology to define essential concepts of an IS. The Frisco framework (Falkenberg *et al.* 1998) aims to provide an ordering and transformation framework allowing one to relate many different IS modelling approaches to each other. Based on the constructivist position, the framework has been built, piece by piece, upon one individual concept, called a thing. The framework is composed of five layers: the fundamental layer, the layer of actors, actions and actands, the layer of cognitive and semiotic concepts, the layer of system concepts, and the layer of organizational and information system concepts.

Our comparative analysis showed that both of the artifacts contain several deficiencies. Due to its objectivist assumptions, Bunge's ontology is oriented towards the physical world, and therefore does not provide concepts for human perception and social context. It can also be questioned whether the BWW model includes constructs that are not relevant for perceiving IS's. The BWW model does not address the issues of the semiotic ontology, the intension/extension ontology, and the language ontology. In addition, only some of the concepts in the UoD ontology and the abstraction ontology are recognized in the model. The Frisco framework does not provide the notion of a point of view, which is important to distinguish and discuss different conceptions about reality. It does not explicitly recognize the semiotic concepts, although they are needed as a baseline for more specialized concepts and constructs. Its suggestions for abstraction concepts are inadequate.

13.1.4 Contextual Ontologies

The contextual ontologies (Chapters 4-7) help us recognize, understand and model phenomena in reality as contexts and/or within contexts. They also assist in distinguishing between different kinds of contexts and examining and modeling contexts from strictly defined points of view. The contextual ontologies comprise four component ontologies: the context ontology, the layer ontology, the perspective ontology, and the model level ontology.

The notion of a context is important to the understanding of things in reality for many reasons. The only way to make sense of a thing is to associate it with its proper environment, whether intentionally, structurally, functionally, organizationally, locationally, and/or historically determined. A thing itself can be a complex context, and it cannot be understood without decomposing it into contextual ingredients. A context is a universal concept, which plays an important role in several disciplines. Also in the IS field, context is frequently referred to (e.g. Mylopoulos 1998; Motschnik-Pitrik 1999; Motschnik-Pitrik 2000; Abecker *et al.* 2000; Myrhaug 2001; Rolland *et al.* 1995; NATURE Team 1996; Rolland *et al.* 2000). There are also approaches which refer, although not with the term of context, to contextual aspects (e.g. Zachman 1987; Sowa *et al.* 1992;

Loucopoulos *et al.* 1998; Kirikova 2000; Mentzas *et al.* 2001; Chiu *et al.* 1999)). Regardless of its importance, context is not included in any uniform ontology in the IS field.

We have constructed the contextual ontologies with five principles. First, we wanted to ensure a sound theoretical basis for these ontologies. For this reason, we made an extensive search for and review of theories on three topmost steps in the semiotic ladder (Stamper 1973; Stamper 1996). The selected theories highlight various aspects of the meaning of a thing. Second, we explicitly specified the contextual approach according to which phenomena in reality can be perceived as contexts and/or within contexts. According to the approach a context is composed of parts which all have specific roles of their own. It is a totality in which each of its parts gets the meaning through its position in the whole and through the relationships it has with the other parts. The parts in the context reflect specific aspects of some of the seven contextual domains that are: purpose, actor, action, object, facility, location, and time. Depending on the chosen point of view (i.e. the semantic view, the pragmatic view, the activity theoretic view), a different set of contextual domains forms the "nucleus" of the context, thus affecting what parts at least should be perceivable in the totality for being considered a context.

Third, derived from semiotic viewpoints we defined the notions of information system, object system and utilizing system, and recognized four processing layers on which information processing may be situated. The processing layers are: information system, information system development, method engineering, and research work. Fourth, we specialized the notion of a point of view to define five generic perspectives. These perspectives are: systelogical, infological, conceptual, datalogical, and physical perspectives. They assist us to recognize certain contextual aspects in the information processing contexts. Their use is necessary to manage the complexity faced with in reality. Fifth, we applied the principle of classification to establish a hierarchy of model levels. The hierarchy is composed of instance models, type models, meta models, meta meta models, etc. Models are specialized into a) informal, semi-formal, and formal models, b) subjective, inter-subjective, and objective models, c) structural and dynamic models, and d) descriptive and prescriptive models. The model levels are related to one another with relationships based on the language ontology and the abstraction ontology.

The contextual ontologies differ substantially from the related works. In our comparative analysis of the most advanced IS artifacts we found that Iivari's conceptual framework (Iivari 1989a) and the ISA framework (Zachman 1987; Sowa *et al.* 1992), although containing large varieties of concepts and constructs corresponding to our context ontology, dismiss several important contextual parts and suggest conceptualizations that remain on quite a general level. Other presentations, such as Essink (1986, 1988), Falkenberg *et al.* (1998), Harmsen (1997) and Olle *et al.* (1988a), were found even more limited. With respect to the perspective ontology, Iivari's framework (Iivari 1989a) is the most comprehensive. Other relevant works are Essink (1986, 1988), Freeman *et al.*

(1994), Olle *et al.* (1988a), Sowa *et al.* (1992) and van Swede *et al.* (1993). Although these presentations specify viewpoints that are, to some extent, comparable to ours, the relationships between the viewpoints are not defined as strictly as we have done and the contents of the viewpoints are not defined on such a level of detail and with such a large scope that would compare with ours. What also makes our approach novel is that the perspective ontology is applicable on all the processing layers, not only for the IS's.

The layer ontology and the model level ontology do not provide any new fundamental principles or insights compared to the earlier work (e.g. Gasser 1986; Iivari 1989a; Falkenberg *et al.* 1992a; ISO 1990; Brinkkemper 1990; OMG 2002) when considered on a general level. In both of the ontologies the issues are, however, considered through the contextual point of view, resulting in, for instance, that the relationships between IS, ISD and ME are specified in a more detailed manner. Our approach is novel in specifying that the contextual ontologies are orthogonal to each other, thus implying that the information processing contexts on each layer can be perceived from the viewpoint of any contextual domain and from any perspective, and modeled on multiple model levels. The component ontologies used together provide an effective intellectual device for the conceptualization of sophisticated contextual IS phenomena.

13.1.5 ISD ontology

The ISD ontology (Chapter 8) provides necessary concepts and constructs for conceiving, understanding, structuring and representing contextual phenomena in ISD. It is composed of two main parts: the ISD domains and the ISD perspectives. The ISD domains comprise concepts and constructs within four contextual domains (i.e. the ISD purpose domain, the ISD actor domain, the ISD action domain, and the ISD object domain). The ISD perspectives specialize the generic principles defined in the perspective ontology and offer concepts and constructs for viewing ISD from four ISD perspectives (i.e the ISD systelogical perspective, the ISD infological perspective, the ISD conceptual perspective, and the ISD datalogical perspective). The ISD ontology is a conceptualization of the contents of an ISD method. Thus, any attempt to specify an ISD method requires the existence of the ISD ontology.

We have deployed the contextual approach to derive the concepts and constructs of the ISD ontology from, and to inter-relate them with, the contextual ontologies. Our aim has been to include only those parts in the ISD ontology that are common to most of the ISD approaches. As we know, there are a large variety of conceptions about ISD, reflecting divergent paradigms (Hirschheim *et al.* 1989; Iivari 1991; Hirschheim *et al.* 1992a), ISD approaches (cf. Hirschheim *et al.* 1992a; Hirschheim *et al.* 1995; Iivari *et al.* 1998a), and ISD principles. Our purpose has been to constitute a generic and uniform conceptualization of the nature, structure and behavior of ISD that could be shared by different approaches. The ISD ontology provides a solid baseline for the definition of the ISD method ontology and ontologies on the ME layer.

In the comparative analysis of six advanced ISD artifacts, proposed for the description, analysis, comparison and/or engineering of the ISD methods, our ISD ontology was found the most comprehensive. Only some of the analyzed artifacts provide concepts for the ISD purpose domain and the ISD actor domain. Also the ISD object domain is insufficiently addressed. There exists no artifact that provides ISD perspectives. Although all the artifacts contain some constructs for the ISD action domains, our ISD ontology comprises, for instance, a much richer collection of ISD action structures (i.e. the ISD management – execution structure, the ISD workflow structure, the ISD phase structure, the ISD problem solving structure, and the IS modelling structure). These structures appeared to be useful in analyzing other artifacts and structuring the contents of the ISD method. Most of the analyzed artifacts turned out to be totally lacking of the theoretical basis. They have just been abstracted from existing ISD methods. This situation is unsatisfactory for two reasons. First, with a sound theoretical background we can ensure that ISD phenomena become properly conceived, understood and structured. Second, abstracting from existing methods replicates properties of existing methods and does not help recognize phenomena of ISD outside the methods. Our ontology is based on the contextual approach built upon several underlying theories. This increases our confidence that the most essential features of ISD are included in the ISD ontology.

13.1.6 ISD Method Ontology

The ISD method ontology (Chapter 9) provides concepts and constructs to conceive, understand, structure and present contextual aspects of ISD methods. It is based on several fundamental classifications related to ISD knowledge and ISD methods. We distinguished between bodies of knowledge about ISD processes, application domains, IC technologies, and human and social issues. We also differentiated between generic methods, domain-specific methods, organization-specific methods, and project-specific methods. Moreover, we specified seven basic views of the methods, referring to them as the methodical views. These views are: the historical view, the application view, the generic view, the contents view, the presentation view, the physical view, and the structural view.

We presented criteria for acknowledging an artifact as the ISD method. We also recognized and defined three types of artifacts that provide methodical support although they are not acknowledged to be methods. These artifacts are the ISD methodical framework, the ISD methodical skeleton, and the ISD methodical tool kit. By applying the structural view, we defined the notions of a method component and a contextual interface. We also presented a multi-dimensional classification of method components based on the contextual ontologies. We illustrated these notions with examples of method components and method integration. Finally, we made a literature analysis to compare our classification of method components with those presented in the literature. Our classification was found to be the most comprehensive and multi-faceted.

In the ISD literature there is a large variety of conceptions about the nature, structure, contents, role, and significance of ISD methods (see Sections 9.2, 9.3, 9.4 and 9.6). There is also a myriad of different methods developed for ISD. Several attempts to cope with this divergence and to establish a shared conceptualization of the contents and structure of the ISD method have been made in the field (e.g. Iivari *et al.* 1983; Lyytinen 1986; Heym *et al.* 1992a; Avison *et al.* 1995a; Hirschheim *et al.* 1995). We made an extensive analysis of seven well-known frameworks and models aimed either at the comparison and evaluation of the ISD methods, or at categorizing method knowledge. The analysis clearly brought out how divergent conceptions and views in the literature really are. It also showed our ISD ontology covers, much better than the others, the contextual features in the ISD and ISD method. In this sense it is more comprehensive than the current presentations. In addition, our way of structuring the parts and features of the ISD method, based on the contextual approach and the methodical views, makes the ontology more contextual, explicit and easier to apply.

13.1.7 ME Ontology

The ME ontology (Chapter 10) provides concepts and constructs to conceive, understand, structure and present contextual phenomena in method engineering. It is composed of two main parts, the ME domains and the ME perspectives. The ME domains comprise concepts and constructs within four contextual domains (i.e. the ME purpose domain, the ME actor domain, the ME action domain, and the ME object domain). The ME perspectives specialize the generic principles defined in the perspective ontology and offer concepts and constructs for viewing ME from four ME perspectives (i.e. the ME systelogical perspective, the ME infological perspective, the ME conceptual perspective, and the ME datalogical perspective).

The ME ontology has been founded on the basic classifications of ME strategies, ME processes and ME contexts, as well as on the framework integrating these classifications. We distinguished between three ME strategies: creation, integration and adaptation. We also specified six ME processes. Three of them “transform” the ISD method more specific to a certain organization or project (i.e. customization, configuration, and realization). The other three ME processes (i.e. abstraction, deconfiguration, and decustomization) generalize descriptions of the ISD method and are reverse to the aforementioned processes. Based on the above classifications, we recognized three kinds of ME contexts: the method development context, the method customization context, and the method configuration context. Finally, we applied the contextual approach to construct the holistic definition of the ME context. These classifications together with the concepts and constructs of the ME domains offer a comprehensive and solid conceptualization of method engineering.

Method engineering is a rather young discipline. For this reason there is a large variety of concepts and terms used to refer to phenomena in the ME context. Divergent prefixes have been coined (e.g. situation ME, incremental

ME, context-specific ME, simulation-based ME, assembly-based ME, and ontology-based ME) to highlight specific features in the foundations, strategies, approaches, and processes of ME. In the middle of this mix of diverse approaches and nomenclature it is important to have a common vocabulary with which it is possible to recognize, understand, and express the structural, functional, organizational, and behavioral aspects of ME. As far as we know, no serious attempt has been earlier made to achieve such a comprehensive and uniform conceptual foundation and vocabulary for ME as we have presented here.

13.1.8 ME Method Ontology

The ME method ontology (Chapter 10) is composed of concepts and constructs with which contextual aspects of ME methods can be conceived, understood, structured and represented. As was the case with the ISD method ontology, we deployed also here the seven methodical views to decompose the ME method ontology into seven parts. The contents view on the ME method corresponds to the ME ontology. We distinguished between generic ME methods, domain-specific ME methods, organization-specific ME methods, and project-specific ME methods. Furthermore, we defined the notions of an ME method component, an ME model, and an ME technique.

The ME literature provides only fragmented views on the ME context and methodical support to it. There is no uniform suggestion for what the ME method should contain and how it should be structured. Compared to this, our ME method ontology means a substantial complement to the body of the current knowledge. Upon this ontology we have constructed the method skeleton for ME, which we will describe in the next sub-section.

13.1.9 MEMES

MEMES (ME MEthodical Skeleton) (Chapters 11 and 12) is a normative prescription of the ME context, structuring and guiding the process of method development. It has been firmly anchored on the ontological framework, in particular on the ISD ontology, the ISD method ontology, and the ME ontology, and the ME method ontology. MEMES covers three ME perspectives (i.e. the ME systelogical perspective, the ME infological perspective, and the ME conceptual perspective) and three ISD perspectives (i.e. the ISD systelogical perspective, the ISD infological perspective, and the ISD conceptual perspective). It is aimed to give generic support for all the ME strategies.

MEMES contains three ME workflows: the ISD method (ISDM) requirements engineering, the ISDM analysis, and the ISDM evaluation. In the ISDM requirements engineering ME stakeholders' requirements concerning the nature, structure and contents of the ISD method are identified, elicited, prioritized and finally stated as parts of the ME goals. The ISDM analysis denotes an ME workflow which aims to produce high-level descriptions of the ISD method from the ISD infological perspective, and the ISD conceptual perspective. In the ISDM evaluation one or more ISD methods, or parts thereof, are evaluated using the defined criteria. For each of the aforementioned ME workflows, sets of approaches and steps have been provided.

MEMES has been built using the deductive approach to method engineering. This is in contrast to all other efforts in which the inductive approach to ME has been deployed. Our approach has brought several advantages. The conceptual content of MEMES is coherent and consistent. MEMES is clearly organized with the main constructs of the ME purpose, ME action and ME object domains. The contents of ME deliverables become evident through well-defined models on multiple model levels. The deductive approach has also given, compared to the artifacts engineered inductively, a more solid basis to elaborate, customize, and configure MEMES further towards a specific ME method.

Next, we consider how MEMES meets the goals stated in Chapter 11. It was required that MEMES is based on a solid and sound view of the relevant sub-domains. This goal is satisfied by the use of OntoFrame as the conceptual groundwork. Second, it was demanded that MEMES is modular and flexible. MEMES has been build from well-defined constructs of ME workflows, ME perspectives and ME deliverables. All these constructs are rooted on the specific parts of the ME ontology, the ISD ontology and the IS ontology. This foundation serves as a conceptual basis with which “modules” in MEMES can be distinguished, elaborated, reorganized, customized and configured.

It was also required that MEMES is applicable. We investigated the applicability of MEMES with empirical and conceptual methods. Because MEMES is not a complete ME method but a methodical skeleton, its use in real ME contexts was restricted. Therefore, we decided to make the retrospective analyses of two prior ME contexts. The first retrospective analysis addressed the OSSAD project with the aim to examine how MEMES served as a frame and to understand its process and outcomes (cf. framing intention). Second, we used MEMES as a prescription in the contruction of MEMES itself (cf. constructive intention). In the process of construction we deployed the reflection-in-action approach (Schön 1983) according to which the whole process was composed of two parallel and iterative sub-processes, the RW process and the reflection process. We described and evaluated this MEMES effort with the means of retrospective analysis. MEMES was conceptually evaluated by conducting a comparative analysis of ME artifacts in the literature. The purpose was to find out how MEMES compares with those artifacts and how MEMES is suited as an analytical tool (cf. analytical intention).

The two retrospective analyses showed that the approaches, processes and deliverables of the ME efforts could be clearly recognized, structured, represented and assessed with the concepts and constructs provided by MEMES. The use of MEMES in engineering MEMES itself was found useful in many ways. MEMES helped us categorize and structure ME actions and ME deliverables. It made iterations more manageable and the RW process more efficient. These assessments are, however, based on the researcher’s subjective views. This reduces their significance from the viewpoint of validation. More empirical research is needed to obtain stronger evidence on these issues. In the comparative analysis MEMES offered a solid foundation with which the backgrounds, application areas, ME approaches, comprehensiveness and emphases of existing ME artifacts could be analyzed and compared. The analysis showed that only a few existing ME artifacts have been constructed upon a sound theoretical basis and with proper research methodologies. Most of the analyzed artifacts have been engineered for the purposes of method customization or configuration, not to engineer a generic or a domain-specific method. Although there are some artifacts that cover the ME workflows more extensively than MEMES, they do not provide as detailed guidelines for the ME workflows as MEMES does. Nor do they cover, as comprehensively as MEMES does, the perspectives and the contextual domains on the ME and ISD layers.

13.1.10 Discussion of the Contributions

OntoFrame, together with MEMES, constitute a very large whole. To many of the parts in them we have presented significant contributions, while the other parts have been included in the thesis mainly to achieve a coherent and consistent body of work. As far we know, there is no other presentation that would cover such a large spectrum of research sub-domains, on such a detailed level, as we do in this work. We have intentionally aspired after this kind of holistic view in order to avoid the fragmentation of views and conceptions that is typical of most of the research in our field. The holistic view provided in this study enables the recognition, comparison, and integration of current artifacts that have been built upon more limited foundations and views. We also hope that upon the groundwork established in this study it is easier in future to engineer specific artifacts that yet are compatible and interoperable with each other.

Owing to its comprehensiveness the thesis is capable of building bridges between different disciplines, sub-domains, approaches, and time periods. First, anchoring OntoFrame on relevant theories has created connections to e.g. philosophy (cf. ontology engineering in general and the core ontology in particular) and linguistics (cf. the contextual approach). For establishing the process of engineering OntoFrame we have combined and applied the principles of ontology engineering and metamodeling. Second, OntoFrame provides a coherent and consistent view on the issues in five sub-domains: IS, ISD, ISD method, ME, and ME method. To achieve this has required the assimilation of quite different views inherent to the sub-domains. Third, there exist a number of approaches with divergent views on ISD (e.g. transformation approach, decision making approach, problem solving approach, learning approach, etc.). We have tried to identify and define concepts and constructs that can be shared by these approaches, and include them in OntoFrame, in order to provide the possibility to obtain an integrated view on the concerned issues and to specialize the concepts and constructs, when needed, according to specific approaches. Fourth, we have intentionally brought forward and acknowledged conceptions and propositions presented in the past decades. Many scientists, such as Brodie, Chen, Codd, Ein-Dor, Falkenberg, Gorry, Guttag, Hoare, Iivari, Järvinen, Kent, Kerola, Langefors, Mumford, Sisk, Smith & Smith, Stamper, Welke, Yourdon, and Zisman, just to mention some of them, have presented, as early as in the 1970's, seminal ideas that still are worthy of respecting and comparing with more recent ones.

In evaluating and comparing the existing artifacts we have frequently applied the criterion of comprehensiveness. This criterion should not be regarded as a quantitative measure. In contrast, it is highly qualitative and closely related to contextuality. By applying the contextual approach we have tried to distinguish, define and include in OntoFrame and MEMES the most relevant concepts and constructs that are needed to identify, understand and represent the contextual features in the five research sub-domains. Thus, when assessing the comprehensiveness of artifacts by using OntoFrame or MEMES as the "yardstick" we have actually wanted to find out the degree to which the artifacts analyzed take into account the contextual features of the subject matters.

Comprehensiveness is not a value in itself. In fact, it may be questioned whether OntoFrame is too large and too complicated to be feasible. It is sometimes said that "simple is beautiful". For instance, Simon's framework of problem solving (Simon 1960) is composed of only three main concepts, intelligence, design, and choice, and it is universally known and frequently applied. We do not want to deny the benefits of simple frameworks. However, in order to cope with such a large research domain as we do here, a framework inevitably becomes complex. This does not, however, mean that in all situations all the concepts and constructs in OntoFrame, or in MEMES, should be used. Both of the artifacts have a modular structure allowing the selection of only those perspectives, components and constructs that are actually needed.

One of the biggest challenges in this thesis has been to cope with the complexity caused by the large number of the parts and components included in OntoFrame, the extensive number of references made to the relevant literature, and the great divergence of views and terminologies integrated into OntoFrame. It can be estimated that the complexity of work has increased "exponentially" in relation to the number of the concepts and constructs included in Ontoframe. Management of that complexity is one of the contributions of this work.

13.2 Further Research

The thesis offers numerous possibilities to direct further research. In this section we consider how the results of the thesis can be elaborated, extended and deployed in research to come.

OntoFrame provides a baseline for many kinds of specializations and elaborations. Let us consider one example. In pervasive computing paradigm location-awareness is nowadays generalized into context-awareness (Matheus *et al.* 2003), thus emphasizing that it is not enough to know where someone or something is located. In contrast, it is necessary to comprehend more generally in which circumstances someone or something acts or something occurs. To perceive, understand, structure and represent the relevant contextual features of those circumstances necessitates the construction and deployment of a context model that suits the needs of an application area. OntoFrame can provide a foundation to decide on the relevant features of a context and to engineer such kind of context model. More specifically, the context ontology can be used to decide which contextual features are relevant to "context-awareness". The model level ontology offers concepts and constructs to specify how to model those features. The perspective ontology and the ISD method ontology can be used to consider how to integrate the constructed conceptual model into a methodical body containing other models and techniques.

OntoFrame and MEMES can be extended along several dimensions. MEMES, for instance, addresses only three ME perspectives and three ISD perspectives. To enhance MEMES into a complete ME method it is necessary to extend it with the concepts and constructs of at least the ME datalogical perspective and the ISD datalogical perspective. The extension with the ME datalogical perspective means that the ME context is described / prescribed, not only in terms of ME purposes, ME actions and ME deliverables, but also with references to ME roles, ME positions and ME facilities. In parallel to decomposing ME actions into tasks and operations, ME workflows should be specialized to fit a particular ME strategy and ME approach. After these extensions MEMES can provide specialized ME workflows for method creation, method integration and method adaptation. The extension with the ISD datalogical perspective implies that MEMES also contains the ISDM design workflow which helps specify how the ISD effort is to be accomplished. This means, for instance, that instructions are given for how ISD roles and ISD positions are established, how ISD actions are decomposed onto a more detailed level, and how CASE tools are used in the ISD.

OntoFrame and MEMES can be utilized in conceptual, empirical and constructive research in many ways. In the following we first consider four examples of using OntoFrame in conceptual research. After that we discuss the use of OntoFrame and MEMES in empirical and constructive research. OntoFrame has been used in this thesis as a framework in the literature analysis of a large variety of issues. There are many other issues on which this kind of research can be carried out with OntoFrame. For instance, models could be assessed and compared either with the core ontology or with the context ontology. Enterprise ontologies, such as the TOVE (Toronto Virtual Enterise, Fox 1992) and the Enterprise Ontology (Uschold *et al.* 1998) could be analyzed with the support of the context ontology to find out how they perceive contextual features of the enterprise. The extensive abstraction ontology provides a solid basis for analyzing abstraction structures included in conceptual models in fields such as information systems, software engineering, and artificial intelligence.

Second, empirical studies aiming at testing hypotheses are based on conceptual models specifying the underlying concepts and relationships between them. Conceptual models are constructed to reflect how the concerned slices of reality are viewed. OntoFrame can provide an integrated foundation from which relevant parts can be separated for the use of empirical studies, and elaborated if necessary (cf. the ontology of software maintenance by Kitchenham *et al.* (1999) for empirical studies of maintenance). Likewise, OntoFrame can be used to compare and integrate conceptual models of the empirical studies carried out in the same or interrelated domains. This way we can unite fragmented views that empirical studies commonly portray.

Third, OntoFrame can be used to explain and support the evolution of ISD methods. Changes in business environments and technology have constantly led into situations in which existing approaches, methods and techniques have been found inappropriate. Thereupon, customary conventions have been abandoned and new working procedures have been searched for. These kinds of "discontinuation points" have enabled breaking out of rigorous methods and led to the search for, the birth of, and the diffusion of new computing and development approaches and paradigms. Following these "discontinuation points", freedom in the selection of one's own working habits has decreased through demands for more efficiency with "rigor" methods. This kind of cyclic evolution of methods and method use has characterized the last decades. An example of this kind of juncture was a shift from formal life-cycle approaches and "universal" structured methods, such as ISAC (Lundeberg *et al.* 1981), SA (Gane *et al.* 1979) and Structured Design (Yourdon *et al.* 1979), to more novel approaches (e.g. evolutionary and prototyping approaches) and methods (e.g. methods for decision support systems, office information systems, and expert information systems). The latest "discontinuation point" came up with the emergence of WWW technologies and business needs for web applications. Web development required shorter time scales, a tighter linkage between business models and software architecture, and greater importance on the contents management of web sites (Henderson-Sellers 2003, 78). These requirements were not possible to achieve with the disciplined use of rigor object-oriented methods, known as plan-driven methods (Boehm 2002). A new approach, called the agile approach, was born with new "values"(Cockburn 2001). OntoFrame can be used as a foundation to investigate e.g. what kinds of trends and "discontinuation points" have existed, and what kinds of changes in views with regards to the IS ontology, the ISD ontology and the ISD method ontology can be identified. OntoFrame can also be used in the next upcoming "discontinuation point", whatever that might be, to analyze needs for, and to find insights into, new structures and features, as well as to elaborate them into more detailed constructs.

Fourth, we have witnessed the emergence of domain-based and ontology-based approaches in the SE/IS fields during the recent years. Domain-based development means that a common basis for understanding is first founded through defining shared concepts and vocabulary, and then upon this basis software is designed and implemented. Domain-based development is a focal issue in domain analysis (e.g. Arango *et al.* 1991; Wartik *et al.* 1992; Sutcliffe 2000), domain modeling (e.g. Kaasboll *et al.* 1996), and domain-specific modeling and languages (Tolvanen *et al.* 2003; Tolvanen *et al.* 2004). Domain-specific modeling aims to raise the level of abstraction beyond programming by specifying the solution directly using domain concepts (Luoma *et al.* 2004, 1). Ontology-based development, in turn, aims to achieve a common basis for the understanding of things in some domain (cf. domain-specific ontology), in some task (cf. task-specific ontology), or in some application (cf. application – specific ontology) (Guarino 1998). This basis can be used, for instance, in application integration (e.g. Ciocoiu *et al.* 2000) and knowledge integration through the Semantic Web (e.g. Baclawski *et al.* 2002). A domain-specific ontology is also expected to offer a natural means of representing real world knowledge for database design (Sugumaran *et al.* 2002) and deriving object-oriented frameworks and patterns (e.g. Guizzardi *et al.* 2001a; Guizzardi *et al.* 2001b). OntoFrame offers a comprehensive baseline to investigate how the domain-based approaches and the ontology-based approaches differ from one another and how to support the research into and development with those approaches.

Research in this thesis has mostly applied conceptual research methods. To obtain stronger evidence on the applicability of the design artifacts constructed, more empirical research is needed. For instance, ISD projects could be investigated, with the approach of Sabherwal *et al.* (1993) for instance, to find out how explicit the constructs included in the ISD ontology, the ISD method ontology and the ME ontology are in practice, and how they should be instantiated into project-specific methods. This would be a part of the process by which component ontologies of OntoFrame could be validated. Second, to gain experience from the use of MEMES, it should be applied in the engineering of an ISD method in practice. Because MEMES is a method skeleton, it should be first elaborated with the issues related to at least the ME datalogical perspective and the ISD datalogical perspective. Case studies can also be used to collect information about how ME workflows, not included in MEMES, are carried out in practice. Empirical studies, carried out as case studies or action research, are also needed to investigate how method engineering is organized in practice, and how those parts of ISD methods that have been recently modified in, or integrated into, the current methods are used in ISD projects.

Finally, it should be investigated what kinds of functionalities are needed in CAME (Computer Aided Method Engineering) environments to support the use of MEMES in method engineering. The current CAME and MetaCase environments (e.g. Ernst & Young 1995; Kelly *et al.* 1996; Harmsen 1997) are quite limited in this respect, because most of them address only the ISD conceptual perspective. Also those which support the engineering of ISD processes succeed in it only partly. As far as we know, MERU (Gupta *et al.* 2001) is the only CAME tool which supports the ME from more than one perspective, but also its capabilities in terms of ME perspectives and ISD perspectives are insufficient when compared to those required by the use of MEMES.

REFERENCES

Abecker, A., Bernardi, A., Hinkelmann, K., Kuhn, O. & Sintek M. 2000. Context-aware, proactive delivery of task-specific knowledge: The KnowMore Project. International Journal on Information Systems Frontiers, Vol. 2, No. 3-4, 139-162.

Ackoff, R. L. 1971. Towards a system of systems concepts. Management Science, Vol. 17, No. 11, 661-671.

Acuna, S. & Juristo, N. 2004. Assigning people to roles in software projects. Software – Practice and Experience, Vol. 34, No. 7, 675-696.

Agile Alliance 2002. Agile Manifesto [Referred on 15.3.2004]. Available at URL: <http://www.agilealliance.org>.

Ahituv, N. & Neumann, S. 1990. Principles of information systems for management. USA: Wm. C. Publishers.

Aken van, J. 1982. On the control of complex industrial organizations. Boston: Kluwer-Nijhoff Publishing.

Aktas, A.Z. 1987. Structured analysis and design of information systems. New Jersey: Prentice-Hall.

Alabiso, B. 1988. Transformation of data flow analysis models to object oriented design. In Proc. of Conf. on Object-Oriented Programming, Systems and Languages (OOPSLA'88). Special Issue of SIGPLAN Notices, Vol. 23, No. 11, 335-353.

Albano, A., Bergamini, R., Ghelli, G. & Orsini, R. 1993. An object data model with role. In R. Agrawal, S. Baker & D. Bel (Eds.) Proc. of the 19th Int. Conf. on Very Large Data Bases (VLDB'93). San Mateo: Morgan Kaufmann, 39-51.

Albert, M., Pelechano, V., Fons, J., Ruiz, M. & Pastor, O. 2003. Implementing UML association, aggregation, and composition. A particular interpretation based on a multidimensional framework. In J. Eder & M. Missikoff (Eds.) Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003). LNCS 2681, Berlin: Springer, 143-157.

Allen, J. 1984. Towards a general theory of action and time. Artificial Intelligence, Vol. 23, No. 2, 123-154.

Allen, P. & Frost S. 1998. Component-based development for enterprise systems: applying the SELECT Perspective. Cambridge: Cambridge University Press, SIGS Books.

Alston, W. 1980. Theories of meaning. In A. Lehrer & K. Lehrer (Eds.) Theory of Meaning. New Jersey: Prentice-Hall, 17-43.

Alter, S. 1996. Information systems: a management perspective. Redwood City: The Benjamin/Cummings Publishing Company.

Amberg, M. 1996. A pattern–oriented approach to a methodical evaluation of modeling methods. Australian Journal of Information Systems, Vol. 4, No. 1, 3-10.

Anda, B., Dreiem, H., Sjøberg, D. & Jørgensen, M. 2001. Estimating software development effort based on use cases – experiences from industry. In Proc. of 4th Int. Conf. on the Unified Modelling Language (UML 2001) – ”Modelling Languages, Concepts and Tools”, Toronto, Canada, October 1-5, 2001 [Referred on 20.3.2002]. Available at URL: <http ://www.idi.ntnu.no/emner/sif8080/docs/faglig/uml2001-anda.pdf>

Andersen, N., Kensing, F., Lundin, J., Mathiassen, L., Munk-Madser, A., Pasbech, M. & Sorgaard, P. 1990. Professional systems development: experiences, ideas and action. Hemel Hempstead: Prentice-Hall.

Ang, J. 1993. Performance criteria of a sound office analysis methodology. International Journal of Information Management, Vol. 13, No. 1, 51-67.

ANSI/X3 SPARC 1975. American National Standards Institute, Study Group on Data Base Management Systems, Interim Report, FDT Bulletin 7 (2).

Anthony, R. 1965. Planning and control systems: a framework for analysis. Boston, Harward University, Graduate School of Business Administration.

Aoyama, M. 1993. Concurrent development process model. IEEE Software, Vol. 10, No. 4, 46-55.

Arango, G. 1994. Domain analysis methods. In W. Schäfer, R. Prieto-Diaz & M. Matsumoto (Eds.) Software Reusablity. Chichester, England: Ellis Horwood, 17-49.

Arango, G. & Prieto-Diaz, R. 1991. Introduction and overview: domain analysis concepts and research directions. In R. Prieto-Diaz & G. Arango (Eds.) Domain Analysis and Software Systems Modeling. Los Alamitos, CA: IEEE Computer Society Press Tutorial, 9-32.

Argyris, C. & Schön, D. 1974. Theory in practice: increasing professional effectiveness. USA: Jossey-Bass.

Argyris, C. & Schön, D. 1978. Organizational learning: a theory of action perspective. Reading: Addison-Wesley.

Armenise, P., Bandinelli, S., Ghezzi, C. & Morzenti, A. 1993. A survey and assessment of software process representation formalisms. International Journal of Software Engineering and Knowledge Engineering, Vol. 3, No. 3, 410-426.

Arnauld, A. 1964. The art of thinking: Port-Royal Logic. Translated by J. Dickoff & P. James. New York: Bobbs-Merrill.

Arnesen, S. & Krogstie, J. 2002. Assessing enterprise modelling languages using a general quality framework. In T. Halpin. K. Siau, J. Krogstie (Eds.) Proc. of 7th CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD’02), Toronto, 66-77.

Arnold, P., Bodoff, S., Coleman, D., Gilchrist, H. & Hayes, F. 1991. An evaluation of five object-oriented development methods. Bristol: HP Laboratories Technical Report, HPL-91-52.

Astels, D., Miller, G. & Noval, M. 2002. A practical guide to eXtreme programming. The Coad Series, New Jersey: Prentice-Hall.

Auramäki, E., Hirschheim, R. & Lyytinen, K. 1992a. Modelling offices through discourse analysis: The SAMPO Approach. The Computer Journal, Vol. 35, No. 4, 342-352.

Auramäki, E., Hirschheim, R. & Lyytinen, K. 1992b. Modelling offices through discourse analysis: a comparison and evaluation of SAMPO and OSSAD and ICN. The Computer Journal, Vol. 35, No. 5, 492-500.

Auramäki, E., Lehtinen, E. & Lyytinen, K. 1988. A speech-act-based office modeling approach. ACM Trans. on Office Information Systems, Vol. 6, No. 2, 126-152.

Auramäki, E. & Leppänen, M. 1989. Exceptions and office information systems. In B. Pernici & A. Verrijn-Stuart (Eds.) Proc. of the IFIP WG 8.4 Working Conf. on Office Information Systems: the Design Process. Amsterdam: North-Holland, 167-182.

Austin J. 1962. How to do things with words, edited by J. Urmson & M. Sbisa. Cambridge: Harward University Press.

Avison, D. 1996. Information system development methodologies: a broader perspective. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf on Method Engineering. London: Chapman & Hall, 263-277.

Avison, D. & Fitzgerald, G. 1995a. Information systems development: methodologies, techniques and tools. 2nd ed., London: McGraw-Hill.

Avison, D. & Fitzgerald, G. 2003. Where now for development methodologies. Comm. of the ACM, Vol. 46, No. 1, 79-82.

Avison, D. & Nandhakumar, J. 1995b. The discipline of information systems: let many flowers bloom. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 1-17.

Avison, D. & Wood-Harper, T. 1990. Multiview: an exploration in information systems development. Maidenhead: McGraw-Hill.

Avison, D., Wood-Harper, A., Vidgen, R. & Wood, R. 1996. Multiview: a further exploration in information systems development. Maidenhead: McGraw-Hill.

Backlund, P., Hallenborg, C. & Hallgrimsson, G. 2003. Transfer of development process knowledge through method adaptation and implementation. In Proc. of the 11th European Conference of Information Systems (ECIS 2003), Naples.

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J. & Aronson M. 2001. Extending UML to support ontology engineering for the semantic web. In Proc. of the 4th Int. Conf. on UML (UML2001), Toronto.

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Letkowski, J. & Emery, P. 2002. Extending the Unified Modeling Language for ontology development. Software and Systems Modeling, Vol. 1, No. 2, 142-156.

Bai, G. 1998. Embryonic approach to the development of information systems. Journal of Strategic Information Systems, Vol. 6, No. 4, 299-311.

Bailey, R. 1989. Human performance engineering – using human factors/ergonomics to achieve computer system usability. 2nd edition., London: Prentice-Hall.

Bajaj, A. 2001. EOUC (Ease of use of a concept): an experimental methodology to compare the use-of-use of each concept in a conceptual model. In J. Krogstie, K. Siau & T. Halpin (Eds.) Proc. of the 6th CaiSE/IFIP8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'01), Interlaken, Switzerland.

Bajaj, A. 2002. Measuring the effect of number of concepts on the readability of conceptual models. In T. Halpin, K. Siau & J. Krogstie J. (Eds.) Proc. of 7th CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'02), Toronto, 191-202.

Bajaj, A. & Ram, S. 1999. An empirical methodology to evaluate the completeness of conceptual business process models. Journal of Information Technology, Cases and Applications, Vol. 4, No. 1, 5-30.

Bajec, M., Krisper, M. & Rupnik, R. 2004. The scenario for constructing flexible, people-focused systems development methodologies. In T. Leino, T. Saarinen & S. Klein (Eds.) Proc. of the 12th European Conference of Information Systems, The European IS Profession in the Global Networking Environment (ECIS-2004), Turku, Finland.

Baldwin, D. 1993. Applying multiple views to information systems: a preliminary framework. Data Base, Vol. 24, No. 4, 15-30.

Baldwin, J. (Ed.) 1940. Dictionary of philosophy and psychology. New York: Peter Smith.

Bandinelli, S., Fuggett, A. & Ghezzi, C. 1993. Software process model evolution in the SPADE environment. IEEE Transactions on Software Engineering, Vol. 19, No. 12, 1128-1144.

Bansler, J. & Havn, E. 1994. Information systems development with generic systems. In W. Baets (Ed.): Proc. of the Second European Conference on Information Systems. Nijenrode University, Breukelen: Nijenrode University Press, 707-715.

Barber, G. 1983. Supporting organizational problem solving with a workstation. Trans. on Office Information Systems, Vol. 1, No. 1, 45-67.

Barbic, F., Ceri, S., Bracchi, G. & Mostacci P. 1985. Modeling and integrating procedures in office information systems design. Information Systems, Vol. 10, No. 2, 149-168.

Barbier, F. & Henderson-Sellers, B. 2001. The whole-part relationship in object modeling: a definition in c01Or. Information and Software Technology, Vol. 43, No. 1, 19-39.

Bardram, J. 1998. Collaborations, coordination, and computer support. Aarhus University, Department of Computer Science, Dissertation Thesis.

Baron, R. & Beslmuller, E. (Eds.) 1989. Field Test Report, OSSAD Esprit Project No. 285, Munich.

Barron, T., Chiang, R. & Storey, V. 1999. A semiotic framework for information systems classification and development. Decision Support Systems, Vol. 25, No. 1, 1-17.

Barros, O. 1991. Modeling and evaluation of alternatives in information systems. Information Systems, Vol. 16, No. 5, 537-558.

Basili, V. & Rombach, H. 1987. Tailoring the software process to project goals and environments. In Proc. of 9th Intern. Conf. on Software Engineering, IEEE Computer Society, 345-357.

Baskerville, R. 1989. Logical controls specification: an approach to information systems secuity. In H. Klein & K. Kumar (Eds.) Proc. of the IFIP Working Conf. on Systems Development for Human Progress. Amsterdam: North-Holland, 241-255.

Baskerville, R. 1996. Structured artifacts in method engineering: the security imperative. In S. Brinkkmer, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf. on Method Enginering - Principles of Method Construction and Tool Support. London: Chapman & Hall, 8-28.

Baskerville, R. & Pries-Heje J. 2001. Racing the e-bomb: how the internet is redefining information systems development. In L. Russo, B. Fitzgerald & J. DeGross (Eds.), Proc. of IFIP TV8/WG8.2 Working Conf. on Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspectives. Amsterdam: North-Holland, 49-68.

Baskerville, R. & Pries-Heje, J. 2004. Short cycle time systems development. Information Systems Journal, Vol. 14, No. 3, 237-264.

Baskerville, R., Travis, J. & Truex, D. 1992. Systems without method: the impact of new technologies on information systems development projects. In K. Kendall, K. Lyytinen & J. DeGross (Eds.) The Impact of Computer Supported Technologies on Information Systems Development. IFIP Trans. A8. Amsterdam: North-Holland, 241-269.

Baskerville, R. & Wood-Harper, A. 1998. Diversity in information systems action research methods. European Journal of Information Systems, Vol. 7, No. 2, 90-107.

Bass, L., Clements, P. & Kazman R. 1998. Software architecture in practice. Reading: Addison-Wesley.

Batini, C., Ceri, S. & Navathe, S. 1992. Conceptual database design: an entity-relationship approach. Redwood City: Benjamin/Cummings Pub.

Benjamin, P., Menzel, C., Mayer, R., Fillion, F., Futrell, M., deWitte, P. & Lingineni, M. 1994. Information Integration for Concurrent Engineering (IICE). Ohio: Amstrong Laboratory AL/HRGA Wright-Patterson Air Force Base, IDEF5 Method Report.

Benyon, D. 1990. Information and data modelling. Oxford: Blackwell Scientific Publications.

Benyon, D. & Skidmore, S. 1987. Towards a tool kit for the systems analyst. The Computer Journal, Vol. 30, No. 1, 2-7.

Bergenti, F. & Poggi, A. 2000. Exploiting UML in the design of multi-agent systems. In A. Omicini, R. Tolksdorf & F. Zambonelli (Eds.) Engineering

Societies in the Agent World (ESAW'2000). LNCS 1972, Berlin: Springer, 106-113.

Bergheim, G., Sanders, E. & Sölvberg A. 1989. A taxonomy of concepts for the science of information systems. In E. Falkenberg & P. Lindgren (Eds.) Proc. of the IFIP TC8/WG8.1 Working Conference on Information Systems Concepts: an In-Depth Analysis. Amsterdam: North-Holland, 269-321.

Bergsten, P., Bubenko, J., Dahl, R., Gustafsson, M. & Johansson, L. 1989. Ramatic – A CASE Shell for implementation of specific CASE tools, Tempora T6.1 Report, First draft, SISU.

Bertalanffy von, L. 1968. General system theory. New York: George Braziller.

Bertels, K. & Nauta, D. 1969. Introduction to the notion of model (In Dutch: Inleiding tot het modelbegrips), De Haan, Bussum.

Bertino, E. & Guerrini, G. 1995. Objects with multiple most specific classes. In W. Olthoff (Ed.) Proc. of the 9th European Conf. on Object-Oriented Programming (ECOOP'95), LNCS 952, Berlin: Springer-Verlag, 102-126.

Bertolazzi, P., Fugini, M. & Pernici, B. 2001. Information system design based on reuse of conceptual components. In M. Rossi & K. Siua (Eds.) Information Modeling in the New Millennium. London: IDEA Group Publishing, 219-230.

Berztiss, A. 1999. Domain analysis for business software systems. Information Systems, Vol. 24, No. 7, 555-568.

Berztiss, A. & Bubenko J. 1995. A software process model for business reengineering. In A. Solvberg, J. Krogstie & A.H. Selveit (Eds.) Proc. of Conf. on Information Systems Development for Decentralised Organizations. London: Chapman & Hall, 184-200.

Beslmuller, E., Caserta, S., Conrath, D. & Dumas P. 1987. OSSAD methodology: results of the analysis phase. Munchen: OSSAD Interim Report.

Beslmuller, E., Conrath, D. & Simone C. 1986. Bridging the gap between users and vendors of office support systems. In The Commission of the European Communities (Eds.) ESPRIT'85, Part 2. Amsterdam: North-Holland, 1025-1032.

Bezivin, J. & Gerbe O. 2001. Towards a precise definition of the OMG/MDA framework. In Proc. of the 16th Annual International Conference on Automated Software Engineering (ASE 2001), Los Alamitos, California: IEEE Computer Society, 273-280.

Bielkowics, P. 2002. Evaluating information systems development methods: a new framework. In Z. Bellahsene, D. Patel & C. Rolland (Eds.) Proc. of the 8th Int. Conf. on Object-Oriented Information Systems (OOIS'2002). LNCS 2425, Berlin: Springer–Verlag, 311-322.

Bielkowicz, P. & Tun, T. 2001. A comparison and evaluation of data requirement specification techniques in SSADM and the Unified Process. In K. Dittrich, A. Geppert & M. Norrie (Eds.) Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2001). LNCS 2001, Berlin: Springer-Verlag, 46-59.

Bittner, T. 1999. On ontology and epistemology of rough location. In C. Freksa & D. Mark (Eds.) Spatial information theory - Cognitive and computational foundations of geographic information science (COSIT 99). LNCS 1661, Berlin: Springer-Verlag, 433-448.

Bittner, T. & Stell J.G. 2002. Vagueness and Rough Location. Geoinformatica, Vol. 6, No. 2, 99-121.

Bjerknes, G., Ehn, P. & Kyng, M. (Eds.) 1987. Computers and Democracy. Avebury, Aldershot, UK.

Blackmore, S. 2000. The meme machine. Oxford: Oxford University Press, 2000.

Blum, B. 1994. A taxonomy of software development methods. Comm. of the ACM, Vol. 37, No. 11, 82-94.

Bochman, A. 1990. Concerted instance-interval temporal semantics: temporal ontologies. Notre Dame Journal of Formal Logic, Vol. 31, No. 3, 403-414.

Bock, D. & Ryan, T. 1993. Accuracy in modeling with extended entity relationship and object oriented data models. Journal of Database Management, Vol. 4, No. 4, 30-39.

Bodart, F., Flory, A., Leonard, M., Rochefeld, A., Rolland, C. & Tardieu, H. 1983. Evaluation of CRIS 1 I.S. development methods using a three cycles framework. In T. Olle, H. Sol & C. Tully (Eds.) Information Systems Design Methodologies: a Feature Analysis. Amsterdam: North-Holland, 191-206.

Bodart, F., Patel, A., Sim, M. & Weber R. 2001. Should optional properties be used in conceptual modeling? A theory and three empirical tests. Information Systems Research, Vol. 12, No. 4, 384-405.

Bodger, S. 1987. Through the interface – a human activity approach to user interface design. Aarhus University, Department of Computer Science, Dissertation Thesis.

Boehm, B. 1984. Verifying and validating software requirements and design specifications. IEEE Software, Vol. 1, No. 1, 75-88.

Boehm, B. 1988. A spiral model of software development and enhancement. IEEE Computer, Vol. 21, No. 5, 61-72.

Boehm, B. 2002. Get ready for agile methods, with case. IEEE Computer, Vol. 35, No. 1, 64-69.

Boer, N.-I., van Baalen, P. & Kumar K. 2002. An activity theory approach for studying the situatedness of knowledge sharing. In Proceedings of the 35th Hawaii Intern. Conf. on Systems Sciences.

Boertien, N., Steen, M. & Jonkers, H. 2001. Evaluation of component-based development methods. In J. Krogstie, K. Siau & T. Halpin (Eds.) Proc. of 6th CaiSE/IFIP8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'01).

Bommel, P. van ter Hofstede, A. & van der Weide Th. 1991. Semantics and verification of object-role models. Information Systems, Vol. 16, No. 5, 471-495.

Booch, G. 1991. Object oriented design with applications. Redwood City: The Benjamin Cummings Publishing Co., Inc..

Booch, G., Rumbaugh, J. & Jacobson I. 1999. The Unified Modeling Language – user guide. Reading: Addison-Wesley.

Borgida, A. 1985. Features of languages for the development of information systems at the conceptual level. IEEE Software, Vol. 2, No. 1, 63-72.

Borgida, A. 1988. Modelling class hierarchies with contradictions. In H. Boral & P.-A. Larson (Eds.) Proc. of ACM SIGMOD International Conference on Management of Data, Chicaho, IL., 434-443.

Borgida, A., Mylopoulos, J. & Wong, H. 1984. Generalization / specialization. In M. Brodie, J. Mylopoulos & J. Schmidt (Eds.) On Conceptual Modelling. Berlin: Springer-Verlag, 87-114.

Borgo, S., Guarino, N. & Masolo C. 1996. Towards an ontological theory of physical objects. In Proc. of IMACS-IEEE/SMC Conference on Computational Engineering in Systems Applications (CESA'96), Symposium of Modelling, Analysis and Simulation, Lille, France, 535-540.

Borning, A. 1986. Class versus prototypes in object-orietned languages. In Proc. of the Fall Joint Computer Conference. Washington: IEEE Computer Society Press, IEEE Catalog Number 86CH2345-7, 36-40.

Bracchi, G. & Pernici, B. 1984. The design requirements of office systems. ACM Trans. on Office Information Systems, Vol. 2, No. 2, 151-170.

Brachman, R. 1983. What IS-A is and isn't: an analysis of taxonomic links of semantic networks. IEEE Computer, Vol. 16, No. 10, 30-36.

Brandt, I. 1983. A comparative study of information systems design methodologies. In T. Olle, H. Sol & C. Tully (Eds.) Information Systems Design Methodologies – A Feature Analysis. Amsterdam: North-Holland, 9 – 36.

Bratman, M. 1987. Intentions, plans, and practical reason. Cambridge: Harward University Press.

Brezillon, P., Pomerol, J.-Ch. & Saker I. 1998. Contextual and contextualized knowledge: an application in subway control. International Journal of Human-Computer Studies, Vol. 48, No. 3, 357-373.

Briand, L. & Labiche, Y. 2002. A UML-based approach to system testing. Software and Systems Modeling, Vol. 1, No.1, 10-42.

Brinkkemper, S. 1990. Formalization of information systems modeling. University of Nijmegen, Amsterdam: Thesis Publishers, Dissertation Thesis.

Brinkkemper, S. 1996. Method engineering: engineering of information systems development methods and tools. Information and Software Technology, Vol. 38, No. 4, 275-280.

Brinkkemper, S., Harmsen, F. & Oei, H. 1995. Configuration of situational process models: an information systems engineering perspective. In W. Schäfer (Ed.) Proc. of 4th European Workshop on Software Process Technology (EWSPT '95). LNCS 913, Berlin: Springer-Verlag, 193-196.

Brinkkemper, S., Saeki, M. & Harmsen, F. 1999. Meta-modelling based assembly techniques for situational method engineering. Information Systems. Vol. 24, No. 3, 209-228.

Brodie, M. 1978. The application of data types to databases. Universität Hamburg, Fachbereich für Informatik, Bericht Nr. 51.

Brodie, M. 1981. Association: a database abstraction. In P. Chen (Ed.) Entity-relationship Approach to Information Modelling and Analysis. Amsterdam: North-Holland, 583-608.

Brodie, M., Mylopoulos, J. & Schmidt J. (Eds.) 1984. On conceptual modelling. Berlin: Springer-Verlag.

Brodie, M., Rijanovic, D. & Silva, E. 1983. On a framework for information systems design methodologies. In T. Olle, H. Sol H. & C. Tully (Eds.) Information Systems Design Methodologies – A Feature Analysis. Amsterdam: North-Holland, 231-242.

Brodie, M. & Silva, E. 1982. Active and passive component modeling: ACM / PCM. In T. Olle, H. Sol & A. Verrijn-Stuart (Eds.) Information Systems Design Methodologies: a Comparative Review. Amsterdam: North-Holland, 41-92.

Brodie, R. 1996. Virus of the Mind: The New Science of the Meme. Integral Press.

Brody, B. 1980. Identity and essence. Princeton: Princeton University Press.

Bubenko, jr. J. & Lindencrona, E. 1984. Konceptuell modellering – informationanalys. Lund: Studentlitteratur.

Buckingham, R., Hirschheim, R., Land, F. & Tully C. 1987. Information systems curriculum: a basis for course design. In R. Buckinham, R. Hirschheim, F. Land & C. Tully (Eds.) Information Systems Education: Recommendations and Implementation. Cambridge: Cambridge Unversity Press.

Budde, R., Kuhlenkamp, K., Mathiassen, L. & Zullighoven H. (Eds.) 1984. Approaches to prototyping. Berlin: Springer-Verlag.

Bull, M. 1989. Systems development using structured techniques. London: Chapman & Hall Ltd.

Bunge, M. 1974. Treatise on basic philosophy, Vol. 1, Semantics I: Sense and Reference. Dortrecht: D. Reidel Publishing Company.

Bunge, M. 1977. Treatise on basic phisolophy, Vol. 3: Ontology I: The furniture of the world. Dortrecht: D. Reidel Publishing Company.

Burns, R. & Dennis, A. 1985. Selecting the appropriate application development methodology. Database, Vol. 17, No. 1, 19-23.

Burrell, G. & Morgan, G. 1979. Sociological paradigms and organizational analysis. London: Heinemann.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. 1996. Pattern-Oriented Software Architecture – A System of Patterns. New York: John Wiley & Sons.

Calway, B. 1995. Semiotic approach for object abstraction. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 234-246.

Carey, J. & Carlson, B. 2002. Framework process patterns: Lessons learned developing application frameworks. Reading: Addison-Wesley.

Carey, M. & McLeod, R. 1988. Use of system development methodology and tools. Journal of Systems Management, Vol. 39, No. 3, 30-35.

Carnap, R. 1956. Meaning and necessity. A study in semantics and modal logic. Chicago: Chicago University Press.

Carroll, J. 2003. The process of ISD methodology selection and use: a case study. In. Proc. of the 11th European Conference of Information Systems (ECIS 2003), Naples.

Carroll, J. 2004. Completing design in use: closing the appropriation cycle. In Proc. of the 12th European Conference of Information Systems (ECIS 2004): The European IS Profession in the Global Networking Environment, Turku, Finland.

Carroll, J. & Rosson, M. 1991. Deliberated evolution: stalking the view matcher in design space. Human-Computer Interaction, Vol. 6, No. ¾, 281-318.

Carvalho, J. 1999. Information systems? Which one do you mean? In E. Falkenberg, K. Lyytinen & A. Verrijn-Stuart (Eds.) Proc. of IFIP WG8.1 Int. Working Conf. on Information System Concepts: An Integrated Discipline Emerging. Dordrecht: Kluwert Academic Publishers, 259-276.

Castano, S., De Antonellis, V. 1993. Reusing process specification. In N. Prakash, C. Rolland & B. Pernici (Eds.) Proc. of the IFIP WG8.1 Working Conf. on Information System Development Process. Amsterdam: North-Holland, 267- 283.

Castano, S. & De Antonellis, V., Francalanci, C. & Pernici B. 1994. Reusability-based comparison of requirements specification methodologies. In A. Verrijn-Stuart & T. Olle (Eds.) Methods and Associated Tools for the Information Systems Life Cycle. Amsterdam: North-Holland, 63-84.

Castro, J., Kolp, M. & Mylopoulos, J. 2001. A requirements-driven development methodology. In K. Dittrich, A. Geppert & M. Norrie (Eds.) Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2001). LNCS 2001, Berlin: Springer, 108-123.

Champeaux de, D. & Faure, P. 1992. A comparative study of object-oriented analysis methods. Journal of Object-Oriented Programming, Vol. 5, No. 1, 21-33.

Chandrasekaran, B., Josephson, J. & Benjamins, R. 1998. Ontology of tasks and methods [Referred on 12.5.2001]. Available at URL <http://www.cse.ohio-state.edu/~chandra/Ontology-of-Tasks-Methods.PDF> (expanded version of those presented at the 1997 AAAI Spring Symposium and the 1998 Banff Knowledge Acquisition Workshop).

Chandrasekaran, B., Josephson, J. & Benjamins, R. 1999. What are ontologies, and why do we need them? IEEE Intelligent Systems, Vol. 14, No. 1, 20-26.

Chang, L.-h., Lin, T.-c. & Wu S. 2002. The study of information system development (ISD) process from the perspective of power development stage and organizational politics. In Proc. of the 35th Hawaii Intern. Conf. on Systems Sciences.

Charbonnel, G., Calmes, F. & Dumas, P. 1991. La Méthode OSSAD Tome 2, Guide Pratique OSSAD, Les Editions d'Organization, Paris.

Chatzoglou, P. 1997. Use of methodologies: an empirical analysis of their impact on the economics of the development process. European Journal of Information Systems, Vol. 6, No. 4, 256-270.

Chaves, A. & Carvalho, J. 1996. Expressiveness and legibility in conceptual modeling: a comparison of three techniques. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'96), Crete, Creece.

Checkland, P. 1981. Systems thinking, systems practice. Chichester: John Wiley & Sons.

Checkland P. 1988. Information systems and system thinking: time to unite? International Journal of Information Management. Vol. 8, No. 4, 239-248.

Chen, P. 1976. The entity-relationship model – toward a unified view of data. ACM Trans. on Database Systems. Vol. 1, No. 3, 9-36.

Chen, W. & Hirschheim, R. 2004. A paradigmatic and methodological examination of information systems research from 1991 to 2001. Information Systems Journal, Vol. 14, No. 3, 197-235.

Chikovsky, E. 1988. Software technology people can really use. IEEE Software, Vol. 5, No. 2, 8-10.

Chisholm, R. 1996. A realistic theory of categories – an essay on ontology. 1st edition, Cambridge: Cambridge University Press.

Chiu, D., Li, Q. & Karlapalem, K. 1999. A meta-modeling approach to workflow management systems supporting exception handling. Information Systems, Vol. 24, No. 2, 159-184.

Chomsky, N. 1966. Syntactic structures. The Hague: Mouton.

Christie, A. 1993. A graphical process definition language and its application to a maintenance project. Information and Software Technology. Vol. 25, No. 6/7, 364-374.

Chung, L., Nixon, B., Yu, E. & Mylopoulos J. 2000. Non-functional requirements in software engineering. Dordrecht: Kluwert.

Churchman, C. W. 1971. The design of inquiring systems. New York: Basic Books.

Ciborra, C. 1998. Crisis and foundation: an inquiry into the nature and limits of models and methods in the information systems discipline. Journal of Strategic Information Systems, Vol. 7, No. 1, 5-16.

Ciborra, C. 1999. A theory of information systems based on improvisation. In W. Currie & B. Galliers (Eds.) Rethinking Management Information Systems – An Interdisciplinary Perspective. Oxford: Oxford University Press, 136-155.

Cimitile, A. & Visaggio, G. 1994. A formalism for structured planning of a software project. International Journal of Software Engineering and Knowledge Engineering, Vol. 4, No. 2, 277-300.

Ciocoiu, M., Gruninger, M. & Nau D. 2000. Ontologies for integrating engineering applications. Journal of Computing and Information Science in Engineering, Vol. 1, No.1, 12-22.

Clark, H. & Carlson, T. 1981. Context for comprehension. In J. Long & A. Baddeley (Eds.) Attention and Performance, IX. Hillsdale, NJ: Erlbaum, 313-330.

Clarke, B. 1981. A calculus of individuals based on "connection". Notre Dame Journal of Formal Logic, Vol. 22, No. 3, 204-218.

Cleland, D. & King, W. 1972. Management: a systems approach. New York: McGraw-Hill.

Clifford, J. & Rao, A. 1988. A simple, general structure for temporal domains. In C. Rolland, F. Bodart & M. Leonard (Eds.) Temporal Aspects of Information Systems. Amsterdam: North-Holland, 17-28.

Cockburn, A. 2000. Selecting a project's methodology. IEEE Software, Vol. 17, No. 4, 64-71.

Cockburn, A. 2001. Agile software development. Reading: Addison-Wesley.

Codd, E. 1970. A relational model of data for large shared data banks. Comm. of the ACM, Vol. 13, No. 6, 377-387.

Codd, E. 1972. Further normalization of the data base relational model. In R. Rustin (Ed.) Data Base Systems. Englewood Cliffs: Prentice-Hall, 33-64.

Codd, E. 1979. Extending the data base relational model to capture more meaning. ACM Trans. on Database Systems, Vol. 4, No. 4, 397-434.

Colter, M. 1984. A comparative examination of system analysis techniques. MIS Quarterly, Vol. 8, No. 1, 51-66.

Conklin, J. & Begeman, M. 1988. gIBIS: a hypertext tool for exploratory policy discussion. ACM Trans. on Office Information Systems, Vol. 6, No. 4, 303-331.

Conradi, R., Fernström, C. & Fuggetta, A. 1993. A conceptual framework for evolving software processes. ACM Software Engineering Notes, Vol. 18, No. 4, 26-35.

Conrath, D., De Antonellis, V. & Simone, C. 1988. A comprehensive approach to modeling office organization and support technology. In B. Pernici & A. Verrijn-Stuart (Eds.) Proc. of the IFIP WG8.4 Working Conf. on Office Information Systems: The Design Process. Amsterdam: North-Holland, 73-92.

Conrath, D., De Antonellis, V. & Simone, C. 1992. Dynamic modeling for office support system analysis and design. In H. G. Sol & R. L. Crosslin (Eds.) Dynamic Modelling of Information Systems, II. Amsterdam: North-Holland, 75-93.

Conrath, D. & Dumas, P. (Eds.) 1989. Office support systems analysis and design (OSSAD) – a Manual, IOT, Munich.

Conrath, D. & Savolainen, V. 1999. Overview of the OSSAD methodology. In V. Savolainen (Ed.) Perspectives of Information Systems. Berlin: Springer-Verlag, 67-89.

Constantine, L. 1991. Building structured open teams to work. In Proc. of Software Development '91. San Francisco: Miller-Freeman.

Cooper, R. & Swanson, B. 1979. Management information requirements assessment: the state of the art. Data Base, Vol. 11, No. 2, 5-16.

Corcho, O., Fernandez-Lopez, M. & Gomez-Perez, A. 2003. Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering, Vol. 46, No. 1, 41-64.

Costa, H. 1999. Epistemic context, defeasible inference and conversational implicature. In P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti & F. Castellani (Eds.) Proc. of 2nd International and Interdisciplinary on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer-Verlag, 15-27.

Cotterman, W. & Kumar, K. 1989. User cube: a taxonomy of end users. Comm. of the ACM, Vol. 32, No. 11, 1313-1320.

Couger, D., Higgins, L. & McIntyre, S. 1993. (Un)structured creativity in information systems organizations. MIS Quarterly, Vol. 17, No. 4, 375-397.

Coulondre, S. & Libourel, T. 2002. Towards a new role paradigm for object-orietend modeling. In J.-M. Bruel & Z. Bellahsene (Eds.) Advances in Object-Oriented Information Systems (OOIS 2002). LNCS 2426, Berlin: Springer-Verlag, 44-52.

Cranefield, S. & Purvis, M. 1999. UML as an ontology modelling language. In Proc. of the Workshop on Intelligent Information Integration. Held in conjunction with the 16th Intern. Joint Conf. on Artificial Intelligence (IJCAI-99).

Cronholm, S. & Goldkuhl, G. 1994. Meanings and motives of method customization in CASE environments – observations and categorizations from an empirical study. In B. Theodoulidis (Ed.) Proc. of the 5th Workshop on the Next Generation of CASE Tools, University of Twente, 67-79.

Cronholm, S. & Ågerfalk, P. 1999. On the concept of method in information systems development. In T. Käkölä (Ed.) Proc. of the 22nd Information Systems Research Seminar in Scandinavia (IRIS 22): "Enterprise Architectures for Virtual Organisations", Vol. 1. University of Jyväskylä, Department of Computer Science and Information Systems, Technical Reports TR-21, 229–236.

Curtis, B. & Kellner, M. & Over, J. 1992. Process modeling. Comm. of the ACM, Vol. 35, No. 9, 75-90.

Curtis, B., Krasner, H. & Iscoe, N. 1988. A field study of the software design process for large systems. Comm. of the ACM, Vol. 31, No. 11, 1268-1287.

Cysneiros, L., Leite, J. & Neto J. 2001. A framework for integrating non-functional requirements into conceptual models. Requirements Engineering, Vol. 6, No. 2, 97-115.

Dahchour, M., Pirotte, A. & Zimanyi, E. 2002. A generic role model for dynamic objects. In A. Banks Pidduck, J. Mylopoulos, C. Woo & T. Ozsu (Eds.) Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'2002). Berlin: Springer, 643-658.

Dahl, O.-J., Myhrhaug, B. & Nygaard, K. 1968. SIMULA 67 Common Base Language. Norwegian Computer Center, Oslo.

Dahlbom, B. & Mathiassen L. 1993. Computer in context. The philosophy and practice of systems design. Cambridge: Blackwell.

Dam, K. & Winikoff, M. 2004. Comparing agent-oriented methodologies. In P. Giorgini, B. Henderson-Sellers & M. Winikoff (Eds.) Proc. of the 5th Int. Workshop on Agent-Oriented Information Systems (AIOS-2003) at AAMAS'03. LNCS 3030, Berlin: Springer-Verlag, 79-94.

Danzinger, M. & Haynes, P. 1989. Managing the CASE environment. Journal of Systems Management, Vol. 40, No. 5, 29-32.

Dart, P. & Zobel, J. 1988. Conceptual schema applied to deductive databases. Information Systems, Vol. 13, No. 3, 273-287.

Dasgupta, S. 1989. The structure of design process. In M. Yovits (Ed.) Advances in Computers. New York: Academic Press, 1-68.

Davies, I., Green, P., Milton, S. & Rosemann, M. 2003. Using Meta Models for the Comparison of Ontologies. In K. Siau, T. Halpin, J. Krogstie (Eds.) Proc. of 8th CAiSE/IFIP8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'03), 160-169.

Davis, G. 1982. Strategies for information requirements determination. IBM Systems Journal, Vol. 21, No. 1, 4-30.

Davis, G. & Olson, M. 1985. Management information systems – conceptual foundations, methods and development. 2nd edition. New York: McGraw-Hill.

Dawkins, R. 1976. The selfish gene. New York: Oxford University Press.

De, P., Sen, A. & Gudes, E. 1982. A new model for data base abstraction. Information Systems, Vol. 7, No. 1, 1-12.

De Cindio, F., De Michelis, G., Simone, C., Vassalo, R. & Zanaboni A. 1986. CHAOS as a coordinating technology. In I. Greif (Ed.) Proc. of MCC Conf. on Computer-Support for Collaborative Work (CSCW'86), 325-342.

Deiters, W. & Gruhn, V. 1994. The funsoft net approach to software process management. International Journal of Software Engineering and Knowledge Engineering, Vol. 4, No. 2, 229-256.

Demirors, O., Demirors, E., Tarhan, A. & Yildiz A. 2000. Tailoring ISO/IEC 12207 for instructional software development. In Proc. of the 26th Euromicro Conference.

Dietz, J. 1987. Modelling and specification of information systems (In Dutch: Modelleren en specificeren van informatiesystemen). Technical University of Eindhoven, Dissertation Thesis.

Dietz, J. 1992. Subject-oriented modeling of open active systems. In E. Falkenberg, C. Rolland & E. El-Sayed (Eds.) Proc. of the IFIP WG81. Conference on Information Systems Concepts: Improving the Understanding. Amsterdam: North-Holland, 227-238.

Dietz, J. 1994. Modeling business processes for the purpose of redesign. In Proc of IFIP TC8 Open Conference on Business Process Reengineering. Amsterdam: North-Holland, 249-258.

Dietz, J. 1999. Understanding and modeling business processes with DEMO. In J. Akoka, M. Bouzeghoub, I. Comyn-Wattia & E. Metais (Eds.) Proc. of the

18th Intern. Conf. on Conceptual Modeling (ER'99). Berlin: Springer-Verlag, 188-202.

Dietz, J. 2003. The atoms, molecule and fibers of organizations. Data & Knowledge Engineering, Vol. 47, No. 3, 301-325.

Dik, S. 1989. The theory of functional grammar, Part I: The structure of the clause. Functional Grammar Series, Dordrecht: Fories Publications.

Dilley, R. (Ed.) 1999. The problem of context. Berghahn Books.

Dogru, A. 2003. A process model for component-oriented software engineering. IEEE Software, Vol. 20, No. 2, 34-41.

Dominques, E., Zapata, M. & Rubio, J. 1997. A conceptual approach to meta-modelling. In A. Olive & J. Pastor (Eds.) Proc. of 9th Int. Conf. on Advanced Information Systems Engineering (CAiSE'97). Berlin: Springer, 319-332.

Dowson, M. 1987. Iteration in the software process. In Proc. of the 9th Int. Conf. on Software Engineering. New York: ACM Press, 36-39.

D'Souza, D. & Wills A. 1999. Objects, components, and frameworks with UML: The Catalysis approach. Reading: Addison-Wesley.

Dumas, P., de Petra, G. & Charbonnel, G. 1986. Towards a methodology for office analysis: Introduction and field study. Esprit #285 / OSSAD, Internal Report.

Duranti, A. & Goodwin, C. (Eds.) 1992. Rethinking context: language as an interactive phenomenon. Cambridge: Cambridge University Press.

Dutton, J. 1993. Commonsense approach to process modelling. IEEE Software, Vol. 10, No. 4, 56-64.

Ein-Dor, P. & Segev, E. 1978. Managing management information systems. Reading: Lexington.

Ein-Dor, P. & Segev, E. 1993. A classification of information systems: analysis and interpretation. Information Systems Research, Vol. 4, No. 2, 166-204.

ELEKTRA 1998. EKD User Guide. Esprit Programme 7.1, Project No. 22927.

Ellis, C. 1979. Information control nets: a mathematical model of office information flow. In P.F. Roth & G.J. Nutt (Eds.) Proc. of Conf. on Simulation, Measurement, and Modeling of Computer Systems. New York: ACM Press, 225-240.

Ellis, C. & Bernal M. 1982. OFFICETALK-D: an experimental office information systems. In Proc. of ACM SIGOA Conference on Office Systems. New York: ACM Press, 131-140.

Elmasri, R. & Navathe, S. 2000. Fundamentals of database systems. 3rd edition, Reading: Addison-Wesley.

Eloranta, K. 1974. Heuristiikat ja heuristisuus. University of Tampere, Finland. Dissertation Thesis (in Finnish).

El-Sayed, A.-Z. 1999. An autopoetic view of the concept "information system". In E. Falkenberg, K. Lyytinen & A. Verrijn-Stuart (Eds.) Proc. of IFIP WG8.1 Int. Working Conf. on Information System Concepts: An Integrated Discipline Emerging (ISCO-4).

Engeström, Y. 1987. Learning by expanding: an activity theoretical approach to developmental research. Helsinki: Orienta-Konsultit.

Engeström, Y. 1999. Activity theory and individual and social transformation. In Y. Engeström, R. Miettinen & R. Punamäki (Eds.) Perspectives on Activity Theory. Cambridge, UK: Cambridge University Press, 19-38.

Episkopou, D. 1987. The theory and practice of information systems methodologies: a grounded theory of methodology evolution. University of East Anglia, UK, Dissertation Thesis.

Ernst & Young 1995. Navigator Systems Series, Relaese 3.0 (on CD-ROM).

Essink, L. 1986. A modeling approach to information system development. In T. Olle, H. Sol & A. Verrijn-Stuart (Eds.) Proc. of the IFIP WG 8.1 Working Conf. on Comparative Review of Information Systems Design Methodologies: Improving the Practice. Amsterdam: North-Holland, 55-86.

Essink, L. 1988. A conceptual framework for information systems development methodologies. In H. J. Bullinger *et al*. (Eds.) Information Technology for Organizational Systems. Amsterdam: North-Holland, 354-362.

Euromethod 1996, The Euromethod Documentation (Version 0) [Referred on 1.2.2002]. Available at URL: <http://www.eesi.es/Euromethod/docover.html>.

Evernden, R. 1996. The information framework. IBM Systems Journal, Vol. 35, No. 1, 37-68.

Fairley, R. 1985. Software engineering concepts. New York: McGraw-Hill.

Falbo, R., de Menezes, S. & Rocha, A. R. 1998a. Using ontologies to improve knowledge integration in software engineering environments. In Proc. of 2nd World Multiconference on Systemics, Cybernetics and Informatics (SCI'98/ISAS'98), Vol. I. International Institute of Informatics and Systemics, Caracus, Venezuela, 296-304.

Falbo, R., de Menezes, S. & Rocha A. R 1998b. A systematic approach for building ontologies. In Proc. of the Sixth IberoAmerican Conf. on Artificial Intelligence (IBERAMIA'98), Lisbon, Portugal, 349-360.

Falkenberg, E. 1976. Significations: the key to unify data base management. Information Systems, Vol. 2, No. 1, 19-28.

Falkenberg, E:, Hesse, W., Lindgreen, P., Nilsson, B., Oei, J. L. H., Rolland, C., Stamper, R., van Asche, F., Verrijn-Stuart, A. & Voss, K. 1998. A framework of information system concepts, The FRISCO Report (Web edition), IFIP.

Falkenberg, E., Nijjsen, G., Adams, A., Bradley, L., Bugeia, P., Campbell, A., Carkeet, M., Lehman, G. & Shoesmith, A. 1983. Feature analysis of ACM/PCM, CIAM, ISAC and NIAM. In T. Olle, H. Sol & C. Tully (Eds.) Information Systems Design Methodologies – A Feature Analysis. Amsterdam: North-Holland, 169- 190.

Falkenberg, E., Oei, J. & Proper, H. 1992a. A conceptual framework for evolving information systems. In H. Sol & R. Crosslin (Eds.) Dynamic Modelling of Information Systems II. Amsterdam: North-Holland, 353-375.

Falkenberg, E., Oei, J. & Proper, H. 1992b. Evolving information systems: beyond temporal information systems. In A. Tjoa & I. Ramos (Eds.) Proceedings of the Data Base and Expert Systems Applications Conference (DEXA'92). Berlin: Springer Verlag, 282-287.

Fayad, M. & Schmidt, D. 1997. Object-oriented application frameworks. Comm. of the ACM, Vol. 40, No. 10, 32-38.

Feibleman, J. 1951. Ontology. Baltimore: The Johns Hopkins Press.

Fensel, D., Motta, E., van Harmelen, F., Benjamins, R., Crubezy, M., Decker, S., Gaspari, M., Groenboom, R., Grosso, W., Musen, M., Plaza, E., Schreiber, G., Studer, R. & Wielinga, B. 2003. The unified problem-solving method development language UPML. Knowledge and Information Systems, Vol. 5, No. 1, 83-131.

Fernandez-Lopez, M., Gomez-Perez, A., Pazos-Sierra, A. & Pazos-Sierra, J. 1999. Building a chemical ontology using METONTOLOGY and the ontology design environment. IEEE Intelligent Systems & Theory Applications, Vol. 4, No. 1, 37-46.

Fickas, S. 1985. Automating the transformational development of software. IEEE Trans. on Software Engineering, Vol. 11, No. 11, 1268- 277.

Fife, D. 1987. How to know a well-organized software project when you find one. In R. Thayer (Ed.) Tutorial: Software Engineering Project Management. Washington: IEEE Computer Society Press, 268-276.

Fillmore, C. 1968. The case for case. In E. Bach & R. T. Harms (Eds.) Universals in Linguistic Theory. New York: Holt, Rinehart and Winston, 1-88.

Finkelstein, A. & Fuks H. 1988. A cooperative framework for program development. Information and Software Technology, Vol. 30, No. 8, 467-476.

Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J. & Nuseibeh, B. 1994. Inconsistency handling in multi-perspective specifications. Trans. on Software Engineering, Vol. 20, No. 8, 569-578.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. & Goedicke, M. 1992. Viewpoints: A framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering, Vol. 1, No. 2, 31-58.

Firesmith, D. 2002. Requirements engineering. Journal or Object Technology, Vol. 1, No. 4, 93-103.

Firesmith, D. 2003a. Specifying good requirements. Journal of Object Technology, Vol. 2, No. 4, 77-87.

Firesmith, D. 2003b. Using quality models to engineer quality requirements. Journal of Object Technology, Vol. 2, No. 5, 67-75.

Firesmith, D. 2004. Creating a project-specific requirements engineering process. Journal of Object technology, Vol. 3, No. 5, 31-44.

Firesmith, D. & Henderson-Sellers B. 1999. Improvements to the OPEN process metamodel. Journal of Object-Oriented Programming, Vol. 12, No. 7, 30-35.

Firesmith, D., Henderson-Sellers, B. & Graham, I. 1997. OPEN Modeling Language (OML) Reference Manual. New York: SIG Books.

Fitzgerald, B. 1991. Validating new information systems techniques: a retrospective analysis. In H.-E. Nissen, H. Klein & R. Hirscheim (Eds.) Information Systems Research: Comtemporary Approaches and Emergent Traditions. Amsterdam: North-Holland, 657-672.

Fitzgerald, B. 1994. Whither systems development: time to move the lamppost. In C. Lissoni, T. Richardson, R. Miles, A. Wood-Harper & N. Jayaratna (Eds.) Proc. of the 2nd BCS Conf. on Information Systems Methodologies. Swindon, UK: BCS Publications, 371-380.

Fitzgerald, B. 1996a. A field study of the usage of systems development methodologies. University College, Cork, Ireland, ESRC Research and Discussion Papers Ref. 8/96a.

Fitzgerald, B. 1996b. Formalized systems development methodologies: a critical perspective. Information Systems Journal, Vol. 6, No. 1, 3-23.

Fitzgerald, B. 1997. Systems development methodologies: a need for new canons. ESRC Research and Discussion Papers Ref 01/97, University College, Cork, Ireland.

Fitzgerald, B. 1998a. An empirical investigation into the adoption of systems development methodologies. Information & Management, Vol. 34, No. 6, 317-328.

Fitzgerald, B. 1998b. An empirically-grounded framework for the information systems development process. In R. Hirschheim, M. Newman & J. deGross (Eds.) Proc. of the 19th Int. Conf. in Information Systems. USA: Omnipress, 103-114.

Fitzgerald, B. & Fitzgerald, G. 1998. Alternative paradigms for information systems development: from old to new. University College Cork, Ireland, ESRC Research and Discussion Papers, Paper Ref. 1/98.

Fitzgerald, B., Russo, N. & O'Kane, T. 2003. Software development method tailoring at Motorola. Comm. of the ACM, Vol. 46, No. 4, 65-70.

Fitzgerald, B., Russo, N. & Stolterman, E. 2002. Information systems development – methods in action. London: McGraw Hill.

Fitzgerald, G., Stokes, N. & Wood, J. 1985. Feature analysis of contemporary information systems methodologies. The Computer Journal, Vol. 28, No. 3, 223-230.

Floyd, C. 1984. A systematic look at prototyping. In R. Budde, K. Kuhlenkam, L. Mathiassen & H. Zullighowen (Eds.) Approches to Prototyping. Berlin: Springer-Verlag, 1-18.

Floyd, C. 1986. A comparative evaluation of system development methods. In T. Olle, H. Sol & A. Verrijn-Stuart (Eds.) Information Systems Design Methodologies: Improving the Practice. Amsterdam: North-Holland, 19-54.

Floyd, C. 1987. Outline of a paradigm change in software engineering. In G. Bjerknes, P. Ehn & M. Kyng (Eds.) Computers and Democracy – a Scandinavian Challenge. Brookfield: Avebury Gower Pub., 193-210.

Flynn, D. & Fragoso-Diaz, O. 1993. Conceptual Euromodelling: how do SSADM and MERISE compare? European Journal of Information Systems, Vol. 2, No. 3, 169-183.

Forsell, M., Halttunen, V. & Ahonen, J. 2000. Use and identification of components in component-based software development methods. In W. Frakes (Ed.) Proc. of the 6th Int. Conf. on Software Reuse: Advances in Software Reusability (ICSR-6). LNCS 1844, Berlin: Springer, 284-301.

Fox, M. 1992. The TOVE Project: a common-sense model of the enterprise. In F. Belli & F. Radermacher (Eds.) Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. LNAI 604, Berlin: Springer-Verlag, 25-34.

Fox, M. 1998. Enterprise modeling – artificial intelligence. AI Magazine, Vol. 19, No. 3, 109-121.

Franckson, M. 1994. The Euromethod deliverable model and its contributions to the objectives of Euromethod. In A. Verrijn-Stuart & T. Olle (Eds.) Methods and Associated Tools for the Information Systems Life Cycle. Amsterdam: North-Holland, 131-150.

Frank, U. 2002. Multi-perspective enterprise modeling (MEMO) – conceptual framework and modeling language. In Proc. of the 35th Hawaii International Conference on Systems Sciences.

Frankel, D. S. 2003. Model driven architecture applying MDA to enterprise computing. OMG Press Books.

Freeman, P. 1987. Software perspectives: The system is the message. Reading: Addison-Wesley.

Freeman, M. & Layzell, P. 1994. A meta-model of information systems to support reverse engineering. Information and Software Technology, Vol. 36, No. 5, 283-294.

Freundschuh, S. & Egenhofer, M. 1997. Human conceptions of spaces: implications for GIS. Transactions in GIS, Vol. 2, No. 4, 361-375.

Furtado, A. & Neuhold, E. 1986. Formal techniques for data base design. Berlin: Springer-Verlag.

Galbraith, J. 1973. Designing complex organizations. Reading: Addision-Wesley.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. 1995. Design patterns: elements of reusable object-oriented software. Reading: Addison-Wesley.

Gane, C. & Sarson T. 1979. Structured systems analysis: tools and techniques. Englewood Cliffs: Prentice-Hall.

Garner, B. & Raban, R. 1999. Context management in modeling information systems (IS). Information and Software Technology, Vol. 41, No. 14, 957-961.

Gasser, L. 1986. The integration of computing and routine work. ACM Trans. on Office Information Systems, Vol. 4, No. 3, 205-225.

Gemino, A. & Wand, Y. 2002. Towards common dimensions in empirical comparisons of conceptual modelling techniques. In T. Halpin, K. Siau & J. Krogstie (Eds.) Proc. of 7th CaiSE/IFIP WG8.1 International Workshop on

Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'02), 144-151.

Gerstl, P. & Pribbenow, S. 1996. A conceptual theory of part-whole relations and its applications. Data & Knowledge Engineering, Vol. 20, No. 2, 305-322.

Ghidini, C. & Serafini, L. 1999. A context-bsed logic for distributed knowledge representation and reasoning. In P. Bouque, L. Serafini, P. Brezillon, M. Benerecetti & F. Castellani (Eds.) Proc. of Second International and Interdisciplinary Conf. on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer Verlag, 157-172.

Giddens, A. 1984. The constitution of society. Cambridge: Polity Press.

Gigch van, J. 1991. System design modeling and metamodeling. New York: Plenum Press.

Glasson, B. 1986. Supporting controlled variety in systems development environments. In T. Olle, H. Sol & A. Verrijn-Stuart (Eds.) Information Systems Design Methodologies: Improving the Practice. Amsterdam: North-Holland, 271-288.

Glasson, B. 1989. Model of system evolution. Information and Software Technology, Vol. 31, No. 7, 351-356.

Godwin, A., Gleeson, J. & Gwillian, D. 1989a. An assessment of the IDEF notations as descriptive tools. Information Systems, Vol. 14, No. 1, 13-28.

Godwin, A., Gore, M. & Salt, D. 1989b. A comparison of JSD and DFD as descriptive tools. The Computer Journal. Vol. 32, No. 3, 202 – 211.

Goldkuhl, G. 1991. Information systems design as argumentation – an investigation into design rationale as a conceptualization of design. In K. Ivanov (Ed.) Proc. of the 14th Information Systems Research Seminar in Scandinavia (IRIS'1991), Umeå.

Goldkuhl, G. & Cronholm, S. 1993. Customizable CASE environments: a framework for design and evaluation. Institutionen for Datavetenskap, Universitetet och Tekniska Högskolan, Linköping, Research Report.

Goldkuhl, G., Lind, M. & Seigerroth, U. 1998. Method integration as a learning process. In N. Jayaratna, B. Fitzgerald, T. Wood-Harper & J.-M. Larasquet (Eds.) Training and Education of Methodological Practitioners and Researchers. Berlin: Springer-Verlag, 113-118.

Goldkuhl, G. & Röstling, A. 1988. Förändringsanalysi – arbetsmetodik och förhållningssätt för goda förändringsbelust. Lund: Studentlitterature.

Goldstein, R. & Storey, V. 1999. Data abstraction: Why and how? Data & Knowledge Engineering, Vol. 29, No. 3, 293-311.

Gomaa, H. & Scott, D. 1981. Prototyping as a tool in the specification of user specifications. In Proc. of the 5th IEEE International Conference on Software Engineering. IEEE Computer Society, 333-342.

Gomez, C. & Olive A. 2002. Evolving partitions in conceptual schemas in the UML. In A. Banks Pidduck, J. Mylopoulos, C. Woo & T. Ozsu (Eds.) Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'2002). Berlin: Springer, 467-483.

Gomez-Perez, A. 1995. Some ideas and examples to evaluate ontologies. In Proc. of the 11th Conf. on Artifical Intelligence for Applications, 299-305.

Gorry, G.A. & Scott-Morton, M. 1971. A framework for management information systems. Sloan Management Review, Vol. 13, No. 1, 55-57.

Gottlob, G., Schrefl, M. & Röck B. 1996. Extending object-oriented systems with roles. ACM Trans. on Office Information Systems, Vol. 14, No. 3, 268-296.

Graham, D. 1989. Incremental development: review of nonmonolithic life-cycle development models. Information and Software Technology, Vol. 31, No. 1, 7-20.

Graham, I. 1995. Migrating to object technology. Reading: Addison-Wesley.

Graham, I., Henderson-Sellers, B. & Younessi, H. 1997. The OPEN process specification. Reading: Addison-Wesley.

Grant, D. & Ngwenyama, O. 2003. A report on the use of action reseach to evaluate a manufacturing information systems development methodology in a company. Information Systems Journal, Vol. 13, No. 1, 21-35.

Gregor, S. 2002. Design theory in information systems. Australian Journal of Information Systems, Vol. 9, Special Issue, 14-22.

Gregor, S. & Jones, D. 2003. The formulation of design theories for information systems. In Proc. of 12th Int. Conf. on Information Systems Development (ISD'03).

Green, P. 1997. Use of information systems analysis and design (ISAD) grammars in combination in upper CASE tools – an ontological evaluation. In K. Siau, Y. Wand & J. Parson (Eds.) Proc. of the Second CAiSE/IFIP8.1 Intern. Workshop on the Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'97), 1-12.

Green, P. & Rosemann M. 2000. Integrated process modeling: an ontological evaluation. Information Systems, Vol. 25, No. 2, 73-87.

Griethuysen van, J. (Ed.) 1982. Concepts and terminology for the conceptual schema and the information base. ISO/TC95 Computers and Information Processing, New York, ISO/TC97/SC5/WG3.

Grosz, G., Rolland, C., Schwer, S., Souveyet, C., Plihon, V., Si-Said, S., Achour, C. & Gnaho, C. 1997. Modelling and engineering the requirements engineering process: an overview of the NATURE approach. Requirements Engineering, Vol. 2, No. 2, 115-131.

Gruber, T. 1993. A translation approach to portable ontology specification. Knowledge Acquisition, Vol. 5, No. 2, 119-220.

Gruber, T. 1995. Towards principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies, Vol. 43, No. 5/6, 907-928.

Grundy, J. & Venable, J. 1996. Towards an integrated environment for method engineering. In K. Lyytinen, S. Brinkkemper & R. Welke (Eds.) Proc. of the IFIP TC8, WG 8.1/8.2 Working Conference on Method Engineering – Principles of Method Constrcution and Tool Support. London: Chapman & Hall, 45-62.

Gruninger, M. & Fox, M. 1995. Methodology for the design and evaluation of ontologies. In Proc. of Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95).

Guarino, N. 1997. Understanding, building and using ontologies. International Journal of Human-Computer Studies, Vol. 46, No. 2/3, 293-310.

Guarino, N. 1998. Formal ontology and information systems. In N. Guarino (Ed.) Proc. of Conf. on Formal Ontology in Information Systems (FOIS'98). Amsterdam: IOS Press, 3-15.

Guarino, N., Carrara, M. & Giaretta, P. 1995. Ontologies and knowledge bases: towards a terminological clarification. In N. Mars (Ed.) Towards Very Large Knowledge Bases, Knowledge Building and Knowledge Sharing. Amsterdam: IOS Press, 25-32.

Guarino, N., Pribbenow, S. & Vieu, L. 1996. Modeling parts and wholes. Data and Knowledge Engineering, Special Issue, Vol. 20, No. 3, 257-258.

Guarino, N. & Welty, C. 2000. Towards a methodology for ontology based model engineering. In J. Bezevin & J. Ernst (Eds.) Proc. of the First Int. Workshop on Model Engineering (IWME-2000).

Guizzardi, G., Falbo, R. & Filho, J. 2001a. From domain ontologies to object-oriented frameworks. In G. Stumme, A. Maedche & S. Staab (Eds.) Proc. of Workshop on Ontologies (ONTO'2001), 1-14..

Guizzardi, G., Falbo, R. & Goncalves, J. 2001b. Using framework and patterns to implement domain ontologies. In C. Werner & A. Russo (Eds.) Proc. of the 16th Brazilian Symposium on Software Engineering (SBES'2001), 36-51.

Gupta, D. & Prakash, N. 2001. Engineering methods from method requirements specifications. Requirements Engineering, Vol. 6, No. 3, 135-160.

Gustafsson, M., Karlsson, T. & Bubenko, J. 1982. A declarative approach to conceptual information modeling. In T. Olle, H. Sol & A. Verrij-Stuart (Eds.) Information Systems Design Methodologies: a Comparative Review. Amsterdam: North-Holland, 93-142.

Guttag, J. 1977. Abstract data types and the development of data structures. Comm. of the ACM, Vol. 20, No. 6, 396-404.

Habermas, J. 1984. The theory of communicative action: reason and the rationalization of society. Boston: Beacon Press, Vol. 1.

Hackathorn, R. & Karimi, J. 1988. A framework for comparing information engineering methods. MIS Quarterly, Vol. 12, No. 2, 203-220.

Halliday, M. 1978. Language as social semiotic: the social interpretation of meaning. London: Edwards Arnold.

Halpin, T. 1998. ORM/NIAM Object-Role Modelling. In P. Bernus, K. Mertins & G. Schmidt (Eds.) Handbook on Information Systems Architecture. Berlin: Springer-Verlag, 81-101.

Hammer, M. & McLeod, D. 1981. Database description with SDM: a semantic database model. ACM Trans. on Database Systems, Vol. 6, No. 3, 351-386.

Hardgrave, B. & Dalal, N. 1995. Comparing object oriented and extended entity relationship models. Journal of Database Management, Vol. 6, No. 3, 15-21.

Hardy, C. Thompson, B. & Edwards, H. 1995. The use, limitations and customization of structured systems development methods in the UK. Information and Software Technology, Vol. 37, No. 9, 467-477.

Harmsen, F. 1997. Situational method engineering. University of Twente, Moret Ernst & Young Management Consultants, The Netherlands, Dissertation Thesis.

Harmsen, F., Brinkkemper, S. & Oei, J. 1994. Situational method engineering for information systems project approach. In H. Olle & A. Verrijn–Stuart (Eds.) Proc. of the IFIP WG 8.1. Conf. on Methods and Associated Tools for Information Life Cycle (CRIS-94). IFIP Transactions A-55, Amsterdam: North-Holland, 169-194.

Harmsen, F., Lubbers, I. & Wijers, G. 1995. Success-driven selection of fragments for situational methods: The S^3 model. In K. Pohl & P. Peters (Eds.) Proc. of the Second Int. Workshop on Requirements Engineering: Foudations of Software Quality (REFSQ'95). Aachen: Aachener Beiträge zur Informatik, Band, 104-115.

Harmsen, F. & Saeki, M. 1996. Comparison of four method engineering languages. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf on Method Engineering. London: Chapman & Hall, 209-231.

Hautamäki, A. 1986. Points of views and their logical analysis. Helsinki: Acta Philosophica Fennica, Vol. 41.

Hayes, P. 1995. A catalog of temporal theories. University of Illinois: Technical Report UIUC-BI-AI-96-01.

Hazeyama, A. & Komiya S. 1993. Software process management system supporting the cooperation between manager and developers. In S. Brinkkemper & F. Harmsen (Eds.) Proc. of the Fourth Workshop on the Next Generation of CASE Tools. Memoranda Informatica 93-32, University of Twente, The Netherlands, 183-188.

Hedberg, B. 1980. Using computerized information systems to design better organizations and jobs. In N. Björn-Andersen (Ed.) The Human Side of Information Processing. Amsterdam: North-Holland, 19-37.

Heijst van, G., Schreiber, A. & Wielinga B. 1997. Using explicit ontologies in KBS development. International Journal of Human and Computer Studies, Vol. 46, No. 2-3, 293-310.

Heineman, G.T., Botsford, J.E., Caldiera, G., Kaiser, G., Kellner, M. & Madhavji, N. 1994. Emerging technologies that support a software process life cycle. IBM Systems Journal, Vol. 33, No. 3, 501-529.

Heiskanen, A. 1995. Reflecting over a practice – Framing issues for scholar understanding. Information Technology & People, Vol. 8, No. 4, 3-18.

Heiskanen A. 2005. Control, trust, and the dynamics of information system outsourcing: a case study of contractual software development. Department of Information Processing Science, University of Oulu, submitted to be published.

Heller, F. 1991. Participation and competence: a necessary relationship. In R. Russel & V. Rus (Eds.) International Handbook of Participation in Organizations, 265-281.

Henderson-Sellers, B. 1992. A book of object-oriented knowledge: Object-oriented analysis, design, and implementation: A new approach to software engineering. Englewood-Cliffs: Prentice-Hall.

Henderson-Sellers, B. 1999. A methodological metamodel of process. Journal of Object-Oriented Programming, Vol. 11, No. 9, 56-63.

Henderson-Sellers, B. 2003. Method engineering for OO systems development. Comm. of the ACM, Vol. 46, No. 10, 73-78.

Henderson-Sellers, B. & Barbier, F. 1999a. What is this thing called aggregation? In R. Mitchell, A. C. Wills, J. Bosch & B. MeyerProc. (Eds.) Proc. of TOOLS EUROPE'99. Silver Spring, MD: IEEE Computer Society Press, 236-250.

Henderson-Sellers, B., Collins, G., Due, R. & Graham, I. 2001. A qualitative comparison of two processes for object-oriented software development. Information and Software Technology, Vol. 43, No. 12, 705-724.

Henderson-Sellers, B. & Edwards, J. 1993. The O-O methodology for the object oriented life cycle. Software Engineering Notes, Vol. 18, No. 4, 54-60.

Henderson-Sellers, B. & Edwards, J. 1995. Book Two of Object-Oriented Knowledge. The Working Object. Sydney: Prentice-Hall.

Henderson-Sellers, B. & Firesmith, D. 1999b. Comparing OPEN and UML: the two third-generation OO development approaches. Information and Software Technology, Vol. 41, No. 3, 139-156.

Henderson-Sellers, B. & Mellor, S. 1999c. Tailoring process-focused OO methods. Journal of Object-Oriented Programming, Vol. 12, No. 4, 40-45.

Hendrix, G. 1979. Encoding knowledge in partitioned networks. In N. Findler. (Ed.) Associative Networks: Representation and Use of Knowledge by Computers. New York: Academic Press, 51-92.

Herbst, H. 1995. A meta-model for business rules in systems analysis. In J. Iivari, K. Lyytinen & M. Rossi (Eds.) Advanced Information Systems Engineering. LNCS 932, Berlin: Springer, 186-199.

Herbst, H., Knolmayer, T. & Schlesinger, M. 1994. The specification of business rules: a comparison of selected methodologies. In A. Verrijn-Stuart A. & T. Olle (Eds.) Methods and Associated Tools for the Information Systems Life Cycle. Amsterdam: North-Holland, 29-46.

Herbst, H.& Myrach, T. 1997. A repository system for business rules. In R. Meersman & Mark L. (Eds.) Proc. of the 6th IFIP TC-2 Working Conf. on Data Application Semantics. London: Chapman & Hall, 119-138.

Hevner, A., March, S., Park J. & Ram S. 2004. Design science in information systems research. MIS Quarterly, Vol. 28, No. 1, 75-105.

Heym, M. & Österle, H. 1992a. A reference model for information systems development. In K. Kendall, K. Lyytinen & J. DeGross (Eds.) Proc. of the IFIP WG 8.2 Working Conference on the Impacts on Computer Supported Technologies on Information Systems Development. Amsterdam: North-Holland, 215-240.

Heym M. & Österle H. 1992b. A semantic data model for methodology engineering. In G. Forte & N. Madhavji (Eds.) Proc. of the Fifth CASE '92 Workshop. Los Alamitos, CA: IEEE Computer Society Press, 215-239.

Hicks, J. 1993. Management information systems: a user perspective. St. Paul: West Publishing Company.

Hidding, G. 1997. Reinventing methodology: Who reads it and why? Comm. of the ACM, Vol. 40, No. 11, 102-109.

Hidding, G., Freund, G. & Joseph, J. 1993. Modeling large processes with task packages. In Proc. of Workshop on Modeling in the Large, AAAI Conference, Washington, DC.

Hillegersberg van, J. & Kumar, K. 1999. Using metamodeling to integrate object-oriented analysis, design and programming concepts. Information Systems, Vol. 24, No. 2, 113-129.

Hirschheim, R. 1986. Understanding the office: a social analytic perspective. Trans. on Office Information Systems, Vol. 4, No. 3, 331-344.

Hirschheim, R. & Klein, H. 1989. Four paradigms of information systems development. Comm. of the ACM, Vol. 32, No. 10, 1199-1216.

Hirschheim, R. & Klein, H. 1992a. Paradigmatic influences on information systems development methodologies evolution and conceptual advances. In M. Yovits (Ed.) Advances in Computers. Vol. 33. New York: Academic Press, 293-392.

Hirschheim, R. & Klein, H. 1992b. A research agenda for future information systems development methodologies. In W. Cotterman & J. Senn (Eds.) Challenges and Strategies for Research in Systems Development. New York: John Wiley & Sons Ltd., 235-253.

Hirschheim, R., Klein, H. & Lyytinen, K. 1995. Information systems development – conceptual and philosophical foundations. Cambridge: Cambridge University Press.

Hoare, C.A. & Wirth, N. 1973. An axiomatic definition of the programming language PASCAL. Acta Informatica, Vol. 2, No. 4, 335-355.

Hoc, J.-M. 1988. Cognitive psychology of planning. London: Academic Press.

Hofstede ter, A. & Proper, H. 1998. How to formalize it? Formalization principles for information system development methods. Information and Software Technology, Vol. 40, No. 10, 519-540.

Hofstede ter, A., Proper, H. & Weide van der, Th. 1993a. Formal definition of a conceptual language for the description and manipulation of information models. Information Systems, Vol. 18, No. 7, 489-523.

Hofstede ter, A. & Verhoef, T. 1997. On the feasibility of situational method engineering. Information Systems, Vol. 22, No. 6/7, 401-422.

Hofstede ter, A. & Weide van der, Th. 1992. Formalisation of techniques: chopping dow the methodology jungle. Information and Software Technology, Vol. 34, No. 1, 57-65.

Hofstede ter, A. & Weide van der, Th. 1993b. Expressiveness in conceptual data modeling. Data & Knowledge Engineering, Vol. 10, No. 1, 65-100.

Holm, P. & Karlgren, K. 1995. Theories of meaning and different perspectives on information systems. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 20-32.

Holt, J. 1997. Current practice in software engineering - A survey. Computing and Control Engineering Journal, Vol. 8, No. 4, 167-172.

Hommes, B.-J. & van Reijswound, V. 1999. The quality of business process modeling methods – Illustration of a framework for understanding modeling quality In E. Falkenberg, K. Lyytinen & A. Verrijn-Stuart (Eds.) Proc. of IFIP WG8.1 Int. Working Conf. on Information System Concepts: An Integrated Discipline Emerging. Amsterdam: North-Holland, 117-126.

Hommes, B.-J. & van Reijswound V. 2000. Assessing the quality of business process modeling techniques. In Proc. of the 33rd Hawaii International Conf. on Systems Science.

Hong, S., Goor van der, & Brinkkemper S. 1993. A formal approach to the comparison of object-oriented analysis and design methodologies. In F. Nunamaker & R. Sprague (Eds.) Proc. of 26th Hawaii Int. Conf. on Systems Sciences, Vol. IV, 689-698.

Howard, G., Bodnovich, T., Janicki, T., Klein, S., Albert, P. & Cannon, D. 1999. The efficacy of matching information systems development methodologies with application characteristics – an empirical study. The Journal of Systems and Software, Vol. 45, 177-195.

Hruby, P. 2000a. Structuring software development artifacts with UML. Journal of Object-Oriented Programming, Vol 12, No. 9, 22-33.

Hruby, P. 2000b. Designing customizable methodologies. Journal of Object-Oriented Programming, Vol. 13, No. 8, 22- 31.

Hughes, J. & Reviron, E. 1996. Selection and evaluation of information system development methodologies: the gap between the theory and practice. In N. Jayaratna & B. Fitzgerald (Eds.) Proc. of the 4th Conf. on Information Systems Methodologies: Lessons Learned from the Use of Methodologies. British Computer Society, 309-319.

Hull, R. & King, R. 1987. Semantic database modeling survey, Applications and research issues. ACM Computing Surveys, Vol. 19, No. 3, 210-260.

Hutching, T., Hyde, M., Marca, D. & Cohen L. 1993. Process improvement that lasts: an integrated training and consulting method. Comm. of the ACM, Vol. 36, No. 10, 105-113.

IBM 1984. Business systems planning, Information Systems Planning Guide, GE20-0527-4, Atlanta: IBM Corporation.

IEEE 1990. Standard Glossary of software engineering terminology. IEEE Standard 610.12-1990.

IEEE 1991. IEEE Std. 830-1984, In IEEE Software Engineering Standards Collection. New York: IEEE.

Iivari, J. 1978. Pragmatic control of the development of data processing function, Department of Data Processing Science, University of Oulu, Licentitate Thesis (in Finnish).

Iivari, J. 1982. Taxonomy of the experimental and evolutionary approaches to systemeering. In J. Hawgood (Ed.) Evolutionary Information Systems. Amsterdam: North-Holland, 101-119.

Iivari, J. 1983. Contributions to the theoretical foundations of systemeering research and the PIOCO model. Acta Universitatis Ouluensis, A150, University of Oulu, Oulu, Finland, Dissertation Thesis.

Iivari, J. 1989a. Levels of abstraction as a conceptual framework for an information system. In E. Falkenberg & P. Lindgren (Eds.) Information System Concepts: An In-Depth Analysis. Amsterdam: North–Holland, 323-352.

Iivari, J. 1989b. A methodology for IS development as organizational change: a pragmatic contingency approach. In H. Klein & K. Kumar (Eds.) Systems Development for Human Progress. Amsterdam: North-Holland, 197-217.

Iivari, J. 1990a. Hierarchical spiral model for information system and software development. Part 1: Theoretical background. Information and Software Technology, Vol. 32, No. 6, 386-399.

Iivari, J. 1990b. Hierarchical spiral model for information system and software development. Part 2: Design process. Information and Software Technology, Vol. 32, No. 7, 450-458.

Iivari, J. 1991. A paradigmatic analysis of contemporary schools of IS development. European Journal of Information Systems, Vol. 1, No. 4, 249-272.

Iivari, J. 1992. Relationships, aggregations and complex objects. In S. Ohsuga, H. Kangassalo, H. Jaakkola., K. Hori & N. Yonezaki (Eds.) Proc. of European-Japanese Conference Information Modelling and Knowledge Bases III: Foundations, Theory, and Applications. Amsterdam: IOS Press, 141-159.

Iivari, J. 1994. Object-oriented information systems analysis: a comparison of six object-oriented analysis methods. In T. Olle & A. Verrijn-Stuart (Eds.) Proc. of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle (CRIS'94). IFIP Transactions A-55, Amsterdam: North-Holland, 85-110.

Iivari, J. 2003. Towards information systems as a science of meta-artifacts. Comm. of the Association of Information Systems, Vol. 12, Article 37, 568-581.

Iivari, J., Hirschheim, R. & Klein, H. 1998a. A paradigmatic analysis of contrasting IS development approaches and methodologies. Information Systems Research, Vol. 9, No. 2, 164-193.

Iivari, J., Hirschheim, R. & Klein, H. 2001. A dynamic framework for classifying information systems development methodologies and approaches. Journal of Management Information Systems, Vol. 17, No. 3, 179-218.

Iivari, J., Hirschheim, R. & Klein, H. 2004. Towards a distinctive body of knowledge for information systems experts: coding ISD process knowledge in two IS journals. Information Systems Journal, Vol. 14, No. 4, 313-342.

Iivari, J. & Kerola, P. 1983. A sociocybernetic framework for the feature analysis of information systems design methodologies. In T. Olle, H. Sol & C. Tully (Eds.) Proc. of the IFIP WG 8.1 Working Conf. on Feature Analysis of Information Systems Development Methodologies. Amsterdam: North-Holland, 87-139.

Iivari, J. & Koskela, E. 1987. The PIOCO Model for information systems design. MIS Quarterly, Vol. 11, No. 3, 410-419.

Iivari, J. & Linger H. 1999. Knowledge work as collaborative work: a situated activity theory view. In Proc. of the 32nd Hawaii International Conf. on System Sciences (HICSS-32). Washington: IEEE Computer Society.

Iivari, J. & Maansaari, J. 1998b. The usage of system development methods: Are we stuck to old practices? Information and Software Technologies, Vol. 40, No. 9, 501-510.

ISO 1984. Open Systems Interconnections (OSI) – basic reference model. ISO 7498/TC97/SC21 Information Processing Systems.

ISO 1991. ISO 9000-3, Quality management and quality assurance standards, Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software.

ISO 1990. International Standard. Information Resource Dictionary System (IRDS) – Framework ISO/IEC 10027.

ISO 1996. Information Technology – Open Distributed Processing - Reference Model: Overview, 10746-1.

Jaaksi, A. 1995. Object-oriented specification of user interfaces. Software – Practice and Experience, Vol. 25, No. 11, 1203-1221.

Jaaksi, A. 1997. Object-oriented development of interactive systems, Tampere, Finland: Tampere University of Technology, Pub. 201, Dissertation Thesis.

Jackson, M. 1983. System development. Englewood Cliffs: Prentice-Hall, Inc.

Jacobson, I., Booch, G. & Rumbaugh, J. 1999. The Unified Software Development Process. Reading: Addison-Wesley.

Jacobson, I., Christeson, M., Jonsson, P. & Övergaard, G. 1992. Object-oriented software engineering, A use case driven approach. Reading: Addison-Wesley.

Janson, M. & Woo, C. 1995. Comparing IS development tools and methods: using speech act theory. Information & Management, Vol. 28, No. 1, 1-12.

Jarke, M. 1992. Strategies for integrating CASE environments. IEEE Software, Vol. 9, No. 2, 54-61.

Jarke, M., Gallensdörfer, R., Jeusfeld, M., Staudt, M. & Eherer, S. 1995. ConceptBase: a deductive object base for meta data management. Journal of Intelligent Information Systems, Vol. 4, No. 2, 167-192.

Jarke, M., Jeusfeld, M. & Rose, T. 1990. A software process data model for knowledge engineering in information systems. Information Systems, Vol. 15, No. 1, 85-116.

Jarke, M., Mylopoulos, J., Schmidt, J. & Vassiliou, Y. 1992. DAIDA: an environment for evolving information systems. ACM Trans. on Information Systems, Vol. 10, No. 1, 1-50.

Jarke, M. & Pohl K. 1993. Vision driven system engineering. In N. Prakash, C. Rolland & B. Pernici (Eds.) Information Systems Development Process (A-30). Amsterdam: North-Holland, 3-20.

Jayaratna, N. 1994. Understanding and evaluating methodologies: NIMSAD – a systemic framework. London: McGraw-Hill.

Jefferies, R., Miller, J., Wharton, C. & Udea, K. 1991. User interface analysis in real world: a comparison of four techniques. In Proc. of the ACM CHI'91 Conference in Human Factors in Computing Systems, New York, 119-124.

Jenkins, A., Neumann, J. & Wetherbe, J. 1984. Empirical investigation of systems development practices and results. Information & Management, Vol. 7, No. 2, 73-82.

Johnson, G., Scholes, K. & Sexty, R. W. 1989. Exploring strategic management. Englewood Cliffs: Prentice-Hall.

Jones, C. 1986. Systematic software development with VDM. Englewood Cliffs: Prentice-Hall.

Jones, L. & Kydd, C. 1988. An information processing framework for understanding success and failure in MIS development methodologies. Information Management, Vol. 15, No. 5, 263-271.

Kaasboll, J. 1995. Abstraction and concretizing in information systems and problem domains: implications for system descriptions. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 250-265.

Kaasboll, J. & Smordal, O. 1996. Human work as context for development of object-oriented modeling technique. In S. Brinkkemper, K. Lyytinen & R. Welke R. (Eds.) Proc. of the IFIP TC8 WG 8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 111-125.

Kabeli, J. & Shoval P. 2002. A comparison of the FOOM and OPM methodologies for user comprehension of analysis specifications. In T. Halpin, K. Siau & J. Krogstie (Eds.) Proc. of 7th CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'02), 23-36.

Kahn, K. & Gorry G. 1977. Mechanising temporal knowledge. Artificial Intelligence, Vol. 9, No. 1, 87-108.

Kaipala, J. 1997. Augmenting CASE tools with hypertext: desired functionality and implementation issues. In A. Olive & A. Pastor (Eds.) Proc. of the 9th Int. Conf. on Advanced Information Systems Enginering (CAiSE'97). Berlin: Springer, 217-230.

Kaiser, G., Popovich, S. & Ben-Shaul I. 1993. A bi-level language for software process modeling. In Proc. of the 15th Int. Conf. on Software Engineering, Washington: IEEE Computer Society Press, 132-143.

Kangassalo, H. 1982. On the concept of concept in a concept schema. In H. Kangassalo (Ed.) Proc. of the First Scandinavian Research Seminar on

Information Modelling and Data Base Modelling. Acta Universitatis Tamperensis, Ser. B., Vol. 17, University of Tampere, Finland, 129-172.

Kangassalo, H. 2002. Foreword., In S. Spaccapietra, S. March & Y. Kambayaski (Eds.) Proc. of 21st Intern. Conf. on Conceptual Modeling (ER 2002). Berlin: Springer, V-VI.

Kant, I. 1787. Kritik der reinen Verrnunft, translated by N. Kemp Smith as Critique of Purse Reason. New York: St. Martin's Press.

Karam, G. & Casselman, R. 1993. A cataloging framework for software development methods. IEEE Computer, Vol. 26, No. 2, 34-46.

Karlsson, F. 2002. Bridging the gap – between method for method configuration and situational method engineering. In Proc. of the Second Annual Knowledge Foundation Conference for the Promotion of Research in IT.

Karlsson, F., Ågerfalk, P. & Hjalmarson, A. 2001. Method configuration with development tracks and generic project types. In J. Krogstie. K. Siau & T. Halpin (Eds.) Proc. of the 6th CaiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'01).

Kashyap, V. & Sheth, A. 1996. Semantic and schematic similarities between database objects: a context-based approach. The VLDB Journal, Vol. 5, No. 4, 276-304.

Katayama, T. 1989. A hierarchical and functional software process description and its enaction. In Proc. of the 11th Int. Conf. on Software Engineering. IEEE Computer Society / ACM Press, 343-352.

Katz, R. 1990. Toward a unified framework for version modeling in engineering databases. ACM Surveys, Vol. 22, No. 4, 375-408.

Katzenstein, G. & Lerch, J. 2000. Beneath the surface of organizational processes: a social representation framework for business process redesign. ACM Trans. on Information Systems, Vol. 18, No. 4, 383-422.

Kauppi, R. 1967. Einführung in die Theorie der Begriffssysteme. Acta Universitatis Tamperensis, Serie A., Vol. 15, Tampere, Finland.

Kautz, K. & McMaster, T. 1994. Introducing structured methods: an undelivered promise? – a CASE study. Scandinavian Journal of Information Systems, Vol. 6, No. 2, 59-78.

Kavakli, V. & Loucopoulos, P. 1999. Goal-driven business process analysis application in electricity deregulation. Information Systems, Vol. 24, No. 3, 187-207.

Kavakli, E. & Loucopoulos P. 2003. Goal driven requiremetns engineering: evaluation of current methods. In K. Siau, T. Halpin & J. Krogstie (Eds.) Proc. of the 8th CaiSE / IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'03), 1-10.

Kayed, A. 2002. Colomb R., Extracting ontological concepts for tendering conceptual structures. Data & Knowledge Engineering, Vol. 40, No. 1. 71-89.

Keen, P. 1981. Information systems and organizational change. Comm. of the ACM, Vol. 24, No. 1, 24-33.

Keen, P. & Scott Morton, M. 1978. Decision support systems: an organizational perspecetive. Reading: Addison-Wesley.

Kelly, J. & Sherif, Y. 1992. Comparison of four design methods for real-time software development. Information and Software Technology, Vol. 34, No. 2, 76-82.

Kelly, S., Lyytinen, K. & Rossi, M. 1996. MetaEdit+: a fully configurable multi-user and multi-tool CASE and CAME environment. In Y. Vassiliou & J. Mylopoulos (Eds.) Proc. of the 8th Conf. on Advanced Information Systems Engineering (CAiSE'96). Berlin: Springer, 1-21.

Kelly, S. & Tahvanainen V.-P. 1994. Support for incremental method engineering and MetaCASE. In B. Theodoulidis (Ed.) Proc. of the 5th Workshop on the Next Generation of CASE Tools. Memoranda Informatica 94-25. Enschede, The Netherlands: Universiteit Twente, 140-148.

Kendall, J. & Kendall, K. 1993. Metaphors and methodologies: living beyond the systems machine. MIS Quarterly, Vol. 17, No. 2, 149-171.

Kent, W. 1978. Data and reality. Amsterdam: North-Holland.

Kerola, P. 1980. On infological research into the systemeering process. In F. Lucas, T. Land, H. Lincoln K. Supper (Eds.) The Information Systems Environment. Amsterdam: North-Holland, 199-217.

Kerola, P. & Järvinen, P. 1975. Systemeering II – System theoretical and cybernetical model of data system development and use (in Finnish). Helsinki: Gaudeamus.

Kerola, P. & Taggart, W. 1982. Human information processing styles in the information systems development process. In J. Hawggod (Ed.) Evolutionary Information Systems, Amsterdam: North-Holland, 63-86.

Kettinger, W., Teng, J. & Guha S. 1997. Business process change: a study of methodologies, techniques, and tools. MIS Quarterly, Vol. 21, No. 1, 55-80.

Kim, Y. & March, S. 1995. Comparing data modeling formalisms for representing and validating information requirements. Comm. of the ACM, Vol. 38, No. 6, 103-115.

Kinnunen, K. & Leppänen M. 1994. O/A matrix and a technique for methodology enginering. In J. Zupancic & S. Wrycza (Eds.) Proc. of the Fourth International Conference on Information Systems Development (ISD'94), 113-125.

Kinnunen, K. & Leppänen 1996. M., O/A matrix and a technique for methodology engineering. Journal of Systems and Software, Vol. 33, No. 2, 141-152.

Kirchmer, M. 1999. Business process-oriented implementation of standard software. Berlin: Springer-Verlag.

Kirikova, M. 2000. Explanatory capability of enterprise models. Data & Knowledge Engineering, Vol. 33, No. 2, 119-136.

Kishore, R., Zhang, H. & Ramesh, R. 2004. A Helix-Spindel model for ontological engineering. Comm. of the ACM, Vol. 47, No. 2, 69-75.

Kitchenham, B. 1996a. Evaluation software engineering methods and tools – Part 1: The evaluation context and evaluation methods. Software Engineering Notes, Vol. 21, No. 1, 11-15.

Kitchenham, B. 1996b. Evaluation software engineering methods and tools – Part 2: Selecting an appropriate evaluation method – technical criteria. Software Engineering Notes, Vol. 21, No. 2, 11-15.

Kitchenham, B. 1996c. Evaluating software engineering methods and tools – Part 3: Selecting an appropriate evaluation method – Practical Issues. Software Engineering Notes, Vol. 21, No. 4, 9-12.

Kitchenham, B., Travassos, H., von Mayrhauser, A., Nielssink, F., Schneiderwind, N., Singer, J., Takada, S., Vehvilainen, R. & Yang, H. 1999. Towards an ontology of software maintenance. Journal of Software Maintenance: Research and Practice, Vol. 11, No. 6, 365-389.

Klein, H. & Hirschheim, R. 1987. A comparative framework of data modeling paradigms and approaches. The Computer Journal, Vol. 30, No. 1, 8-15.

Klemke, R. 1999. The notion of context in organizational memories. In P. Bouguet, P. Brezillon, F. Castellani, L. Serafini & M. Benerecetti Proc. of Second International and Interdiciplinary Conf. on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer, 483-486.

Kleppe, A., Warmer, J. & Bast, W. 2003. MDA explained: the Model Driven Architecture: practice and promise. Reading: Addison Wesley Professional.

Kling, R. 1987. Defining the boundaries of computing across complex organizations. In R. Boland & R. Hirschheim (Eds.) Critical Issues in Information Systems Research. Chichester: John Wiley & Sons, 307-362.

Klir, G. 1969. An approach to general systems theory. New York: van Nostrand Reinhold Co.

Klooster, M. 1996. Empirical research on the situational dependency of methods for information systems development projects. University of Twente, Master's Thesis.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K. & Smith J. 2002. UML for ontology development. The Knowledge Engineering Review, Vol. 17, No. 1, 61–64.

Kokinov, B. 1999. Dynamics and automaticity of context: a cognitive modeling approach. In P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti & F. Castellani (Eds.) Proc. of 2nd International and Interdisciplinary Conf on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer Verlag, 200-213.

Koontz, H. & O'Donnell C. 1972. Principles of management: an analysis of management functions. 5. edition, New York: McGraw-Hill.

Konsynski, B., Braker, L. & Bracker, W. 1982. A model for specification of office communication. IEEE Trans. on Communications COM-30 (1), 27-36.

Korpela, M., Mursu, A. & Soriyan, H. 2002. Information systems development as an activity. Computer Supported Cooperative Work, Vol. 11, No. 1-2, 111-128.

Korpela, M., Soriyan, H. & Olufokunbi, K. 2000. Activity analysis as a method for information systems development: general introduction and experiments from Nigeria and Finland. Scandinavian Journal of Information Systems, Vol. 12, No. 1, 191-210.

Koskinen, M. 2000. Process metamodelling: Conceptual foundations and applications. Jyväskylä Studies of Computing, No. 7, University of Jyväskylä, Dissertation Thesis.

Koubarakis, M. & Plexousakis, D. 2000. A formal model for business process modeling and design. In B. Wangler & L. Bergman (Eds.) Proc. of 12th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2000). Berlin: Springer-Verlag, 142-156.

Koubarakis, M. & Plexousakis, D. 2002. A formal framework for business process modelling. Information Systems, Vol. 27, No. 5, 299-319.

Kraiem, N., Bourguida, I. & Selmi, S. 2000. Situational method for information system project. In Proc. of Int. Conf. on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet.

Kramer, B. & Luqi, R. 1991. Towards formal models of software engineering processes. Journal of Systems and Software, Vol. 15, No. 1, 63-74.

Kroenke, D. & Dolan, K. A. 1987. Business computer systems: an introduction. Santa Cruz, CA: Mitchell Publishing.

Krogstie, J. 1995. Conceptual modeling for computerized information systems support in organizations. NTH, University of Trondheim, Norway, Dissertation Thesis.

Krogstie, J. 2002. A semiotic approach to quality in requirements specification. In K. Liu, R. Clarke, P.B., Andersen, & R. Stamper R. (Eds.) Proc. of IFIP TC8 / WG8.1 Working Conf. on Organizational Semiotics: Evolving a Science of Information Systems. IFIP Conference Proceedings, Kluwer, 231-249.

Krogstie, J., Lindland, O. & Sindre, G. 1995. Defining quality aspects for conceptual models. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 216-231.

Krogstie, J. & Sindre, G. 1994. Extending a temporal rule language with deontic operators. In Proc. of the 6th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'94), 314-321.

Krogstie, J. & Sölvberg, A. 1996. A classification of methodological frameworks for computerized information systems support in organizations. In B. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8 WG 8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 278-295.

Krogstie, J. & Sölvberg 2000. A., Information systems engineering: conceptual modeling in a quality perspective. Trondheim, Norway: Information Systems Groups, NTNU.

Kronlöf, K. (Ed.) 1993. Method integration: concepts and case studies. Chichester: John Wiley & Sons.

Kruchten, P. 2000. The Rational Unified Process: An introduction. Reading: Addison-Wesley.

Kueng, P., Bichler, P., Kawalek, P. & Schrefl, M. 1996. How to compose an object-oriented business process model. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8 WG 8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 94-110.

Kuhn, T. 1970. The structure of scientific revolution. 2nd edition, Chicago: University of Chicago Press.

Kumar, K. 1984. Participant values in information systems development. McMaster University, Hamilton, Ontario, Unpublished Doctoral Dissertation.

Kumar, K. & Welke, R. 1992. Methodology engineering: a proposal for situation specific methodology construction. In W. Kottermann & J. Senn (Eds.) Challenges and Strategies for Research in Systems Development. Chichester: John Wiley & Sons, 257-269.

Kuutti, K. 1991. Activity theory and its applications to information systems research and development. In H.-E. Nissen, H. Klein & R. Hirschheim (Eds.) Information Systems Research: Contemporary Approaches and Emergent Traditions. Amsterdam: North-Holland, 529-549.

Kuutti, K. 1994. Information systems, cooperative work and activity subjects: the activity theoretical perspective. Department of Information Processing Science, University of Oulu, Finland, Dissertation Thesis.

Kyng, M. & Mathiassen, L. (Eds.) 1997. Computers and design in context. Cambridge, MA, USA: MIT Press.

Ladd, I. & Tsichritzis, D. 1980. An office form flow model. In Proc. of AFIPS National Computer Conference, Vol. 49. Virginia: AFIPS Press, 533-539.

Lang, M. & Duggan, J. 2001. A tool to support collaborative software requirements management. Requirements Engineering, Vol. 6, No. 3, 161-172.

Langefors, B. 1971. Theoretical analysis of information systems. Lund, Sweden: Studentlitterature.

Langefors, B. & Sundgren, B. 1975. Information systems architecture. New York: Petrocelli.

Lanzara, G. 1983. The design process: Frames, metaphors and games. In U. Briefs, C. Ciborra & L. Schneider (Eds.) Systems Design for, with, and by the Users. Amsterdam: North-Holland, 29-40.

Latour, B. 1999. On recalling ANT. In J. Law & J. Hassard (Eds.) Actor Network Oxford. England: Blackwell Publishers, 15-25.

Law, D. 1988. Methods for comparing methods: techniques in software develoment. NCC Publications.

Law, D. & Stamper, R. 1984. Criteria for comparing methodologies for defining system requirements. GDM/NCC SDSS Joint Research Programme, Project SD Technical Paper, NCC.

Layder, D. 1993. New strategies in social research. Cambridge: Polity Press.

Lee, J., Xue, N.-L. & Kuo, J.-Y. 2001. Structuring requirement specifications with goals. Information and Software Technology, Vol. 43, No. 2, 121-135.

Lee, R. 1983. Epistemological aspects of knowledge-based decision support systems. In H. Sol (Ed.) Proc. of Int. Conf. on Processes and Tools for Decision Support Systems. Amsterdam: North-Holland, 25-36.

Lehman, M. 1984. Program evolution. Information Processing Management, Vol. 20, No. 1-2, 19-36.

Lei, Y. & Singh, M. 1997. A comparison of workflow metamodels. In Proc. of the ER-97 Workshop on Behavioural Modelling and Design Transformations: Issues and Opportunities in Conceptual Modelling, Los Angeles.

Lenat, D. & Guha, R. 1990. Building large knowledge-based systems. Reading: Addison-Wesley.

Leont'ev, A. 1978. Activity, consciousness, and personality. Englewood Cliffs: Prentice-Hall.

Leppänen, M. 1984a. Conceptual schema language and a model for view modeling – a linguistic approach. Department of Information Processing, University of Jyväskylä, Finland, Licentiate Thesis (in Finnish).

Leppänen, M. 1984b. Abstraction in information systems development. In M. Sääksjärvi (Ed.) Proc. of the 7th Scandinavian Research Seminar on Systemeering, Part I. Helsinki: Helsinki School of Economics, Studies B-75, 243-286.

Leppänen, M. 1989a. Abstraction analysis of OSSAD-methodology. Research Report, University of Jyväskylä, Finland.

Leppänen, M. 1989b. Conceptual analysis of socio-technical analysis. In K. Fuchs-Kittowski, C. Harman & E. Muhlenberg (Eds.) Proc. of Int. IFIP TC9.1 Conf. on Information Systems, Work and Organizational Design, Berlin, GDR, 77-89.

Leppänen, M. 1993. Database design. Jyväskylä: Department of Information Systems and Computer Science, University of Jyväskylä, Lecture Notes (in Finnish).

Leppänen, M. 1994. Metamodelling: concept, benefits and pitfalls. In J. Zupancic & S. Wrycza (Eds.) Proc. of the Fourth Int. Conf. on Information Systems Development (ISD'94), 126-137.

Leppänen, M. 2000. Toward a method engineering (ME) method with an emphasis on the consistency of ISD methods. In K. Siau K. (Ed.) Proc. of the Fifth CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'00).

Leppänen, M. 2001. Database application design. Jyväskylä: Department of Information Systems and Computer Science, University of Jyväskylä, Lecture Notes (in Finnish).

Leppänen, M., Lyytinen, K. & Halttunen, V. 1991. Tietojenkäsittelystrategian määrittely – Strateginen tietohallintopalveluiden kehittämismenetelmä (SPITS) (in Finnish). Jyväskylä: Publications of the Department of Computer Science, TU-10, University of Jyväskylä.

Leppänen, M. & Savolainen, V. 1988. Refinement of OSSAD methodology by multi-client field tests. In J. Kaasboll J. (Ed.) Report of the 11th Int. Research Seminar on Information Systems (IRIS'88), 395-429.

Leppänen, M. & Savolainen, V. 1989a. OSSAD application in a Finnish paper machine company – developing strategies for implementation of teleservices in a sales department. In R. Baron & E. Beslmuller (Eds.) OSSAD Field Test Report, ESPRIT Project No. 285, R&D Area 4.1, Office Systems Science and Human Factors, Munich: IOT, Chapter 4.

Leppänen, M. & Savolainen V. 1989b. A classification framework for OIS methodologies. In K. Boyanov K. & V. Angelinov (Eds.) Network Information Processing Systems. Amsterdam: North-Holland, 299-307.

Levinson, S. 1983. Pragmatics. London: Cambridge University Press.

Li, Q. & Dong, G. 1994. A framework for object migration in object-oriented databases. Data & Knowledge Enginering, Vol. 13, No. 3, 221 - 242.

Liang, Y. 2000. An approach to assessing and comparing object-oriented analysis methods. Journal of Object-Oriented Programming, June, 27-33.

Liberman, H., Stein, A. & Ungar, D. 1988. Of types and prototypes: the treaty of Orlando. OOPSLA '87 Addendum to the Proceedings. Special Issue of SIGPLAN Notices, Vol. 23, No. 5, 43-44.

Lin, C.-Y. & Ho, C.-S. 1999. Generating domain-specific methodical knowledge for requirements analysis based on methodology ontology. Information Sciences, Vol. 114, No. 1-4, 127-164.

Lindgreen, P. 1995. Anything, everything and things playing roles: three realizing principles as a contribution to a platform for understanding. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of the IFIP Int. Working Conf. on Information System Concepts – Towards a Consolidation of views. London: Chapman & Hall, 195-214.

Lindland, O., Sindre, G. & Sölvberg A. 1994. Understanding quality in conceptual modeling. IEEE Software, Vol. 11, No. 2, 42-49.

Lindtner, P. 1992. Domänenwissen in Methoden zur Analyse Betrieblischer Informationssysteme. The Institute for Information Management, University of St. Gallen, St. Gallen, Switzerland, Dissertation Thesis.

Liskov, B. & Zilles S. 1974. Programming with abstract data types. In Proc. of ACM SIGPLAN Conf. on Very High Level Languages, SIGPLAN Notices, Vol. 9, No. 4, 50-59.

Liu, L. & Yu, E. 2002. Designing web-based systems in social context: a goal and scenario based approach. In A. Banks Pidduck, J. Mylopoulos, C. Woo C. & M. Tamer Ozsu (Eds.) Proc. of 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'2002). LNCS 2348, Berlin: Springer-Verlag, 37-51.

Ljunberg, J. & Holm, P. 1996. Speech acts on trial. Scandinavian Journal of Information Systems, Vol. 8, No. 1, 29-52.

Lonchamp, J. 1993. A structure conceptual and terminological framework for software process engineering. In Proc. of the 2nd Int. Conf. on the Software Process. Washington: IEEE Computer Society Press, 41-53.

Louadi, M., Polladis, Y. & Teng, J. 1991. Selecting a system development methodology: a contingency framework. Information Resources Management Journal, Vol. 4, No. 1, 11-19.

Loucopoulos, P. 2000. From information modeling to enterprise modeling. In S. Brinkkemper, E. Lindencrona & A. Sölvberg (Eds.) Information Systems Engineering – State of the Art and Research Themes. Berlin: Springer-Verlag, 67-78.

Loucopoulos, P., Kavakli, V., Prekas, N., Rolland, C., Grosz, G. & Nurcan, S. 1998. Using the EKD approach: the modelling component. ELEKTRA – Project No. 22927, ESPRIT Programme 7.1.

Louridas, P. & Loucopoulos, P. 1996. A framework for evaluating design rationale methods. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'96).

Lucas, H. 1978. The evolution of an information system: from key-man to every person. Sloan Management Review, Vol. 19, No. 2, 39-52.

Lundeberg, M. 1982. The ISAC approach to specification of information systems and its application to the organization of an IFIP working conference. In T. Olle, H. Sol & A. Verrijn-Stuart (Eds.) Information Systems Design Methodologies: A Comparative Review. Amsterdam: North-Holland, 173-234.

Lundeberg, M., Goldkuhl, G. & Nilsson, A. 1981. Information systems development: A systematic approach. Englewood-Cliff: Prentice-Hall.

Luoma, J., Kelly, S. & Tolvanen, J.-P. 2004. Defining domain-specific modeling languages: collected experiences. In J.-P. Tolvanen, J. Sprinkle & M. Rossi (Eds.) Proc. of the 4th OPSLA Workshop on Domain-Specific Modeling (DSM'04). Jyväskylä: Computer Science and Information Systems Reports, Technical Reports TR-33, University of Jyväskylä, 1-10.

Lyons, J. 1977. Semantics. Volume I-II, Cambridge: Cambridge University Press.

Lyons, J. 1981. Language and linguisics: An introduction. Cambridge: Cambidge University Press.

Lyytinen, K. 1985. Implications of theories of language for information systems. MIS Quarterly, Vol. 9, No. 1, 61-74.

Lyytinen, K. 1986. Information systems development as social action: framework and critical implications. Jyväskylä Studies in Computer Science, Economics, and Statistics, No. 8, University of Jyväskylä, Finland, Dissertation Thesis.

Lyytinen, K. & Robey D. 1999. Learning failure in information systems development. Info Systems Journal, Vol. 9, No. 2, 85-101.

Macauley, L. 1993. Requirements capture as a cooperative activity. In Proc. of IEEE International Symposium on Requirements Engineering. Los Alamitos, California: IEEE Computer Science Press, 174-181.

MacLean, A., Young, R., Belloti, V. & Moran, T. 1991. Questions, options and criteria: elements of design space analysis. Human-Computer Interaction, Vol. 6, No. 3/4, 201-250.

MacLennan, B. 1982. Values and objects in programming languages. SIGPLAN Notices, Vol. 17, No. 12, 70-79.

Maddison, R., Baker, G., Bhabuta, L., Fitzgerald, G., Hindle, K., Song, J., Stokes, N. & Wood J. 1984. Feature analysis of five information system methodologies. In T. Bemelmans (Ed.) Beyond Productivity: Information Systems for Organizational Effectiveness. Amsterdam: North-Holland, 277-306.

Makmuri, S. 1998. Best practices for internet commerce. Intelligent Enterprise, Vol. 1, No. 4, 29-40.

March, S. & Smith, G. 1995. Design and natural science research on information technology. Decision Support Systems, Vol. 15, No. 4, 251-266.

Markus, M. L. 1983. Power, politics and MIS implementation. Comm. of the ACM, Vol. 26, No. 6, 430-444.

Markus, M. & Bjorn-Andersen N. 1987. Power over users: its exercise by system professionals. Comm. of the ACM, Vol. 30, No. 6, 498-504.

Markus, M. L., Majchrzak, L. & Gasser L. 2002. A design theory for systems that support emergent knowledge processes. MIS Quarterly, Vol. 26, No. 3, 179-212.

Markus, M. L. & Mao J.-Y. 2004. Participation in development and implementation – updating an old, tired concept for today's IS contexts. Journal of the Association for Information Systems, Vol. 5, No. 11-12, 514-544.

Martin, J. 1982. Strategic data-planning methodologies. Englewood Cliffs: Prentice-Hall.

Martin, J. 1989. Information engineering. Introduction. Englewood Cliffs: Prentice-Hall.

Martin, J. & McGlure, C. 1985. Diagramming techniques for analyst and programmers, Englewood Cliffs: Prentice-Hall.

Martin, R. & Robertson, E. 2000. A formal enterprise architecture framework to support multi-model analysis. In Proc. of the 5th CAiSE/IFIP 8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'00).

Massacci, F. 1996. Contextual reasoning is NP –complete. In H. Shrobe & T. Senator (Eds.) Proc. of the 13th National Conference on Artificial Intelligence and the 8th Innovation Applications of Artificial Intelligence Conference, Vol. 2. Menlo Park: AAAI Press, 621-626.

Masters, S. & Kitson, D. 1992. An analysis of SEI Software Process Assessment Results: 1987-1991. Technical Report CMU/SEU-92-TR-24. Software Engineering Institute, Carnegie Mellon Uniersity, Pittsburg, Pa.

Matheus, C. J., Kokar, M. M. & Baclawski K. 2003. A core ontology for situation awareness. In Proc. of the 6th International Conf. on Information Fusion (FUSION'03), 545 –552.

Mathiassen, L. 1982. Systemudvikling og systemudviklingsmetode, Datalogisk Afdelning, Matematisk Institut, Aarhus Universitet, Doktorsavhandling.

Mathiassen, L. 1998. Reflective systems development. Scandinavian Journal of Information Systems, Vol. 10, No. 1/2, 67-117.

Mathiassen, L. & Munk-Madsen, A. 1986. Formalizations in systems development. Behaviour and Information Technology, Vol. 5, No. 2, 145-155.

Mathiassen, L. & Munk-Madsen, A. 1988. Myths and reality in software development. Technical Report, Institute of Elecronic Systems, Aalborg University.

Mathiassen, L., Munk-Madsen, A., Nielsen, P. & Stage, J. 1996. Method engineering: Who's the customer? In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8 WG 8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 232-245.

Mathiassen, L., Munk-Maddsen, A., Nielsen, P. & Stage, J. 2000. Object-oriented analysis & design. Aalborg, Denmark: Marko Publishing ApS.

Mattos, N. 1988. Abstraction concept: the basis for data and knowledge modeling. In C. Batini (Ed.) Proc. of the 7th Intern. Conf. on E-R Approach. Amsterdam: North-Holland, 331-350.

Matwin, S. & Kubat, M. 1996. The role of context in concept learning. In Proc. of the 13th International Conference on Machine Learning (ICML-96), Workshop on Learning in Context-Sensitive Domains, Bari, Italy, 1-5.

Mayer, R., Menzel, C., Painter, M., deWitte, P., Blinn, T. & Perakath, B. 1995. Information integration for concurrent engineering (IICE) IDEF3 Process Description Capture Method Report [Referred on 23.6.2002]. Available at URL: < http://www.idef.com/pdf/Idef3_fn.pdf >.

McChesney, I. 1995. Towards a classification scheme for software process modeling approaches. Information and Software Technology, Vol. 37, No. 7, 363-374.

McDermott, D. 1982. A temporal logic for reasoning about processes and plans. Cognitive Science, Vol. 6, No. 2, 101-155.

McGregor, D. M. 1960. The human side of enterprise. London: McGraw-Hill.

McGuinness, D., Fikes, R., Hendler, J. & Stein, L. 2002. DAML+OIL: an ontology language for the Semantic Web. IEEE Intelligent Systems, Vol. 17, No. 5, 72-80.

McLeod, D. & King, R. 1980. Applying a semantic database model. In P. Chen (Ed.) Entity-relationship Approach to Systems Analysis and Design. Amsterdam: North-Holland, 193-210.

Meersman, R., Shi, Z. & Kung, C.-H. (Eds.) 1990. Proc. of the IFIP TC2/TC8/WG2.6/WG8.1 Working Conf. on Artificial Intelligence in Database and Information Systems (DS-3). Amsterdam: North-Holland.

Melao, N. & Pidd, M. 2000. A conceptual framework for understanding business processes and business process modeling. Information Systems Journal, Vol. 10, No. 2, 105-129.

Mentzas, G. 1994. A functional taxonomy of computer-based information systems. International Journal of Information Management, Vol. 14, No. 6, 397-410.

Mentzas, G., Halaris, C. & Kavadias, S. 2001. Modeling business processes with workflow systems: an evaluation of alternative approaches. International Journal of Information Management, Vol. 21, No. 2, 123-135.

Merbeth, G. 1991. Maestro II – das integrierte CASE-system von Softlab. In H. Balzert (Ed.) CASE Systeme und Werkzeuge. BI Wissenschaftsverlag, 319-336.

Mesarovic, M., Macko, D. & Takahara, Y. 1970. Theory of hierarchical, multilevel, systems. New York: Academic Press.

Meyer, B. 1990. Introduction to the theory of programming languages. Englewood Cliffs: Prentice-Hall.

Mi, P. & Scacchi, W. 1996. A meta-model for formulating knowledge-based models of software development. Decision Support Systems, Vol. 17, No. 3, 313-330.

Middleton, P. 1994. Euromethod: the lessons from SSADM. In W. Baets (Ed.) Proc. of the 2nd European Conference on Information Systems (ECIS 1994). Brekelen, The Netherlands : Nijenrode University Press, 359-366.

Middleton, P. 1999. Managing information system development in bureaucracies. Information and Software Technology, Vol. 41, No. 8, 473-482.

Miller, G. 1990. Wordnet: an online lexical database. International Journal of Lexicography, Vol. 3, No. 4, 235-312.

Minsky, M. 1965. Models, minds, machines. In Proc. of IFIP Congress. Montvale, New Jersey: AFIPS Press, 45-49.

Monteiro, E. 2000. Actor-network theory. In C. Ciborra (Ed.), From Control to Drift: the Dynamics of Corporate Information Infrastructures. Oxford: Oxford University Press.

Moody, D. 2003a. The method evaluation model: a theoretical model for validating information systems design methods. In Ivan *et al.* (Eds.) Proc. of the European Conference of Information Systems.

Moody, D. 2003b. Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice. In Ivan *et al.* (Eds.) Proc. of the European Conference of Information Systems.

Moody, D. 2003c. Theoretical and practical issues in evaluating the quality of conceptual models. In A. Olive, M., Yoshikawa & E. Yu et al. (Eds.) Proceedings of the Advanced Conceptual Modelling Techniques, ER 2002 Workshops – ECDM, MoblMod, IWCMO, and eCOMO. LNCS 2784, Berlin: Springer Verlag, 241-242.

Moor de, A. & Jeusfeld, M. 2001. Making workflow change acceptable. Requirements Engineering, Vol. 6, No. 2, 75-96.

Morisio, M., Seaman, C., Basili, V., Parra, A., Kraft, S. & Condon, S. 2002. COTS-based software development: processes and open issues. The Journal of Systems and Software, Vol. 61, No. 3, 189-199.

Morris, C. W. 1938. Foundations of the theory of signs. In O. Neurath, R. Carnap & C. Morris (Eds.) International Encyclopedia of Unified Science. Chicago: University of Chicago Press, 77-138.

Morris, C. W. 1946. Signs, language, and behaviour. Englewood Cliffs: Prentice-Hall.

Morris, C. W. 1964. Signification and significance. Cambridge: MIT Press.

Morris, J. H. Jr. 1973. Types are not sets. In Proc. of the ACM Symposium on Principles of Programming Languages (POPL). ACM Press, 120-124.

Motik, B., Maedche, A. & Volz, R. 2002. A conceptual modelling approach for semantic-driven enterprise applications. In R. Meersman & Z. Tari (Eds.) On the Move to Meaningful Internet Systems - Confederated International Conferences DOA, CoopIS and ODBASE 2002. LNCS 2519, Berlin: Springer-Verlag. 1082-1099.

Motschnig-Pitrik, R. 1993. The semantics of parts versus aggregates in data/knowledge modeling. In C. Rolland, F. Bodard & C. Cauvet (Eds.) Proc. of the 5th Int. Conf. on Advanced Information Systems Enginering (CAiSE'93). LNCS 685, Berlin: Springer, 352-373.

Motschnig-Pitrik, R. 1995. An integrating view on the viewing abstraction: contexts and perspectives in software development, AI, and databases. Journal of Systems Integration, Vol. 1, No. 1, 23-60.

Motschnig-Pitrik, R. 1999. Context and views in object-oriented languages. In P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti & F. Castellani (Eds.) Proc. of Second International and Interdisciplinary Conf. on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer Verlag, 256-269.

Motschnig-Pitrik, R. 2000. A generic framework for the modeling of contexts and its applications. Data & Knowledge Engineering, Vol. 32, No. 2, 145-180.

Motschnig-Pitrik, R. & Kaasboll, J. 1999. Part-Whole relationship categories and their application in object-oriented analysis. IEEE Trans. on Knowledge and Data Engineering, Vol. 11, No. 5, 779-797.

Motschnig-Pitrik, R. & Mylopoulos, J. 1996. Semantics, features, and applications of the viewpoint abstraction. In P. Constantopoulos, J. Mylopoulos & Y. Vassiliou (Eds.) Proc. of the 8th International Conference on Advanced Information Systems Engineering (CAiSE'96). LNCS 1080, Berlin: Springer-Verlag, 514-539.

Motschnig-Pitrik, R. & Nykl, L. 2001. The role and modeling of context in a cognitive model of Rogers' person-centred approach. In V. Akman, P. Bouquet, R. Thomason & R. Young (Eds.) Third International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT' 2001). LNCS 2116, Berlin: Springer-Verlag, 275-289.

Motschnig-Pitrik, R. & Storey, V. 1995. Modelling of set membership: the notion and the issues. Data & Knowledge Engineering, Vol. 16, No. 2, 147-185.

Moulin, B. & Creasy, P. 1992. Extending the conceptual graph approach for data conceptual modelling. Data & Knowledge Engineering, Vol. 8, No. 3, 223-248.

Moynihan, T. 1993. Modelling the software process in terms of the system representations and transformation steps used. Information and Software Tchnology, Vol. 35, No. 3, 181-188.

Moynihan, T. 1996. An attempt to compare object-orientation and functional-decomposition in communicating information system functionality to users. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'96).

Muchlen zur, M. 1999. Evaluation of workflow management systems using meta models. In Proc. of the 32nd Hawaii International Conf. on System Sciences, IEEE.

Mumford, E. 1981. Participative systems design: structure and method. Systems, Objectives, Solutions, Vol. 1, No. 1, 5-19.

Mumford, E. 1983. Designing human systems for new technology, The ETHICS method. Machester, UK: Machester Business School.

Mumford, E. & Weir, M. 1979. Computer systems in work design – the ETHICS method. London: Associated Business Press.

Murphy, L. 1996. Information product evaluation as asynchronous communication in context: a model for organisational research. In Proc. of the First ACM International Conference on Digital Libaries, 134-142.

Mustonen, S. 1978. Tavoitteisen järjestelmän kyberteettinen analyysi päätäntäteorioiden ja systemoinnin metatutkimuksessa (in Finnish). Report A7, Institute of Data Processing Science, University of Oulu, Oulu.

Mylopoulos, J. 1998. Information modelling in the time of the revolution. Information Systems, Vol. 23, No. 3/4,127-155.

Mylopoulos, J., Berstein, P. & Wong, H. 1980. A language facility for designing database-intensive applications. ACM Trans. on Database Systems, Vol. 5, No. 2, 185-207.

Mylopoulos, J., Borgida, A., Jarke, M. & Koubarakis, M. 1990. Telos: a language for representing knowledge about information systems. ACM Trans. on Office Information Systems, Vol. 8, No. 4, 325-362.

Mylopoulos, J., Chung, L., Liao, S. & Wang, H. 2001. Exploring alternatives during requirements analysis. IEEE Software, Vol. 18, No. 1, 92-96.

Mylopoulos, J., Chung, L. & Nixon, B. 1992. Representing and using non-functional requirements: a process-oriented approach. IEEE Trans. on Software Engineering, Vol. 18, No. 6, 483-497.

Myrhaug, H. 2001. Towards life-long and personal context spaces. In Proc. of Workshop on User Modelling for Context-Aware Applications.

Nandhakumar, J. & Avison, D. 1999. The fiction of methodological development: a field study of information systems development. Information Technology & People, Vol. 12, No. 2, 176-191.

Nardi, B. 1996. Activity Theory and Human-Computer Interaction. In B. Nardi (Ed.) Context and Consciousness: Activity Theory and Human-Computer Interaction. Cambridge, Massachusetts: MIT Press, 7-16.

NATURE Team 1996. Defining visions in context: models, processes and tools for requirements engineering. Information Systems, Vol. 21, No. 6, 515-547.

Naumann, J.D., Davis, G. & McKeen, J. 1980. Determining information resources: a contingency method for selection of a requirements assurance strategy. The Journal for Systems and Software, Vol. 1, No. 4, 273-281.

NCC 1987. The STARTS Guide. 2nd edition, NCC.

Necco, C., Gordan, C. & Tsai, N. 1987. Systems analysis and design: current practices. MIS Quarterly, Vol. 11, No. 4, 461-476.

Newman, M. & Noble, F. 1990. User involvement as an interaction process: a case study. Information Systems Research. Vol. 1, No. 1, 89-113.

Newman, W. 1980. Office models and office systems design. In N. Naffah (Ed.) Integrated Office Systems, Amsterdam: North-Holland, 3-10.

Nguyen, L. & Swatman, P. 2003. Managing the requirements engineering process. Requirements Engineering, Vol. 8, No. 1, 55-68.

Nguyen, M. & Conradi, R. 1996. Towards a rigorous approach for managing process evolution. In C. Montangero (Ed.) Proc. of the 5th European Workshop on Software Process Technology (EWSPT'96). LNCS 1149, Berlin: Springer-Verlag, 18-35.

Niiniluoto, J. 1999. Critical scientific realism. Oxford: Oxford University Press.

Nijholt, A. 1988. Computers and languages. Amsterdam: North-Holland.

Nijssen, G. & Halpin, T. 1989. Conceptual schema and relational database design: a fact oriented approach. Englewood Cliffs: Prentice-Hall.

Nilsson, B. 2000. On levels of business modeling, communication and model architectures. In S. Brinkkemper, E. Lindencrona & A. Sölvberg (Eds.) Information Systems Engineering – State of the Art and Research Themes. Berlin: Springer, 275-287.

Nissen, H.-E. 1980. Towards a multi subject groups conception of information systems. In K. Lyytinen & E. Petola (Eds.) Report of the 3rd Scandinavian Research Seminar on Systemeering Models. Institute of Computer Science, University of Jyväskylä, 141-170.

Nissen, H., Jeusfeld, M., Jarke, M., Zemanek, G. & Huber, H. 1996. Managing multiple requirements perspectives with metamodels. IEEE Software, Vol. 13, No. 2, 37-48.

Nonaka, I. 1994. Dynamic theory of organizational knowledge creation. Organization Science, Vol. 5, No. 1, 14-37.

Nonaka, I. & Takeuchi, H. 1995. The knowledge-creating company. New York: Oxford University Press.

Noy, N. & McGuinness, D. 2001. Ontology development 101: a guide to creating your first ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical

Report SMI-2001-0880 [Referred on 3.6.2004]. Available at URL: <http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-2001-880.html>

Nunamaker, J., Chen, M. & Purdin, T. 1991. Systems development in information systems research. Journal of Management Information Systems Research, Vol. 7, No. 3, 89-106.

Nurminen, M. 1988. People or computers: three ways of looking at information systems. Lund: Studentlitteratur.

Nuseibeh, B. & Finkelstein, A. 1992. ViewPoints: a vehicle for method and tool integration. In G. Forter, N. H. Madhavji & H. A. Muller (Eds.) Proc. of the Int. Workshop on CASE (CASE'92). Montreal, Canada: IEEE Computer Society Press, 50-60.

Nuseibeh, B., Finkelstein, A. & Kramer, J. 1996. Method engineering for multi-perspective software development. Information and Software Technology, Vol. 38, No. 4, 267-274.

Nygaard, K. & Dahl, O.-J. 1978. The development of the SIMULA languages. In R. Wexelblat (Ed.) ACM SIGPLAN History of Programming Languages Conference, Los Angeles, SIGPLAN Notices, Vol. 13, No. 8, 245-272.

Oberweis, A. & Lausen, G. 1988. On the representation of temporal knowledge in office systems. In C. Rolland, F. Bodart & M. Leonard (Eds.) Temporal Aspects in Information Systems. Amsterdam: North-Holland, 125-139.

Odell, J. 1994. Six different kinds of compositions. Journal of Object-Oriented Programming, Vol. 7, No. 8, 10-15.

Odell, J. 1996. A primer to method engineering. In S. Brinkkemper, K. Lyytinen, R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 1- 7.

Oei, J. 1995. A meta model transformation approach towards harmonization in information system modeling. In E. Falkenberg, W. Hesse & A. Olive (Eds.) Proc. of IFIP WG8.1 Working Conf. on Information System Concepts – Towards a Consolidation of Views. London: Chapman & Hall, 106-127.

Oei, J., Proper, H. & Falkenberg, E. 1994. Evolving information systems: meeting the ever-changing environment. Information Systems, Vol. 4, No. 3, 213-233.

Ogden, C. & Richards I. 1923.The meaning of meaning. London: Kegan Paul.

Olarnsakul, M. & Batanov, D. 2003. A method for developing component-oriented applications: a use-context driven approach toward component coordination. Knowledge and Information Systems, Vol. 5, No. 4, 466-502.

Oliga, J. 1988. Methodological foundations of systems methodologies. Systems Practice, Vol. 1, No. 1, 87-112.

Olive, A. 1983. Analysis of conceptual and logical models in information systems development methodologies. In T. Olle, H. Sol & C. Tully (Eds.) Information Systems Design Methodologies: A Feature Analysis. Amsterdam: North-Holland, 63-85.

Olive, A. 2002. Representation of generic relationship types in conceptual modelling. In A. Banks Pidduck, J. Mylopoulos, C. Woo & T. Ozsu (Eds.) Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'2002). Berlin: Springer, 675-691.

Olle, T., Hagelstein, J., MacDonald, I., Rolland, C., Sol, H., van Assche, F. & Verrijn-Stuart, A. 1988a. Information Systems Methodologies – A Framework for Understanding. 2nd edition. Reading: Addison-Wesley.

Olle, T., Sol, H. & Bhabuta, J. (Eds.) 1988b. Proc. of the IFIP WG8.1 Working Conf. on Computerized Assistance During the Information Systems Life Cycle. Amsterdam: North-Holland.

Olle, T.W., Sol, H. & Tully, C. (Eds.) 1983. Proc. of the IFIP WG8.1 Working Conf. on Feature Analysis of Information Systems Design Methodologies. Amsterdam: North-Holland.

Olle, T.W., Sol, H. & Verrijn-Stuart, A. (Eds.) 1982. Proc. of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies. Amsterdam: North-Holland.

Olle, T.W., Sol, H. & Verrijn-Stuart, A. (Eds.) 1986. Proc. of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies: Improving the Practice. Amsterdam: North-Holland.

OMG 2001. Unified Modeling Language, v 1.4 [Referred on 24.6.2002]. Available at URL: http://www.omg.org/docs/formal/01-09-67.pdf.

OMG 2002. Meta-Object Facility (MOF) Specification, v. 1.4, April [Referred on 12.1.2003]. Available at URL: <http://www.omg.org/cgi-bin/doc?formal/2002-04-03>

OMG 2003. Unified Modeling Language, v 2.0, Superstructure specification [Referred on 24.6.2004] Available at URL: <http://www.omg.org/docs/ptc/03-08-02.pdf >.

Opdahl, A. & Henderson-Sellers, B. 2001a. Grounding the OML metamodel in ontology. The Journal of Systems and Software, Vol. 57, No. 2, 119-143.

Opdahl, A. & Henderson-Sellers, B. 2002. Ontological evaluation of the UML using the Bunge-Wand-Weber model. Software and Systems Modeling, Vol. 1, No. 1, 43-67.

Opdahl, A., Henderson-Sellers, B. & Barbier, F. 2001b. Ontological analysis of whole-part relationships in OO-models. Information and Software Technology, Vol. 43, No. 6, 387-399.

Opdahl, A. & Sindre, G. 1994. A taxonomy for real-world modeling concepts. Information Systems, Vol. 19, No. 3, 229-241.

Orlikowski, W. 1996. Improvising organizational transformation over time: a situated change perspective. Information Systems Research, Vol. 7, No. 1, 63-92.

Orlikowski, W. & Baroudi, J. 1991. Studying information technology in organizations: research approaches and assumptions. Information Systems Research, Vol. 2, No. 1, 1-28.

Orlikowski, W. & Yates, J. 1994. Genre repertoire: the structuring of communicative practices in organizations. Administrative Science Quarterly, Vol. 39, No. 4, 541-574.

Orr, J. 1993. Ethnography and organizational learning: In pursuit of learning at work. In C. Zucchermaglio & S. Stucky (Eds.) The NATO Advanced Research Workshop on Organizational Learning and Technical Change. Berlin: Springer-Verlag, 47-60.

Ouchi, W. 1981. Theory Z. Reading: Addison-Wesley.

Padgham, L. & Taylor, G. 1997. A system for modeling agents having emotion and personality. In L. Cavedon, A. Rao & W. Wobcke (Eds.) Intelligent Agent Systems. LNAI 1209, Berlin: Springer-Verlag, 59-71.

Paige, R. 1999. When are methods complementary. Information and Software Technology, Vol. 41, No. 3, 157-162.

Palvia, P. & Nosek, J. 1993. A field examination of system life cycle techniques and methodologies. Information & Management, Vol. 25, No. 2, 73-84.

Parker, C. S. 1989. Management information systems: strategy and action. London: McGraw-Hill.

Parnas, D. 1972. A technique for software module specification with examples. Comm. of the ACM, Vol. 15, No. 5, 330-336.

Parr, A. & Shanks, G. 2000. A model of ERP project implementation. Journal of Information Technology, Vol. 15, No. 4, 289-303.

Parsons, J. & Wand , Y. 1993. Object-oriented systems analysis: a representation view. Working Paper 93-MIS-001, The University of Bristish Colombia.

Parsons, J. & Wand, Y. 1997. Choosing classes in conceptual modeling. Comm. of the ACM, Vol. 40, No. 6, 63-69.

Parsons, J. & Wand, Y. 2000. Emancipating instances from the tyranny of classes in information modeling. ACM Transactions on. Database Systems, Vol. 25, No. 2, 228-268.

Paulk, M., Curtis, B., Chrissis, M. & Weber, C. 1993. Capability maturity model, version 1.1. IEEE Software, Vol. 10, No. 4, 8-27.

Paulson, D. & Wand, Y. 1992. An automated approach to information systems decomposition. IEEE Trans. on Software Engineering, Vol. 18, No. 3, 174-189.

Peckham, J. & Maryanski, F. 1988. Semantic data models. ACM Computing Surveys, Vol. 20, No. 3, 153-189.

Peirce, C. 1955. Philosophical writings of Peirce, edited by J. Buchle. New York: Dover.

Peirce, C. 1991. The essential Peirce, Vol. 1, edited by N. Houser & C. Kloesel. Bloomington: Indiana University Press.

Penco, C. 1999. Objective and cognitive context. In P. Bouquet, L. Serafini, P. Brezillon, M. Benerecetti & F. Castellani (Eds.) Proc. of the Second International and Interdisciplinary Conf. on Modeling and Using Context (CONTEXT'99). LNAI 1688, Berlin: Springer-Verlag, 270-283.

Peristeras, V. & Tarabanis K. 2000. Towards an enterprise architecture for public adminstration using a top-down approach. European Journal of Information Systems, Vol. 9, No. 4, 252-260.

Peters, P., Mandelbaum, M. & Jarke, M. 1996. Simulation-based method engineering in faderated organizations. In S. Brinkkemper, K. Lyytinen, R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 246-262.

Pfleeger, S. & Hatton, L. 1997. Investigating the influence of formal methods. IEEE Computer, Vol. 30, No.2, 33-43.

Phalp, K. 1998. The CAP framework for business process modeling. Information and Software Technology, Vol. 40, No.13, 731-744.

Pijl van der, G., Swinkels, G. & Verrijdt, J. 1997. ISO 9000 versus CMM: standardization and vertification of IS development. Information & Management, Vol. 32, No. 6, 267-274.

Pirotte, A. & Massart, D. 2004. Integrating two descriptions of taxonomies with materialization. Journal of Object Technology, Vol. 3, No. 5, 143-149.

Pirotte, A., Zimanyi, E., Massart, D. & Yakusheva, T. 1994. Materialization: a prowerful and ubiquitous abstraction pattern. In J. Bocca, M. Jarke M. & C. Zaniolo (Eds.) Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB'94). San Mateo: Morgan Kaufmann, 630-641.

Platt, J. 1971. Grammatical form and grammatical meaning: a tagmemic view of Fillmore's deep structure case concepts. Amsterdam: North-Holland.

Pohl, K. 1993. The three dimensions of requirements engineering. In C. Rolland, F. Bodart & C. Cauvet (Eds.) Proc. of the 5th Int. Conf. on Advanced Information Systems Engineering (CAiSE'93). LNCS 685, Berlin: Springer-Verlag, 275-292.

Pohl, K. 1994. The three dimensions of requirements engineering: a framework and its application. Information Systems, Vol. 19, No. 3, 243-258.

Pohl, K. 1996. Process-centered requirements engineering. New York: John Wiley Research Science Press.

Pohl, K., Dömges, R. & Jarke, M. 1997. Towards method-driven trace capture. In A. Olive & J. Pastor (Eds.) Proc. of the 9th Int. Conf. on Advanced Information Systems Engineering (CAiSE'97). Berlin: Springer, 103-116.

Pohl, K., Weidenhaupt, K., Bömges, R., Haumer, P., Jarke, M. & Klamma, R. 1999. PRIME – towards process-integrated modeling environments. ACM Trans. on Software Engineering and Methodologies, Vol. 8, No. 4, 343-410.

Polo, M., Piattini, M. & Ruiz, F. 2002. Using a qualitative research method for building a software maintenance methodology. Software – Practice and Experience, Vol. 32, No. 13, 1239-1260.

Potter, W. & Kerschberg, L. 1988. A unified approach to modeling knowledge and data. In R. Meersman & A: Sernadas (Eds.) Data and Knowledge (DS-2). Amsterdam: North-Holland, 265-291.

Potts, C. 1989. A generic model for representing design methods. In Proc. of the 11th International Conference on Software Engineering. Los Alamitos: IEEE Computer Society Press, 217-226.

Prakash, N. 1997. Towards a formal definition of methods. Requirements Engineering, Vol. 2, No. 1, 23-50.

Prakash, N. 1999. On method statics and dynamics. Information Systems, Vol. 24, No. 8, 613-637.

Prevelakis, V. & Tsichritzis, D. 1993. Perspectives on software developments. In C. Rolland, F. Bodard & C. Cauvet (Eds.) Proc. of the 5th Int. Conf. on Advanced Information Systems Engineering (CAiSE'93). LNCS 685, Berlin: Springer-Verlag, 586-600.

Punter, T. & Lemmen, K. 1996. The MEMA-model: towards a new approach for methods engineering. Journal of Information and Software Technology, Vol. 38, No. 4, 295-305.

Päivärinta, T. 2002. Comparison of the genre-based ISD approach to 11 others. In T. Halpin, K. Siau & J. Krogstie (Eds.) Proc. of 7th CaiSE/IFIP WG8.1 International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'02), 109-120.

Quine, W. 1953a. "On what there is", reprinted in Quine (1953b), 1-19.

Quine, W. 1953b. From a logical point of view. Cambridge: Harward University Press.

Raccah, P. 1997. Science, language, and situation. In Proc. of the 2nd European Conf. on Cognitive Science, Workshop on Context (ECCS'97).

Raelin, J. A. 2001. Public reflection as the basis of learning. Management Learning, Vol. 32, No. 1, 11-30.

Rahim, M., Seyal, A. & Rahman, M. 1998. Use of software systems development methods – an empirical study in Brunei Darussalam. Information and Software Technology, Vol. 39, No. 13, 949-963.

Ralyte, J. 2002. Requirements definition for the situational method engineering. In C. Rolland, S. Brinkkemper & M. Saeki (Eds.) Proc. of IFIP TC8 / WG8.1 Working Conference on Engineering Information Systems in the Internet Context. Dordrecht: Kluvert Academic Publisher, 127-152.

Ralyte, J., Deneckere, R. & Rolland, C. 2003. Towards a generic model for situational method engineering. In J. Eder & M. Missikoff (Eds.) Proc. of the 15th Int. Conf. on Advanced Information Ssystems Enginering (CAiSE'03). LNCS 2681, Berlin: Springer-Verlag, 95-110.

Ralyte, J. & Rolland, C. 2001. An assembly process model for method engineering. In K. Dittrich, A. Geppert & M. Norrie M. (Eds.) Proc. of the13th Int. Conf. on Advanced Information Systems Engineering (CAiSE'01). LNCS 2001, Berlin: Springer-Verlag, 267-283.

Ramackers, G. 1994. Integrated object modeling, an executable specification framework for business analysis and information system design. Amsterdam: Thesis Publishers, The Netherlands, Dissertation Thesis.

Ramackers, G. & Verrign-Stuart, A. 1990. First and second order dynamics in information systems. In H. G. Sol & K. M. van Hee (Eds.) Proc. of the

International Working Conf. on Dynamic Modelling of Information Systems. TU-Delf, Department of Information Systems.

Ramesh, B. & Dhar, V. 1992. Supporting systems development by capturing deliberations during requirements enginerering. IEEE Trans. on Software Engineering, Vol. 18, No. 6, 498-510.

Ramesh, B. & Jarke, M. 2001. Towards reference models for requirements traceability. IEEE Trans. on Software Engineering, Vol. 27, No. 1, 58-93.

Ramesh, R. & Whinston, A. 1994. Claims, arguments, and decisions: formalism for representation, gaming, and coordination. Information Systems Research, Vol. 5, No. 3, 294-325.

Randell, D. & Cohn, A. 1989. Modelling topological and metrical properties in physical processes. In R. Brachman, H. Levesque & R. Reiter R. (Eds.) Proc. of the First Int. Conf. on Principles of Knowledge Representation and Reasoning. San Mateo: Morgan Kaufmann, 357-368.

Reese, W. 1980. Dictionary of philosophy and religion. Atlantic Highlands, NJ: Humanities Press.

Rettig, M. & Simons, G. 1993. A project planning and development process for small teams. Comm. of the ACM, Vol. 36, No. 10, 45-55.

Rittel, H. & Webber, M. 1984. Planning problems are wicked problems. In N. Cross (Ed.) Developments in Design Methodologies. New York: John Wiley & Sons.

Roberts, T., Gibson, M., Fields, K. & Rainer, R. 1998. Factors that impact implementing a system development methodology. IEEE Trans. on Software Engineering, Vol. 24, No. 8, 640-649.

Roberts, T., Leigh, W., Purvis, R. & Parzinger, M. 2001. Utilizing knowledge links in the implementation of system development methodologies. Information and Software Technology, Vol. 43, No. 11, 635-640.

Robey, D. 1984. Conflict models for implementation research. In R. Schultz (Ed.), Management Science Implementation. New York: American Elsevier.

Rodrigues, A. & Willians, T. 1997. System dynamics in software project management: towards the development of a formal integrated framework. European Journal of Information Systems, Vol. 6, No. 1, 51-66.

Rolland, C., Nurcan, S. & Grosz, G. 2000. A decision making pattern for guiding the enterprise knowledge development process. Information and Software Technology, Vol. 42, No. 5, 313-331.

Rolland, C. & Prakash, N. 1996. A proposal for context-specific method engineering. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/WG8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 191-208.

Rolland, C., Prakash N. & Benjamen, A. 1999. A multi-model view of process modeling. Requirements Engineering, Vol. 4, No. 4, 169-187.

Rolland, C. & Proix, C. 1992. A natural language approach for requirements engineering. In P. Loucopoulos (Ed.) Proc. of 4th Int. Conf. on Advanced

Information Systems Engineering (CAiSE'92). LNCS 593, Berlin: Springer-Verlag, 257-277.

Rolland, R., Souveyet, C. & Ben Achour, C. 1998. Guiding goal modeling using scenarios. IEEE Trans. on Software Engineering, Vol. 24, No. 12, 1055-1071.

Rolland, C., Souveyet, C. & Moreno, M. 1995. An approach for defining ways-of-working. Information Systems, Vol. 20, No. 4, 337-359.

Rosch, E. 1978. Principles of categorization. In E. Rosch & B. Lloyd (Eds.) Cognition and Categorization. Hillsdale, NJ: Erlbaum, 27-48.

Rose, T. & Jarke, M. 1990. A decision-based configuration process model. In Proc. of the 12th Int. Conf. on Software Engineering. Los Alamitos: IEEE Computer Society Press, 316-325.

Rosemann, M. & Green, P. 2002. Developing a meta model for the Bunge-Wand-Weber ontological constructs. Information Systems, Vol. 27, No. 2, 75-91.

Rosemann, M. & zur Muchlen, M. 1997. Evaluation of workflow management systems – a meta model approach. In K. Siau, Y. Wand & J. Parsons (Eds.) Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'97).

Rosis De, F. & Pizzutilo, S. 1998. Formal description and evaluation of user-adapted interfaces. International Journal of Human-Computer Studies, Vol. 49, No. 2, 95-120.

Rossi, M. 1996. Evolution of OO methods: the unified case. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'96).

Rossi, M., Lyytinen, K., Ramesh, B. & Tolvanen, J.-P. 2005. Managing evolutionary method enginering by method rationale. Journal of the Association of Information Systems (JAIS), forthcoming.

Royce, W. 1970. Managing the development of large software system: Concepts and techniques. In Proc. of 9th Int. Conf. on Software Engineering. IEEE Computer Society Press, 1-9.

Rubenstein-Montano, B., Liebowitz, J., Buchwalter, J., McCaw, D., Newman, B. & Rebeck K. 2001. The Knowledge Management Methodology Team, A system thinking framework for knowledge management. Decision Support Systems, Vol. 31, No. 1, 5-16.

Ruiz, F., Vizcaino, A., Piattini, M. 2004. Garcia, F., An ontology for the management of software maintenance projects. International Journal of Software Engineering and Knowledge Engineering, Vol. 14, No. 3, 323-349.

Rumbaugh, J. 1995. What is a method, Journal of Object-Oriented Programming, Vol. 8, No. 6, 10-16, 26.

Russo, N., Hightower, R. & Pearson, J. 1996. The failure of methodologies to meet the needs of current development environments In N. Jayaratna & B. Fitzgerald (Eds.) Proc. of the 4th Conf. on Information Systems Methodologies: Lessons Learned from the Use of Methodologies, 387-394.

Russo, N., Wynekoop, J. & Walz, D. 1995. The use and adaptation of system development methodologies. In M. Khosrowpour (Ed.) Managing

Information & Communications in a Changing Global Environment, Proc. of International Conf. of International Resources Management Association (IRMA), Atlanta, Idea Group Publishing.

Ryan, K., Kronlöf, K. & Sheehan, A. 1996. Method integration. In N. Jayaratna & Fitzerald B. (Eds.) Proc. of the 4th Conf. on Information System Methodologies: Lessons Learned from the Use of Methodologies. London: British Computer Society, 235-246.

Rzevski, G. 1983. On the comparison of design methodologies. In T. Olle, H. Sol & C. Tully (Eds.) Information Systems Design Methodologies – A Feature Analysis. Amsterdam: North-Holland, 259-266.

Saarinen, T. 1990. System development methodology and project success. Information and Management, Vol. 19, No. 3, 183-193.

Sabherwal, R. & Robey, D. 1993. An empirical taxonomy of implementation processes based on sequences of events in information system development. Organization Science, Vol. 4, No. 4, 548-576.

Sabherwal, R. & Robey, D. 1995. Reconciling variance and process strategies for studying information system development. Information Systems Research, Vol. 6, No. 4, 303-327.

Saeki, M. 1998. A meta-model for method integration. Information and Software Technology, Vol. 39, No. 14, 925-932.

Saeki, M. 2003. Embedding metrics into information systems development methods: an application of method engineering technique. In J. Eder & M. Missikoff (Eds.) Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003). LNCS 2681, Berlin: Springer, 374-389.

Saeki, M., Iguchi, K., Wen-yin, K. & Shinokara M. 1993. A meta-model for representing software specification & design methods. In N. Prakash, C. Rolland & B. Pernici (Eds.) Proc. of the IFIP WG8.1 Working Conf. on Information Systems Development Process. Amsterdam: North-Holland, 149-166.

Saeki, M. & Wen-Yin, K. 1994. Specifying software specification & design methods. In G. Wijers, S. Brinkkemper & S. Wasserman (Eds.) Proc. of the 6th Int. Conf. on Advanced Information Systems Engineering (CAiSE'94). Berlin: Springer-Verlag, 353-366.

Sage, A. P. & Palmer J. D. 1990. Software systems engineering. New York: John Wiley & Sons.

Sakai, H. 1983. A method for entity-relationship behavior modeling. In C. G. Davis, S. Jajodia, P. A. Ng & R. T. (Eds.) Entity-Relationship Approach to Software Engineering. Amsterdam: North-Holland, 113-129.

Saksena, M., France, R. & Larrondo-Petric, M. 1998. A characterization of aggregation. In C. Rolland & G. Grosz (Eds.) Proc. of 5th Int. Conf. on Object-Oriented Information Systems (OOIS'98). Berlin: Springer, 11-19.

Sasso, W. 1984. The task analysis methodology: a procedure for office analysis. Technical Report, Human-Computer Interaction Lab, University of Michigan.

Saussure de, F. 1931. Grundlagen der allgemeinen Sprachwissenshaft. Berlin: de Gruyter.

Savolainen, V. 1999. Technical specification of an information system. In V. Savolainen (Ed.) Perspective of Information Systems. Berlin: Springer Verlag, 111-125.

Schank, R. 1973. Identification of conceptualizations underlying natural language. In R. Schank & K. Colby (Eds.) Computer Models of Thought and Language. San Francisco: Freeman, 187-247.

Scheer, A.-W. 1998. Business process engineering, Reference models for industrial enterprises. Berlin: Springer-Verlag.

Schipper, M. & Joosten, S. 1996. A validation procedure for information systems modeling techniques. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'96).

Schmitt, J.-R. 1993. Product modeling for requirements engineering process modeling. In N. Prakash, C. Rolland & B. Pernici (Eds.) Proc. of the IFIP WG8.1 Working Conf. on Information System Development Process. Amsterdam: North-Holland, 231- 246.

Schoebbens, P. 1993. Exceptions in algebraic specifications: on the meaning of "but". Science of Computer Programming, Vol. 20, No. 1-2, 73-111.

Schrefl, M., Tjoa, A. & Wagner, R. 1984. Comparison-criteria for semantic data models. In J. K. Aggarwal (Ed.) Proc. of the First Intenartional Conf. on Data Engineering (ICDE 1984), 120-125.

Schuster, G. & Stuckenschmidt, H. 2001. Building shared ontologies for terminology integration. In G. Stumme, A. Maedche & S. Staab (Eds.) Workshop on Ontologies (ONTO'2001).

Schön, D. 1983. The reflective practitioner – How professionals think in action? New York: Basic Books.

Schönström, M. & Carlsson, S. 2003. Methods as knowledge enablers in software development organizations. In Proc of the 11th European Conference of Information Systems (ECIS 2003), Naples.

Sciore, E. 1989. Object specialization. ACM Trans. on Information Systems, Vol. 7, No. 2, 103-122.

Searle, J. 1969. Speech acts – an essay in the philosophy of language. New York: Cambridge University Press.

Searle, J. 1979. Expression and meaning. New York: Cambridge University Press.

Searle, J. & Vanderveken D. 1985. Foundations of illocutionary logic. New York: Cambridge University Press.

Serour, M., Henderson-Sellers, B., Hughes, J., Winder, D. & Chow, L. 2002. Organizational transition to object technology: theory and practice. In Z. Bellahsene, D. Patel & C. Rolland (Eds.) Proc. of the 8th Int. Conf. on Object-Oriented Information Systems (OOIS'02). LNCS 2425, Berlin: Springer-Verlag, 229-241.

Shanaham, M. 1995. Default reasoning about spatial occupancy. Artificial Intelligence, Vol. 74, No. 1, 147-163.

Shank, G., Tansley, E. & Weber R. 2003. Using ontology to validate conceptual models. Comm. of the ACM, Vol. 46, No. 10, 85-89.

Sharfstein, B.-A. 1989. The dilemma of context. New York: New York University Press.

Shehory, O. & Sturm A. 2001. Evaluation of modeling techniques for agent-based systems. In Proc. of the 5th Int. Conf. on Autonomous Agents (AGENTS'01). Montreal, Canada: ACM Press, 624-631.

Shepard, T., Wortley, C. & Sibbald S. 1992. A visual software process language. Comm. of the ACM, Vol. 35, No. 4, 37-44.

Shoham, Y. 1987. Temporal logics in AI: semantical and ontological considerations. Artificial Intelligence, Vol. 33, No. 1, 89-104.

Short, K. 1991. Methodology integration: evolution of information engineering. Information and Software Technology, Vol. 33, No. 9, 720-732.

Shoval, P. 1996. Experimental comparisons of entity-relationship and object-oriented data models. In K. Siau & Y. Wand (Eds.) Proc. of the Workshop on Evaluation of Modelling Methods in Systems Analysis and Design (EMMSAD'96), Crete, Creece.

Shoval, P. & Shiran S. 1997. Entity-relationship and object-oriented data modeling – an experimental comparison of design quality. Data & Knowledge Engineering, Vol. 21, No. 3, 297-315.

Shubik, M. 1979. Computers and modeling. In M. Dertouzos & J. Moses (Eds.) The Computer Age: a Twenty Year View. Cambridge, MA: MIT Press, 285-305.

Simmons, R. 1973. Semantic networks: their computation and use for understanding English sentences. In R. Schank & K. Colby (Eds.) Computer Models of Thought and Language. San Francisco: Freeman, 63-113.

Simon, H. 1960. The new science of management decisions. New York: Harper & Row.

Simon, H. 1996. The sciences of the artificial. 3rd edition, Cambridge: MIT Press.

Sinha, A. & Vessey I. 1995. End-user data modeling: an ontological evaluation of relational and object-oriented schema diagrams. Working Paper, Indiana: Indiana University.

Sisk, H. 1973. Management and organization. Cincinnati: South Western Pub. Co., International Business and Management Series.

Skidmore, S., Farmer R. & Mills, G. 1992. SSADM Version 4 – Models & Methods. NCC Blackwell.

Slooten van, K. 1995. Situated methods for systems development. Enschede, University of Twente, Dissertation Thesis.

Slooten van, K. & Brinkkemper, S. 1993. A method engineering approach to information systems development. In N. Prakash, C. Rolland & B. Pernici (Eds.) Proc. of the IFIP WG8.1 Working Conf. on Information Systems Development Process. Amsterdam: North-Holland, 167-188.

Slooten van, K. & Hodes, B. 1996. Characterizing IS development projects. In S. Brinkkemper, K. Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8 WG8.1/WG8.2 Working Conf. on Method Engineering: Principles of Method Construction and Tool Support. London: Chapman & Hall, 29-44.

Slooten van, K. & Schoonhoven, B. 1994. Towards contingent information systems development approaches. In J. Zupancic & S. Wrycza (Eds.) Proc. of the 4th Int. Conf. on Information Systems Development (ISD-94), 242-253.

Smith, F. & Medin, A. 1981. Categories and concepts. Cambridge: Harward University Press.

Smith, J. & Smith, D. 1977a. Database abstraction: aggregation. Comm. of the ACM, Vol. 20, No. 6, 405-413.

Smith, J. & Smith, D. 1977b. Database abstraction: aggregation and generalization. ACM Trans. on Database Systems, Vol. 2, No. 2, 105-133.

Smith, W. 1988. Concepts and thoughts. In R. Sternberg & E. Smith (Eds.) The Psychology of Human Thought. Cambridge: Cambridge University Press.

Smolander, K. 1991. OPRR: a model for modeling systems development methods. In K. Lyytinen & V.-P. Tahvanainen (Eds.) Next Generation CASE Tools. IOS Press.

Smolander, K., Tahvanainen, V.-P. & Lyytinen, K. 1990. How to combine tools and methods in practice – a field study. In B. Steinholtz, A. Sölvberg & L. Bergman (Eds.) Proc. of the 2nd Nordic Conference CAiSE'90. Berlin: Springer-Verlag, 195-211.

Snoek, M. & Dedene, G. 2001. Core modeling concepts to define aggregation. L'Objet Software, Databases, Networks, Vol. 7, No. 1.

Snook, C. & Harrison, R. 2001. Practitioners' views on the use of formal methods: an industrial survey by structured interviews. Information and Software Technology, Vol. 43, No. 4, 275-283.

Soffer, P., Golany, B., Dori, D. & Wand, Y. 2001. Modelling off-the-shelf information systems requirements: an ontological approach. Requirements Engineering, Vol. 6, No. 3, 183-199.

Sol, H. 1983. A feature analysis of information systems design methodologies: methodological considerations. In T. Olle, H. Sol. & C. Tully (Eds.) Information Systems Design Methodologies: a Feature Analysis. Amsterdam: North-Holland, 1-8.

Sol, H. 1992. Information systems development: a problem solving approach. In W. Cotterman & J. Senn (Eds.). Challenges and Strategies for Research in Systems Development. New York: John Wiley & Sons, 151-161.

Sol, H. & Crosslin R.L. (Eds.) 1992. Dynamic Modelling of Information Systems II. Amsterdam: North-Holland.

Sommerville, I. 1998. Software engineering. 5th edition. Reading: Addison-Wesley Longman.

Somogyi, E. & Galliers R. 1987. From data processing to strategic information systems – a historical perspective. In E. Somogyi & R. Galliers (Eds.) Towards Strategic Information Systems. Tunbridge Wells, UK: Abacus Press, 2-25.

Song, X. 1997. Systematic integration of design methods. IEEE Software, Vol. 14, No. 2, 107-117.

Song, X. & Osterweil, L. 1992. Towards objective, systematic design-method comparison. IEEE Software, Vol. 9, No. 3, 43-53.

Sorenson, P., Tremblay, J.-P. & McAllister, A. 1988. The Metaview system for many specification environments. IEEE Software, Vol. 5, No. 2, 30-38.

Sowa, J. 1976. Conceptual graphs for data base interface. IBM Journal of Research and Development, Vol. 20, No. 4, 336-357.

Sowa, J. 1984. Conceptual structures: Information processing in minds and machines. Reading: Addison-Wesley.

Sowa, J. 1995. Top-level ontological categories. International Journal of Human-Computer Studies, Vol. 43, No. 5-6, 669-685.

Sowa, J. 2000. Knowledge representation – logical, philosophical, and computational foundations. Pacific Grove, CA: Brooks/Cole.

Sowa, J. & Zachman J. 1992. Extending and formalizing the framework for information system architecture. IBM Systems Journal , Vol. 31, No. 3, 590-616.

Srinivas, K., 1997. How is context represented in implicit and explicit memory. In Proc. of the 2nd European Conf. on Cognitive Science (ECCS'97), Worskhop on Context.

Staab, S., Schnurr, H.P., Studer, R. & Sure, Y. 2001. Knowledge prosesses and ontologies. IEEE Intelligent Systems, Vol. 16, No. 1, 26-34.

Stachowitz, R. 1985. A formal framework for describing and classifying semantic data models. Information Systems, Vol. 10, No. 1, 77-96.

Stamper, R. 1973. Information in business and administrative systems. New York: Wiley & Sons.

Stamper, R. 1975. Information science for systems analysis. In E. Mumford & H. Sackman (Eds.) Human Choice and Computers. Amsterdam: North-Holland, 107-120.

Stamper, R. 1978a. Towards a semantic model for the analysis of legislation. Research Report L17, London School of Economics.

Stamper, R. 1978b. Aspects of data semantics: names, species and complex physical objects. In G. Bracchi & P. Lockemann (Eds.) Information Systems Methodology. Berlin: Springer-Verlag, 291-306.

Stamper, R. 1988. Analysing the cultural impact of a system. International Journal of Information Management, Vol. 8, No. 3, 107-122.

Stamper, R. 1992a. Language and computing in organized behaviour. In R. van de Riet & R. Meersman (Eds.) Linguistic Instruments in Knowledge Engineering. Amsterdam: North-Holland, 143-163.

Stamper, R. 1992b. Signs, organizations, norms and information systems. In Proc. of the 3rd Australian Conf. on Information Systems. Department of Business Systems, Univ. of Wollongong, Australia, 21-65.

Stamper R. 1996. Signs, information, norms and information systems. In B. Holmqvist, P. Andersen, H. Klein & R. Posner (Eds.) Signs are Work:

Semiosis and Information Processing in Organisations. Berlin: De Gruyter, 349-392.

Stamper, R. 1999. Information systems as a social science: an alternative to the Frisco fomalism. In E. Falkenberg, K. Lyytinen & A. Verrijn-Stuart (Eds.) Proc. of IFIP WG8.1 Int. Working Conf. on Information System Concepts: An Integrated Discipline Emerging. Dordrecht: Kluwert Academic Publishers, 1-51.

Steenis van, H. 1990. How to plan, develop & use information systems – a guide to human qualities and productivity. New York: Dorset House Pub.

Steinmann, F. 2000. On the representation of roles in object-oriented and conceptual modeling. Data & Knowledge Engineering, Vol. 35, No. 1, 83-106.

Stolterman, E. 1992. How system designers think about design and methods: some reflections based on an interview study. Scandinavian Journal of Information Systems, Vol. 4, No. 1, 137-150.

Stolterman, E. 1994. The 'transfer of rationality', acceptability, adaptability and transparency of methods. In W. Baets (Ed.) Proc. of Second European Conf. on Information Systems (ECIS'94). Breukelen: Nijenrode University Press, 533-540.

Stowell, F. 1991. Towards client-led development of information systems. Journal of Information Systems, Vol. 1, No. 3, 173-189.

Strauss, A. 1978. Negotiations: varieties, contexts, processes, and social order. San Francisco: Jossey-Bass.

Sturm, A. & Shehory, O. 2004. A Framework for evaluating agent-oriented methodologies. In P. Giorgini, B. Henderson-Sellers & M. Winikoff (Eds.) Proc. of the Fifth Int. Bi-Conference on Agent-Oriented Information Systems (AOIS 2003). LNCS 3030, Berlin: Springer-Verlag, 95-111.

Su, X. & Ilebrekke, L. 2002. A comparative study of ontology languages and tools. In A. Banks Pidduck, Mylopoulos, C. Woo & T. Ozsu (Eds.) Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE'2002). Berlin: Springer-Verlag, 761-765

Sugumaran, V. & Storey V. 2002. Ontologies for conceptual modeling: their creation, use, and management. Data & Knowledge Engineering, Vol. 42, No. 3, 251-271.

Sumner, M. & Sitek J. 1986. Are structured methods for systems analysis and design beign used? Journal of Systems Management, Vol. 4, No. 2, 18-27

Sungren B. 1975. Theory of data bases. New York: Petrocelli/Charter.

Sutcliffe, A. 1996. A conceptual framework for requirements engineering. Requirements Engineering, Vol. 1, No. 3, 170-189.

Sutcliffe, A. 2000. Domain analysis for software reuse. The Journal of Systems and Software, Vol. 50, No. 3, 175-199.

Swartout, B., Ramesh, P., Knight, K. & Russ, T. 1997. Toward distributed use of large-scale ontologies. In A. Farquhar & M. Gruninger (Eds.) Proc. of AAAI Symposium on Ontological Engineering. Stanford, Technical Report SS-97-06.

Swede van, V. & van Vliet, J. 1993. A flexible framework for contingent information systems modeling. Information and Software Technology, Vol. 35, No. 9, 530-548.

Söderström, E., Andersson, B., Johannesson, P., Perjons, E. & Wangler, B. 2002. Towards a framework for comparing process modeling languages. In A. Banks Pidduck, J. Mylopoulos, C. Woo & T. Ozsu (Eds.) Proc. of the 14th Intern Conf on Advanced Information Systems Engineering (CAiSE'2002). Berlin: Springer-Verlag, 600-611.

Tardieu, H. 1992. Issues for dynamic modelling through recent development in European methods. In H. Sol & R. Crosslin (Eds.) Dynamic Modelling of Information Systems II. Amsterdam: North-Holland, 3-23.

Teichroew, D. & Heshey, E. 1977. PSL/PSA: A computer-aided technique for structured documention and analysis of information processing systems. IEEE Transactions on Software Engineering, Vol. 3, No. 1, 41-48.

Teichroew, D., Macasovic, P., Hershey, E. & Yamato, Y. 1980. Application of the Entity-Relationship Approach to information processing systems modeling. ISDOS-project, The University of Michigan.

Thayer, R. 1987. Software engineering project management – a top-down view. In R. Thayer (Ed.) Tutorial: Software Engineering Project Management. IEEE Computer Society Press, 15-56.

Thomas, I. & Nejmeh B. 1992. Definitions of tool integration for environments. IEEE Software, Vol. 9, No. 2, 29-35.

Thomson, I. 1990. SSADM: the last word. Government Computing, Vol. 4, No. 5, 28-29.

Tiles, J. 1981. Things that happen. Aberdeen University Press, Great Britain.

Tollow, D. 1996. Experiences of the pragmatic use of structured methods in public sector projects. In N. Jayaratna & B. Fitzgerald (Eds.) Proc. of the 4th Conf. on Information System Methodologies: Lessons Learned from the Use of Methodologies. London: British Computer Society, 177-186.

Tolvanen, J.-P. 1995. Incremental method development for business modeling: an action reearch case study. In G. Grosz (Ed.) Proc. of the 6th Workshop on Next Generation CASE Tools, University of Paris, 79-98.

Tolvanen, J.-P. 1998. Incremental method engineering with modeling tools – Theoretical principles and empirical evidence. Jyväskylä Studies in Computer Science, Economics and Statistics, No. 47, University of Jyväskylä, Finland, Dissertation Thesis.

Tolvanen, J.-P., Gray, J. & Rossi, M. (Eds.) 2003. Proc. of the 3rd OOPSLA Workshop on Domain-Specific Modeling (DSM'03). Computer Science and Information Systems Reports, Technical Repots TR-28, University of Jyväskylä, Finland.

Tolvanen, J.-P. & Lyytinen, K. 1993. Flexible method adaptation in CASE – the metamodeling approach. Scandinavian Journal of Information Systems, Vol.5, 51-77.

Tolvanen, J.-P., Rossi, M. & Liu H. 1996. Method engineering: current research directions and implications for future research. In S. Brinkkemper, K.

Lyytinen & R. Welke (Eds.) Proc. of the IFIP TC8, WG8.1/8.2 Working Conf on Method Engineering. London: Chapman & Hall, 296-317.

Tolvanen, J.-P., Sprinkle, J. & Rossi, M. (Eds.) 2004. Proc. of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04). Computer Science and Information Systems Reports, Technical Repots TR-33, University of Jyväskylä, Finland.

Tomiyama, T., Kiriyama, T., Takeda, H., Xue, D. & Yoshikaya, H. 1989. Metamodel: a key to intelligent CAD systems. Research in Engineering Development, Vol. 1, No. 1, 19-34.

Tracz, W., Coglianese, L. & Young, P. 1993. A domain-specific software architecture engineering process outline. Software Engineering Notes, Vol. 18, No. 2, 40-49.

Tran, Q.-N., Low, G. & Williams, M.-A. 2003. A Feature Analysis Framework for Evaluating Multi-agent System Development Methodologies. In N. Zhong & Z. Ras, S. Tsumoto & E. Suzuki (Eds.) Proc. of the 14th Int. Symposium of Foundations of Intelligent Systems (ISMIS 2003). Berlin: Springer-Verlag, 613-617.

Truex, D., Baskerville, R. & Klein, H. 1999. Growing systems in emergent organizations. Comm. of the ACM, Vol. 42, No. 8, 117-123.

Truex, D., Baskerville, R. & Travis, J. 2000. Amethodological systems development: the deferred meaning of systems development methods. Accounting, Management & Information Technology, Vol. 10, No. 1, 53-79.

Tsichritzis, D. & Lochovsky, F. 1982. Data models. Englewood Cliffs: Prentice-Hall.

Tudor, D.J. & Tudor, I.J. 1995. Systems analysis & design: A comparison of structured methods. Oxford: NCC Blackwell.

Tuikka, T. 2002. Towards computational instrumens for collaborating product concept designers. University of Oulu, Finland: Oulu University Press, Dissertation Thesis.

Tun, T. T. & Bielkowicz, P. 2003. A Critical Assessment of UML using an Evaluation Framework. In K. Siau, T.Halpin & J. Krogstie (Eds.) Proc. of the 8th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'03), 29- 38.

Turski, W. & Maibaum, T. 1987. The specification of computer programs. Workingham: Addison-Wesley.

Uschold, M. 1996. Building ontologies: towards a unified methodology. In Proc. of 16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems. Cambridge, UK.

Uschold, M. & Gruninger, M. 1996. Ontologies: principles, methods and applications. Knowledge Engineering Review, Vol. 11, No. 2, 93-155.

Uschold, M. & King, M. 1995. Towards a methodology for building ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI'95, Montreal, Canada.

Uschold, M., King, M., Maralee, S. & Zorgios, Y. 1998. The Enterprise Ontology. The Knowledge Engineering Review, Vol. 13, No. 1, 31-89.

Varzi, A. 1996. Parts, wholes, and part-whole relations: The prospects of mereotopology. Data & Knowledge Engineering, Vol. 20, No. 3, 259-286.

Vasconcelos de, F. & Werner, C. 1998. Organizing the software development process knowledge: an approach based on patterns. International Journal of Software Engineering and Knowledge Engineering, Vol. 8, No. 4, 461-482.

Venable, J. 1993. CoCoA: A conceptual data modeling approach for complex problem domains. State University of New York, Binghampton, USA, Dissertation Thesis.

Verrijn-Stuart, A. 1989. Some reflections on the Namur conference on information system concepts. In E. Falkenberg & P. Lindgreen (Eds.) Information Systems Concepts: An In-Depth Analysis. Amsterdam: North-Holland, ix-x.

Verrijn-Stuart, A. 1995. Business model representations. In E. Falkenberg, W. Hesse W. & A. Olive (Eds.) Proc. of the IFIP WG8.1 Working Conf. on Information System Concepts– Towards a Consolidation of Views. London: Chapman & Hall, 266-281.

Verrijn-Stuart, A. & Ramackers, G. 1992. Model integration in information planning tools. In P. Loucopoulos (Ed.) Proc. of the 4th Int. Conf. on Advanced Information Systems Engineering (CAiSE'92). Berlin: Springer-Verlag, 481-493.

Veryard, R. 1987. Information management: Implementing a methodology. Information and Software Technology, Vol. 29, No. 9, 469-474.

Vessey, I. & Conger, S. 1994. Requirements specification: learning object, process, and data methodologies. Comm. of the ACM, Vol. 37, No. 5, 102-113.

Vidgen, R. 2002. Constructing a web information system development methodology. Information Systems Journal, Vol. 12, No. 3, 247-261.

Vincent, P. & Sherwood-Smith, M. 1992. Streamlining office work, The OSSAD Method. Le Bulletin de I'UATI, No. 2.

Vlasblom, G., Rijsenbrij, D. & Glastra, M. 1995. Flexibilization of the methodology of system development. Information and Software Technology, Vol. 37, No. 11, 595-607.

Vonk, R. 1990. Prototyping: the effective use of CASE technology. London: Prentice-Hall.

Vygotsky, L. 1978. Mind in society: the development of higher psychological processes. Compiled from several sources and edited by M. Cole M., V. John-Steiner & S. Scribner, Harvard University Press.

Wagner, G. 1988. Implementing abstraction hierarchies. In C. Batini (Ed.) Proc. of the 7th Int. Conf. on Entity-Relationship Approach. Amsterdam: North-Holland, 267-300.

Walls, J., Widmeyer, G. & El Sawy, O. 1992. Building an information system design theory for vigilant EIS. Information Systems Research, Vol. 3. No. 1, 36-59.

Wand, Y. 1988a. An ontological foundation for information systems design theory. In B. Pernici & A. Verrijn-Stuart (Eds.) Proc. of the IFIP 8.4 Working Conf. on Office Information Systems: The Design Process. Amsterdam: North-Holland, 201-222.

Wand, Y 1988b. A proposal for a formal model of objects. In W. Kim & F. Lockhovsky (Eds.) Object-Oriented Concepts, Databases, and Applications. Reading: Addison-Wesley, 537-559.

Wand, Y. 1996. Ontology as a foundation for meta-modelling and method engineering. Journal of Information and Software Technology, Vol. 38, No. 4, 281-288.

Wand, Y., Monarchi, D., Parson, J. & Woo, C. 1995a. Theoretical foundations for conceptual modeling in information systems development. Decision Support Systems, Vol. 15, No. 4, 285-304.

Wand, Y., Storey, V. & Weber, R. 1999. An ontological analysis of the relationship construct in conceptual modeling. ACM Trans. on Database Systems, Vol. 24, No. 4, 494-528.

Wand, Y. & Weber, R. 1989. An ontological evaluation of systems analysis and design methods. In E. Falkenberg & P. Lindgreen (Eds.) Proc. of the IFIP TC8 /WG 8.1 Working Conference on Information Systems Concepts: an In-Dept Analysis. Amsterdam: North-Holland, 79-107.

Wand, Y. & Weber, R. 1990a. An ontological model of an information system. IEEE Trans. on Software Engineering, Vol. 16, No. 11, 1282-1292.

Wand, Y. & Weber, R. 1990b. Mario Bunge's ontology as a formal foundation for information systems concepts. In P. Weingartner & G. Dorn (Eds.) Studies on Mario Bunge's Treatise, Poznan Studies in the Philosophy of the Science and the Humanities, Vol. 18, 123-150.

Wand, Y. & Weber, R. 1993. On the ontological expressiveness of information systems analysis and design grammars. Journal of Information Systems, Vol. 3, No. 4, 217-237.

Wand, Y. & Weber, R. 1995b. On the deep structure of information systems. Information Systems Journal, Vol. 5, N. 3, 203-223.

Wang, X. & Chan, C. 2001. Ontology modeling using UML. In Y. Wang, S. Patel & R. Johnston (Eds.) Proc. of 7th Int. Conf. on Object-Oriented Information Systems (OOIS'2001). Berlin: Springer-Verlag, 59-70.

Wangler, B., Wohend, R. & Öhlund, S.-T. 1993. Business modeling and rule capture in a CASE environment. In S. Brinkkemper & F. Harmsen (Eds.) Proc. of the 4th Workshop on the Next Generation of CASE Tools. University of Twente, 189-204.

Wartik, S. & Prieto-Diaz, R. 1992. Criteria for comparing reuse-oriented domain analysis approaches. International Journal of Software Engineering and Knowledge Engineering, Vol. 2, No. 3, 403-431.

Wastell, D. 1996. The fetish of technique: methodology as a social defence. Information Systems Journal, Vol. 6, No. 1, 25-40.

Wastell, D. & Newman, M. 1993. The behavioral dynamics of information systems development: a stress perspective. Accounting, Management & Information Technology, Vol. 3, No. 2, 121-148.

Watson, H. & Wood-Harper, T. 1995. Methodology as metaphor: the practical basis for multiview methodology. Information Systems Journal, Vol. 5, No. 3, 225-231.

Weber, R. 1997. Ontological foundations of information systems. Australia: Coopers and Lybrand.

Weber, R. & Zhang Y. 1996. An analytical evaluation of NIAM's grammar for conceptual schema diagrams. Information Systems Journal, Vol. 6, No. 2, 147-170.

Webster 1989. Webster's Encyclopedic Unabridged Dictionary of the English Language. New York: Gramercy Books.

Wegner, P. 1987. Dimensions of object based language design. In N. Meyrowitz (Ed.) Proc. of Conf. on Object-Oriented Programming, Systems and Languages (OOPSLA'87), SIGPLAN Notices, Vol. 22, No. 12, 168-182.

Weick, K. E. 1995. Sensemaking in organizations. California: Sage Publications.

Weide, B. & Defazio, S. 1993. A framework for modeling software engineering processes. International Journal of Software Engineering and Knowledge Engineering, Vol. 3, No. 3, 531-568.

Weinberger, H., Te'eni, D. & Frank, A. 2003. Ontologies of organizational memory as a basis for evaluation. In Proc. of the 11th European Conf. of Information Systems (ECIS 2003), Naples.

Welke, R. 1977. Current information system analysis and design approaches: framework, overview, comments and conclusions for large – complex information system education. In R. Buckingham (Ed.) Education and Large Information Systems. Amsterdam: North-Holland, 149-166.

Welke, R. 1981. IS/DSS: DBMS support for information systems development. ISRAM WP-8105-1.0, McMaster University, Hamilton.

Welke, R. 1982. Current work on a methodology-based theory of information systems. In N. Björn-Andersen (Ed.) Towards Tools for Transition Systems Design Methodologies, IFIP, Den Haag, 17-21.

Welke, R. 1988. The CASE repository. More than another database application. Ann Arbor: MeaSystem Ltd..

Welke, R. & Konsynski, B. 1982. Technology, methodology & information systems: a tripartite view. Data Base, Vol. 14, No. 1, 41-57.

White, J. 1982. A decision tool for assisting with the comprehension of large software systems. In H.-J. Schneider & A. Wasserman (Eds.) Automated Tools for Information System Design. Amsterdam: North-Holland, 49-65.

Wieringa, R. 1989. Three roles of conceptual models in information system design and use. In E. Falkenberg & P. Lindgren (Eds.) Proc. of the IFIP TC8/WG8.1 Working Conf. on Information Systems Concepts: An In-Dept Analysis. Amsterdam: North-Holland, 31-52.

Wieringa, R. 1999. A survey of structured and object-oriented software specification methods and techniques. ACM Computing Survey, Vol. 30, No. 4, 459-527.

Wieringa, R., De Jong, W. & Spruit, P. 1995. Using dynamic classes and role classes to model object migration. Theory and Practice of Object Systems, Vol. 1, No. 1, 61-83.

Wieringa, R. & Dubois, E. 1998. Integrating semi-formal and formal software specification techniques. Information Systems, Vol. 23, No. ¾, 159-178.

Wiggins, D. 1980. Sameness and substance. Oxford: Blackwell.

Wijers, G. 1991. Modelling support in information systems development. Delft University of Technology, Amsterdam: Thesis Publishers, Dissertation Thesis.

Wild, C., Maly, K. & Liu, L. 1991. Decision-based software development. Software Maintenance: Research and Practice, Vol. 3, No. 1, 17-43.

Wilesky, R. 1983. Planning and understanding. Reading: Addision-Wesley.

Wilks, Y. 1977. Good and bad arguments about semantic primitives. D.A.I Research Report No. 42, University of Essex.

Winston, M., Chaffin, R. & Hermann, D. 1987. A taxonomy of part-whole relations. Cognitive Science, Vol. 11, No. 4, 417-444.

Woo, C. & Chang, M. 1992. An approach to facilitate the automation of semi-structured and recurring negotiations in organizations. Journal of Organizational Computing, Vol. 2, No. 1, 47-76.

Wood-Harper, A. & Fitzgerald, G. 1982. A taxonomy of current approaches to systems analysis. The Computer Journal, Vol. 25, No. 1, 12-16.

Wooldridge, M. & Jennings, N. 1995. Intelligent agents: theory and practice. The Knowledge Engineering Review, Vol. 2, No. 10, 115-152.

Wordsworth, J. 1996. Software engineering with B. Reading: Addison-Wesley/Longman.

Wordsworth, J. 1999. Getting the best from formal methods. Information and Software Technology, Vol. 41, No. 14, 1027-1032.

Workflow Management Coalition 1999. Terminology & Glossary, WFMC-TC-1011, Feb-1999, version 3.0 [Referred on 14.11.2004]. Available at URL: <http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf>.

Wright von, G. 1971. Explanation and understanding. London: Routledge & Kegan Paul.

Wynekoop, J. & Russo, N. 1997. Studying system development methodologies: an examination of research methods. Information Systems Journal, Vol. 7, No. 1, 47-65.

Yadav, S., Shaw, N., Webb, L. & Sutcu, C. 2001. Comments on "Factors that impact implementing a system development methodology". IEEE Trans. On Software Engineering, Vol. 27, No. 3, 279-281.

Yang, O., Halper, M., Geller, J. & Perl, Y. 1994. The OODB ownership relationship. In D. Patel, Y. Sun & S. Patel (Eds.) Proc. of Conf. on Object-Oriented Information Systems (OOIS'94). Berlin: Springer-Verlag, 278-291.

Yates, J. & Orlikowski, W. 1992. Genres of organizational communication: a structurational approach to studying communication and media. Academy of Management Review, Vol. 8, 299-326.

Yourdon, E. 1989. Modern structured analysis. Englewood Cliffs: Prentice-Hall.

Yourdon, E. & Constantine, L. 1979. Structured design: fundamentals of a discipline of computer program and systems design. Englewood Cliffs: Prentice Hall.

Yu, E. & Mylopoulos, J. 1995. From E-R to o"A-R" – modelling strategic actor relationships for business process reengineering. International Journal of Intelligent & Cooperative Information Systems, Vol. 4, No 2/3, 125-144.

Yu, E. & Mylopoulos, J. 1997. Modelling organizational issues for enterprise integration. In K. Kosanke & J. Nell (Eds.) Proc. of Int. Conf. on Enterprise Integration and Modeling Technology (ICEIMT'97). Berlin: Springer-Verlag, 529-538.

Zachman, J. 1987. A framework for information systems architecture. IBM Systems Journal, Vol. 26, No. 3, 276-292.

Zdonik, S. & Maier, D. (Eds.) 1990. Readings in object-oriented database systems. San Mateo: Morgan Kaufmann.

Zhang, Z. & Lyytinen, K. 2001. A framework for component reuse in a metamodelling-based software development. Requirements Engineering, Vol. 6, No. 2, 116-131.

Zhou, L., Booker, Q. & Zhang, D. 2002. ROD – Towards rapid ontology development for underdeveloped domains. In Proc. of the 35th Hawaii International Conference on Systems Sciences.

Zhou, Q. & Fikes, R. 2000. A reusable time ontology. Research Report, Knowledge Systems Laboratory, Stanford University.

Zhu, Z. 2002. Evaluating contingency approaches to information systems design. International Journal of Information Management, Vol. 22, No. 5, 343-356.

Zisman, M. 1977. Representation, specification and automation of office procedures. The Wharton School, University of Pennsylvania, Dissertation Thesis.

Zloof, M. 1981. QBE/OBE: A language for office and business automation. IEEE Computer, Vol. 14, No. 5, 13-22.

Zultner, R. 1993. TQM for technical teams. Comm. of the ACM, Vol. 36, No. 10, 79-91.

APPENDIX 1: VOCABULARY

This appendix contains the integrated vocabulary of OntoFrame. The terms with the definitions are categorized according to the structure of OntoFrame. In the definitions the concerned terms are written in bold and the terms already defined in italics. This way we concretely show how the concepts signified by the terms are related to one another.

I. Core Ontology

The **core ontology** provides the key concepts and constructs for conceiving, understanding, structuring and representing fundamentals in reality.

I.1 Generic Ontology

The **generic ontology** provides the most generic concepts from which the concepts of all the other component ontologies in Ontoframe have been derived.

Reality
    Reality is anything that exists, has existed or will (possible) exist.

Subjective reality
    Subjective reality (or the perceived reality) is *reality* that is the result from our mental processes.

Physical reality
    Physical reality (or shortly reality) is *reality* that is independent of any human thinking. It is the source of sense data, which we obtain, and it is thus external to us.

Thing
    A **thing** means any phenomenon in the *physical* or *subjective reality*.

Property
    A **property** is a *thing* that is used to characterize other *thing*(s).

Characterized thing
    A **characterized thing** is a *thing* that is characterized by at least one *property*.

Relationship
    A **relationship** is a *thing* that relates two or more *characterized things* together, each one associated with one *property* characterizing the role of that *thing* within that *relationship*.

Role
    A **role** is a *property*, that reflects a position the *thing* holds, or a function the *thing* conducts, in the *relationship*.

Point of view

A **point of view** is a *thing* by which some *things* or some *properties* of *thing(s)* are selected because they are more relevant than the others.

Framework

A **framework** is a *thing* that guides a human being to select the *points of view* that are the most appropriate for the case or the problem at hand.

I.2 Semiotic Ontology

The **semiotic ontology** provides concepts and constructs to recognize semiotic things in reality. It specializes things according to the semiotic framework.

Concept

A **concept** is a *thing*, some kind of word of mind that refers to a referent (*thing*).

Construct

A **construct** is a composed of related *concepts*.

Referent

A **referent** is a *thing* in *reality* to which a *concept* refers.

Sign

A **sign** is a *thing* that can stand for something else.

I.3 Intension/Extension Ontology

The **intension/extension ontology** provides *concepts* and *constructs* to specialize the notion of a *concept* into more specific notions such as basic concept, derived concept, individual concept, generic concept, etc.

Intension

An **intension**, or comprehension of a *concept*, consists of all its *predicates*.

Predicate

A **predicate** is a *concept*, which is used to characterize the (original*) concept.*

Extension

An **extension** of a *concept* is the set of all (referent) *things* to which the *intension* of the *concept* applies.

Population

A **population** of a *concept* is the set of the existing (*referent*) *things* to which the *intention* of the *concept* applies.

Basic concept

A **basic concept** is a *concept* the *intension* of which is specified without using other *concepts* in question.

Derived concept

A **derived** concept is a *concept* the *intension* of which is derived from the *predicates* of other *concepts*.

Abstract concept

An **abstract concept** is a *concept* that has no *referent things*.

Concrete concept
A **concrete concept** is a *concept* that is not *abstract*.

Individual concept
An **individual concept,** or a particular, is a *concept* that can only refer to one *thing*.

Generic concept
A **generic concept,** or a universal, is a *concept* that can refer to many *things*.

Type
A **type** is a *generic concept*.

Instance
An **instance** is a member of the *extension* of a *type*.

I.4 Language Ontology

The **language ontology** provides *concepts* and *constructs* to specify the syntax and semantics of a language.

Language
A **language** is an abstract *thing* that is used in communication among people, between people and computers, or among parts of the computers.

Abstract syntax
An **abstract syntax** of a *language* gives the conceptual components of a *language* and rules for connecting them, leaving out representational details.

Concrete syntax
A **concrete syntax** of a *language* gives notational elements, called the symbols in the vocabulary of a *language*, and rules for connecting them with one another and with the *concepts* (cf. signification rules).

Semantics
Semantics of a *language* defines the relations of symbols to the *referents* to which the symbols are applicable.

Vocabulary
A **vocabulary** of a *language* is a non-empty and finite set of symbols.

Symbol
A **symbol** is a special *sign* used as an undividable part of an expression.

Expression
An **expression** is a *sign* of a language and a non-empty and finite "arrangement" of *symbols* taken from a *vocabulary*, constricted by the *syntax* and *semantics* of the *language*.

Formal language
A **formal language** is a *language* with a precisely defined *syntax* and *semantics*.

Semi-formal language
A **semi-formal language** is a *language* with a precisely defined *syntax*.

Informal language
An **informal language** is a *language* that is neither formal nor semi-formal.

Label
A **label** is an elementary *expression* used to signify a particular *concept* in an elementary way.

Proper name
A **proper name** is a *label* signifying an *individual concept* or a particular.

Common noun
A **common noun** is a *label* signifying a *generic concept* or a universal.

I.5 State Transition Ontology

The **state transition ontology** provides the *concepts* and *constructs* for conceiving static and dynamic *things* in *reality.*

State
A **state** is a *thing*, which is seen to have some duration.

Transition
A **transition** is a binary *relationship* between two different *things*, called the pre-state and the post-state of that transition.

Event
An **event** is a *thing*, which may trigger a *transition* from the pre-*state* to the post-*state*.

Transition structure
A **transition structure** is composed of related *transitions.*

Composite transition
A **composite transition** is a *transition structure* with a unique pre-*state* and a unique post-*state*.

Elementary transition
An **elementary transition** is a *transition* that does not contain any *transition structure*.

Life cycle
A **life cycle** of a *thing* consists of all the *states*, *transitions* and *events* that are related to the existence of a *thing*.

1.6 UoD Ontology

The **UoD ontology** provides the *concepts* and *constructs* for perceiving and conceiving the UoD, UoD states, UoD behavior, and UoD evolution from a certain *point of view.*

Universe of discourse (UoD)
A **universe of discourse (UoD)** is a *subjective reality* that is relevant from the *point of view* adopted.

UoD state
A **UoD state** is composed of the related *states* of all those *things* that are included in the *UoD*.

UoD behavior

A **UoD behavior** is composed of extensional transitions among the *UoD states.*

UoD evolution

A **UoD evolution** is composed of intensional *transitions* among the *UoD states.*

I.7 Abstraction Ontology

The **abstraction ontology** provides *concepts* and *constructs* for abstraction.

Abstraction

Abstraction is a principle by which irrelevant *things* are ignored and the *things* relevant to understanding some problem of interest are uncovered.

Concretizing

Concretizing is the principle inverse to *abstraction*.

First-order abstraction

First-order abstraction is *abstraction* that concerns the *concept things* and their abstraction *relationships*.

Second-order abstraction or **predicate abstraction**

Second-order abstraction, or **predicate abstraction,** means *abstraction*, which mainly concerns the *predicates* of *concept things* and their abstraction *relationships.*

Classification

Classification is the principle of *abstraction* by which the *concept*, called the *type*, is generated from other *concepts*, called *instances*.

Instantiation

Instantiation is the principle inverse to *classification*.

instanceOf relationship

An **instanceOf relationship** is the *relationship* between an *instance* and its *type.*

Meta type

A **meta type** is a *type*, *instances* of which are *types*.

Objective classification

An **objective classification** is a result of *classification* in which for a *type* there is only one *type extension*.

Subjective classification

A **subjective classification** is a result of *classification* in which for a *type* there may be several (subjective) *type extensions*.

Permanent classification

A **permanent classification** is a result of *classification* that does not change in time, that is to say, there is no *instance*, which refers to more than one *referent*.

Evolving classification

An **evolving classification** is a result of *classification* that changes in time, that is to say, there may be an *instance*, which refers to more than one *referent*.

Strict classification

A **strict classification** is a result of *classification* in which for each *instance* there is certain *type*.

Non-strict classification

A **non-strict classification** is a result of *classification* in which there are *instances* for which there are no *types*.

Factual predicate

A **factual predicate** is a *predicate* that mainly contain *individual concepts*.

Definitional predicate

A **definitional predicate** is a *predicate* that is composed of solely *generic concepts* expressed in *common nouns*.

Intensional predicate derivation

Intensional predicate derivation means that each *predicate* of a *type* is expected to apply to the corresponding *instance concepts*.

Extensional predicate derivation

Extensional predicate derivation means that *factual predicates* of a *type* can be derived from the *factual predicates* of the *instances*.

Generalization

Generalization is the principle of *abstraction* by which differences between some *types*, called **subtypes**, are suppressed and a new *type*, called a **supertype,** is generated based on the commonalities of the *subtypes*.

Specialization

Specialization is the principle of *abstraction*, inverse to *generalization*, to concretize *subtypes* from the *supertype*.

isA relationship

An **isA relationship** means the *relationship* between a *subtype* and its *supertype*.

Superset

A **superset** is the *extension* of the *supertype*.

Subset

A **subset** is the *extension* of a *subtype*.

One-type specialization

One-type specialization means *specialization* in which for each *supertype* there is only a *subtype*.

Hierarchical specialization

Hierarchical specialization means *specialization* in which for each *supertype* there are several *subtypes*.

Lattice specialization

Lattice specialization means *specialization* in which for each *subtype* there may be two or more *supertypes*.

Total specialization

Total specialization means *specialization* in which for each *supertype referent* there is always one *subtype referent*.

Partial specialization

Partial specialization means *specialization* in which there is a *supertype referent* for which there is no *subtype referent*.

Disjoint specialization

Disjoint specialization means *specialization* in which the *extensions* of the *subtypes* are disjoint.

Overlapping specialization

Overlapping specialization means *specialization* in which the *extensions* of the *subtypes* overlap.

Composition

Composition is the principle of *abstraction* by which a **type**, called a **whole type**, is composed of other *types*, called **part types**, or a **whole instance** is composed of related **part instances**.

Decomposition

Decomposition is the principle inverse to *composition*, by which a *whole* (*type*) is decomposed into inter-related *part(s) (types)*.

partOf relationship

A **partOf relationship** means the *relationship* between a *part (type)* and its *whole (type*). The parts in the whole are related to one another.

Syntactic composition

Syntactic composition means *composition*, which deals with *sign things.*

Semantic composition

Semantic composition means *composition*, which deals with *non-sign things*.

Total exclusive composition

Total exclusive composition means *composition* in which a *thing* can be a *part* of only one *whole*.

Arbitrary shared composition

Arbitrary shared composition means *composition* in which a *thing* can be a *part* in arbitrary many *wholes.*

Selectively exclusive composition

Selectively shared composition means *composition* in which a *thing* can be a *part* of one *whole* but of more than one alternative *type.*

Optional composition

Optional composition means *composition*, in which there may be *things* of some *part type* that are related to no *things* of the *whole type*.

Essential relationship

An **essential relationship** is a *partOf relationship* between a *part type* and a *whole type* if each *part instance* must be connected to at least one arbitrary *whole instance* of that *type*.

Immutable relationship

An **immutable relationship** is a *partOf relationship* between a *part type* and a *whole type,* if the *parts of* that *part type* are permanently related to the *whole* since its "birth".

Homogeneous whole

A **homogeneous whole** is a *thing* that is composed of *things* of one *part type.*

Heterogeneous whole

A **heterogeneous whole** is a *thing* that is composed of several *part types.*

Single-part whole

A **single-part whole** is a *thing* that contains only one *thing* of certain *part type.*

Multi-part whole

A **multi-part whole** is a *thing* that contains several *things* of a certain *part type.*

Flexible-structure whole

A **flexible-structure whole** is a *thing* in which parts of some *part types* can be missing.

Fixed-structure whole

A **fixed-structure whole** is a *thing,* which is composed of *parts* of all the defined *part types.*

Grouping

Grouping is the principle of *abstraction* by which a *concept,* called a **group type**, is generated from other *concepts,* called **member types**, or to abstract a **group instance** from **member instances**.

Individualization

Individualization is the principle inverse to *grouping* by which a *member (type)* is distinguished from a *group (type)* for a more detailed consideration.

memberOf relationship

A **memberOf relationship** means the *relationship* between a *member (type)* and a *group (type).*

Homogeneous grouping

Homogeneous grouping means *grouping* in which for a *group type* there is only one *member type.*

Heterogeneous grouping

Heterogeneous grouping means *grouping* in which a *group* can be formed from *members* of several *member types.*

Categorical grouping

Categorical grouping means *grouping* in which a *member type* is related to one *group type* at a time.

Shared grouping

Shared grouping means *grouping* in which a *thing* can be a *member type* of several *group types.*

Disjoint grouping

Disjoint grouping means *grouping* in which an *instance* cannot be a *member* of more than one *group* (of the same or different *type*).

Overlapping grouping

Overlapping grouping means *grouping* in which an *instance* is allowed to be a *member* of several *groups* (of the same of different *type*).

Mandatory grouping

Mandatory grouping means *grouping* in which each *member* must belong to some *group*.

Optional grouping

Optional grouping means *grouping* in which an *instance* can exist without any *memberOf relationship.*

Predicate classification

Predicate classification means *predicate abstraction* by which *predicate instances* are definitionalized into a *predicate type*.

Predicate instantiation

Predicate instantiation means *predicate concretizing* by which a *predicate type* is factualized into *predicate instances*.

Predicate generalization

Predicate generalization means *predicate abstraction* by which special features of predicate *subtypes* are ignored in order to uncover the features common to all the *predicate subtypes*.

Predicate composition

Predicate composition means *predicate abstraction* by which a *predicate* as an entire *construct*, called a *predicate whole (type)*, rather than its *predicate part(s) (types)* is/are examined.

Predicate grouping

Predicate grouping means *predicate abstraction* by which a *predicate group (type)* rather than its *predicate member(s) (types)* is/are examined.

II. Context ontology

The **context ontology** provides *concepts* and *construct* for conceiving, understanding, structuring, and representing *things* as contexts and/or within *contexts*.

Approach

An **approach** provides generalized principles, which help us conceive reality, recognize problems and/or find potential solutions in it.

Contextual approach

A **contextual approach** is a conception-oriented *approach*, which serves the recognition, understanding and specification of the purposes, meanings, and effects of *things*, through considering them to be contexts and/or *parts* within contexts.

Context

A **context** is a conceptual or intellectual *construct* that can help us understand, analyze, and design the natures, meanings, and effects of more elementary *things* in the concerned environment or circumstances. It is a *whole*, which is determined by the focal *thing*(s) of which making sense is important. It is composed of highly related *things*, each of which represents certain contextual domain.

Contextual concept

A **contextual concept** is a *concept*, which belongs to some of the contextual domains.

Contextual relationship

A **contextual relationship** is a *relationship*, which connects two or more *contextual concepts*. Contextual relationships contain intra-domain, inter-domain, and inter-context relationships, explicitly or implicitly defined.

Intra-domain relationship

An **intra-domain relationship** is a *contextual relationship*, which associate two or more contextual *concepts* of the same contextual domain.

Inter-domain relationship

An **inter-domain relationship** means a *contextual relationship* that associates two or more *contextual concepts* of different contextual domains.

Implicit contextual relationship

An **implicit contextual relationship** is a *contextual relationship* that can be derived from other basic or *implicit contextual relationships*.

Contextual framework

A **contextual framework** is a *framework*, which is composed of *contextual concepts* related with one another through *contextual relationships*, and which is used to conceive *things* within *contexts* and/or as *contexts*.

Contextual role

A **contextual role** is a *role*, which a *thing* plays when being *part* of the *context*.

Goal-producing context

A **goal-producing context** is a *context* that produces something, which is used as a goal statement or a requirement in another *context*.

Actor-producing context

An **actor-producing context** is a *context* that ″produces″ objects (e.g. more skilled persons), which act as actors in another *context*.

Rule-producing context

A **rule-producing context** is a *context* that produces objects, which are used as rules in another *context* (e.g. the method engineering context vs. the ISD context).

Object-producing context

An **object-producing context** is a *context* that produces objects (e.g. services), which are utilized in another *context* (cf. the IS context vs. the business system context).

Facility-producing context

A **facility-producing context** is a *context* that produces objects, which are utilized as tools or resources in another *context* (cf. the ISD context producing software vs. the business context).

Location-producing context

A **location-producing context** is a *context* that produces objects, which are used as locations in another *context*.

Purpose domain

The **purpose domain** consists of those *concepts* and *constructs*, which refer, directly or indirectly, to goals, motives, or intentions of someone or some *thing*. They may also express reasons for why someone exists, something has been done, someone is used, etc. in a *context*.

Purpose

A **purpose** is a *generic concept* standing for *things* in the *purpose domain*.

Goal

A **goal** is a *purpose* referring to a desired state of affairs.

Reason

A **reason** is a *purpose* that is used as a basis or cause for some action, fact, event etc.

dueTo relationship

A **dueTo relationship** between a *goal* and a *reason* is a *contextual relationship* meaning that a *reason* gives an explanation, a justification or a basis for setting a *goal*.

Strategic goal

A **strategic goal** is a *goal* with the lifespan of 5 – 10 years.

Tactic goal

A **tactic goal** is a *goal* that shows how to attain *strategic goals*.

Operative goal

An **operative goal** is a *goal* that is generally determined as concrete requirements that are to be fulfilled by specified time point.

Hard goal

A **hard goal** is a *goal* that has pre-specified criteria.

Soft goal

A **soft goal** is a *goal* that has not pre-specified criteria.

Criterion

A **criterion** is a standard of judgment presented as an established rule or principle for evaluating some *thing*.

Requirement

A **requirement** means some *thing* that is necessary and needed, a statement about the future.

Functional requirement

A **functional requirement** is a *requirement* that can be achieved by performing a sequence of operations.

Non-functional requirement

A **non-functional requirement** is a *requirement* that is defined in terms of constraints, to qualify the *functional requirement* related to it.

Problem

A **problem** is the distance or a mismatch between the prevailing *state* and the *state* reflected by the *goal*.

Structured problem

A **structured problem** is a *problem* that is routine, and can be solved using standard solution techniques.

Semi-structured

A **semi-structured problem** is a *problem* for which there are, only to some extent, standard solution techniques available.

Unstructured problem

An **unstructured problem** (or a wicked problem) is a *problem* that does not usually fit a standard mold, and is generally solved by examining different scenarios, and asking "what if" type questions.

Strength

Strength means something in which one is good, something that is regarded as an advantage and thus increasing the possibilities to gain something better.

Weakness

Weakness means something in which one is poor, something that could or should be improved or avoided.

Opportunity

An **opportunity** is a situation or condition favourable for the attainment of a *goal*.

Threat

A **threat** is a situation or condition that is a risk for attainment of a *goal*.

refinement relationship

A **refinement relationship** between the *goals* is a *contextual relationship* that establishes a *goal* hierarchy, meaning that a *goal* can be reached when the *goals* below it (so-called sub-goals) in the hierarchy are fulfilled.

influence relationship

An **influence relationship** is a *contextual relationship* that indicates that the achievement of a *goal* has some influence on the achievement of another *goal*.

causalTo relationship

A **causalTo relationship** between two *problems* is a *contextual relationship* that means that the appearance of one *problem* is at least a partial *reason* for the occurrence of the other *problem*.

Actor domain

The **actor domain** consists of those *concepts*, which refer to human and active parts in a *context* (i.e. individuals, groups, positions, or organizations).

Actor

An **actor** is a human or administrative actor, a *generic concept* used to refer to *things* in the *actor domain*. Actors have an active *role* in a *context*.

Human actor

A **human actor** is an individual person or a group of persons.

Person

A **person** is a human being, characterized by his/her consciousness, emotions, personality, beliefs, desires, intentions, social *relationships*, and behavior patterns conditioned by his/her culture.

Group

A **group** is a *set* of two or more *persons*.

Position

A **position** is a post of employment occupied by a *human actor*.

occupiedBy relationship

An **occupiedBy relationship** is a *contextual relationship* showing a *position* assigned to a *human actor.*

Organizational role

An **organizational role** is a collection of responsibilities, stipulated in an operational or structural manner.

supervision relationship

A **supervision relationship** is a *contextual relationship* involving two *positions* such as one is a supervisor to another that is called a subordinate.

Organization

An **organization** is an administrative arrangement or structure established for some *purposes*, manifesting the division of labor into *actions* and the coordination of *actions* to accomplish the work.

Organizational unit

An **organization unit** is composed of *positions* with the established *supervision relationships*.

Action domain

The **action domain** consists of those *concepts* and *constructs*, which refer to functions, activities, tasks, or operations carried out in a *context*, that is to say, to *state transitions* in *reality.*

Action

An **action** is the *generic concept* that refers to *things* (i.e. deeds or events) belonging to the *action domain*.

Management – execution structure

A **management – execution structure** is a *whole* composed of one or more management actions and those execution actions that implements prescriptions provided by the management actions.

Management action

A **management action** aims at providing the execution actions with prescriptions and resources.

Execution action

An **execution action** aims to implement prescriptions given by *management actions* with the given resources.

Planning

Planning consists of all those *management actions* that lead to the creation, assessment, and selection of alternative future courses of *action* and the program for carrying out the *actions*.

Organizing

Organizing contains all those *management actions* that result in the design of a formal *organization* structure of *actions* and authority relationships.

Staffing

Staffing consists of all those *management actions* required to fulfill and sustain filled *positions* that were established by *organizing*.

Directing

Directing consists of all those *management actions* dealing with the interpersonal aspects through which the personnel come to understand and contribute to the achievement of organizational *goals*.

Controlling

Controlling consists of all those *management actions* that ensure that the actual work goes according to the plans.

Problem solving structure

A **problem solving structure** is a *whole* that is composed of three kinds of *actions*: intelligence, design options, and choice.

Intelligence

Intelligence means *actions* that search the environment for conditions calling for a decision and collect information based on which a decision can be made.

Design

Design consists of the *actions* of inventing, shaping and specifying alternatives for possible courses of *action*.

Choice

Choice means the evaluation and comparison of alternatives and the selection among them.

Rule

A **rule** is a principle or regulation governing a conduct, *action*, procedure, arrangement, etc. It is composed of four *parts*: event, condition, thenAction, and ElseAction.

Event

An **event** is an instantaneous happening in the *context* or in its environment that is significant for the *behavior* of the *context*. An event has no duration.

Condition

A **condition** is a prerequisite for triggering an *action*.

thenAction

A **thenAction** is an *action* that is carried out when the *event* occurs and if the *condition* is true.

elseAction
An **elseAction** is an *action* that is carried out when the *event* occurs but the *condition* is not true.

Dynamic rule
A **dynamic rule** is a *rule* that restricts the allowable *transitions* between the pre-*states* and the post-*states*.

Static rule
A **static rule** is a *rule* that restricts the allowable *states*.

Analytic rule
An **analytic rule** is a *rule* that cannot be broken by an inter-subjectively agreed definition of the terms used in the *rule*.

Empirical rule
An **empirical rule** is a *rule* that cannot be broken according to shared explicit knowledge.

Deontic rule
A **deontic rule** is a *rule* that is socially agreed among the *persons*.

Internal event
An **internal event** is an occurrence happening inside the *context*.

External event
An **external event** is an occurrence happening in the environment of the *context*.

Temporal event
A **temporal event** is an occurrence having time as its impulse.

Action decomposition structure
An **action decomposition structure** is a *whole* composed of *actions*, sub-actions, sub-sub-actions, etc.

Action control structure
An **action control structure** is a *whole* in which the *actions* are logically related to each other according to an execution order.

sequence relationship
A **sequence relationship** between two *actions* is a *contextual relationship* meaning that after selecting one *action* a certain *action* is next to be selected.

selection relationship
A **selection relationship** is a *contextual relationship* meaning that after selecting one *action* there is a set of alternative *actions* from which one *action* (or a certain number of *actions*) is to be selected.

iteration relationship
An **iteration relationship** is a *contextual relationship* meaning that after selecting one *action* the same *action* is selected once more.

Temporal action structure
A **temporal action structure** is a *whole* in which the *actions* are organized on the basis of temporal *conditions* and *events*.

Temporal event
A **temporal event** is a time-driven *event*.

Work procedure

A **work procedure** is a *whole* that is composed of related *rules*.

Process

A **process** is an *instance* of an *action*.

Object domain

The **object domain** consists of those *concepts* and *constructs*, which refer to some objects, which an *action* is targeted to in a *context*. The objects can be material or informational.

Object

An **object** is a *generic concept* used to refer to *things* in the *object domain*.

Material object

A **material** object is an *object* that does not carry or present any information.

Informational object

An **informational object** is an *object* that carries and/or presents some information.

Linguistic object

A **linguistic object is** an *object* that is presented in a *language*.

Conceptual object

A **conceptual object** is composed of UoD constructs which are *signified* by *linguistic object(s).*

Service

A **service** is some *object*, tangible or intangible, composed of *material* and *informational objects,* and made for or given to someone from which it/she/he benefits.

Desciption

A **description** is a descriptive *object*, i.e. representation of information about a slice of the *UoD* (the actual or possible world).

Prescription

A **prescription** is a prescriptive *object*, i.e. a representation of the established practice or authoritative regulation for *action*.

Assertion

An **assertion** is a *description*, which asserts that a certain *state* has existed or exists, or a certain *event* has occurred or occurs.

Prediction

A **prediction** is a *description* of a future possible world with the *assertion* that the course of *events* in the actual world will eventually lead to this *state*.

Rule

A **rule** is a *prescription* which consists of at least two *parts* ((Event or Condition) and Action).

Command

A **command** is a prescription which has neither event nor condition part.

Plan

A **plan** is an *informational object* that may possess aspects of several intentional *subtypes* (e.g. *description*, *prediction*, and *prescription*).

Version

A **version** is a result of an iterative or phased *action* toward the final outcome.

versionOf relationship

A **versionOf relationship** is a *contextual relationship* holding between two *objects*, if the properties of, and the experiences from, one *object* have influenced upon the creation of another *object,* provided that they are for the same purposes and the objects refer to the same UoD.

copyOf relationship

A **copyOf relationship** is a *contextual relationship* that holds between two *objects*, if the original *object* and a copy *object* are exactly, or to an acceptable extent, similar.

supports relationship

A **supports relationship** is a *contextual relationship* that involves two *informational objects*, such that the information "carried" by one *object* is needed to produce the other *object*.

predAbstract relationship

A **predAbstract relationship** between two *informational objects* is a *contextual relationship* meaning that one *object* is more abstract that the other object in terms of *predicate abstraction* and both of the *objects signify* the same *thing*(s) in the *UoD*.

signifies relationship

A **signifies relationship** between a *linguistic object* and a *conceptual object* is a *contextual relationship* that defines the conceptual meaning of the *linguistic object* in terms of UoD constructs, which the *linguistic object signifies*.

UoD construct

A **UoD construct** means any conceptual *construct* in the same or different *context*.

Facility domain

The **facility domain** consists of those *concepts* and *constructs*, which refer to means by which something can be done or is done in a *context*.

Facility

A **facility** is the *generic concept* in the *facility domain*, which means either a tool or a resource.

Tool

A **tool** is a *thing* that is designed, built, installed, etc. to serve a specific *action* affording a convenience, efficiency or effectiveness.

Resource

A **resource** is a kind of the source of supply, support, or aid. It can be money, energy, capital, goods, manpower, etc.

compatability relationship

A **compatability relationship** between two *tools* (or components) is a *contextual relationship* meaning that the interfaces of the *tools* are structurally and functionally interoperable.

Configuration

A **configuration** is a *whole* that is composed of the components with compatible *versions*.

Location domain

The **location domain** consists of those *concepts* and *constructs*, which refer to *parts* of space occupied by someone or some*thing* in a *context*.

Location

A **location** is a *generic concept*, which refers to a physical or logical *location*.

Physical location

A **physical location** is a spatial *thing* (e.g. a room or a building), which is placed in a region of space and which can, through its spatial attachment, provide a place for some other *thing*.

Spatial thing

A **spatial thing** is some *thing* that is necessary or beneficial to localize.

Region

A **region** is a *part* or division of space.

Point

A **point** is the elementary unit in space specified by a single coordinate with reference to a system of two or three geographical dimensions.

Area

An **area** is any particular extent of space specified with at least two coordinates.

Geographical dimension

A **geographical dimension** means any dimension within which space can be specified.

Geographical system

A **geographical system** is a system of two or three *geographical dimensions*.

placedIn relationship

A **placedIn relationship** between a *region* and a *spatial thing* is a *contextual relationship* meaning that the *spatial thing* is located in the *region*.

Logical location

A **logical location**, like a site within a computer network, is a space that is not attached to any geographical *point* or *area*.

topological relationship

A **topological relationship** between two regions states how the regions are related in terms of geographical *points* or *areas* along the *geometric dimensions.*

connectedTo relationship

A **connectedTo relationship** is a *contextual relationship* meaning that two sites can communicate with each other by sending messages.

Time domain

The **time domain** consists of those *concepts* and *construct*, which refer to temporal aspects in a *context*.

Time

Time is indefinite, unlimited duration in which something is considered as happening in the past, present, or future. Most of our knowledge is founded in time and expressed in terms of time units.

Time unit

A **time unit** means a unit of measuring *time*.

Time point

A **time point** is the primitive *time unit* as an indivisible point on the *time* continuum.

Time interval

A **time interval** is an *abstraction* of *time points*, manifesting duration of something.

Convex time interval

A **convex time interval** is a *time interval* that consists of continuous *time points*.

Non-convex time interval

A **non-convex time interval** is an *interval* with some "holes".

Time system

A **time system** is a totally ordered *set* of *time units*.

relatedTo relationship

A **relatedTo relationship** means a *contextual relationship* between two *time systems*.

temporal relationship

A **temporal relationship** means a *contextual relationship* between two *time units*, which are *time points* and/or *time intervals*.

expressedBy relationship

An **expressedBy relationship** between an *actor* and a *purpose* is an *inter-domain contextual relationship* meaning that an *actor* has expressed a *goal*, a *requirement*, a *problem*, or the like concerning the *context* as a whole or some of its *part*, in the same or different *context*.

motivatedBy relationship

A **motivatedBy relationship** between a *human actor* and a *purpose* is an *inter-domain contextual relationship* meaning a subjective or inter-subjective motive or an inner drive that makes a *person* or a *group* to do something or behave as he/she/it does.

strivesFor relationship

A **strivesFor relationship** between an *action* and a *purpose* is an *inter-domain contextual relationship* expressing a *goal*, which an *action* pursues.

intendedFor relationship

An **intendedFor relationship** between an *object*, a *facility*, or a *location*, on one hand, and a *purpose*, on the other hand, is an *inter-domain contextual* *relationship* meaning a *goal* or a *reason* for which an *object*/a *facility*/an *object* is made, acquired, and/or used.

carryOut relationship
A **carryOut relationship** is an *inter-domain contextual relationship* meaning that an *actor* conducts an *action*.

responsibleFor relationhip
A **responsibleFor relationship** is an *inter-domain contextual relationship* specifying those *actions,* which an *organizational role* is responsible for.

occursAt relationship
An **occursAt relationship** is an *inter-domain contextual relationship* defining when an *action* is done, has been done or will be done.

ownedBy relationship
An **ownedBy relationship** is an *inter-domain contextual relationship* meaning that an *actor* is an "owner" of an *object*.

viewedBy relationship
A **viewedBy relationship** is an *inter-domain contextual relationship* meaning that an *object* is a view, insight, opinion, etc. of an *actor*.

useAbility relationship
A **useAbility relationship** is an *inter-domain contextual relationship* meaning that it is possible for an *actor* to use a *facility*.

input relationship
An **input relationship** is an *inter-domain contextual relationship* meaning that an *object* is used as an input to an *action.*

output relationship
An **output relationship** is an *inter-domain contextual relationship* meaning that an *action* produces an *object* as its output.

involvedBy relationship
An **involvedBy relationship** is an *inter-domain contextual relationship* meaning that a *UoD construct* is involved by an *action* through *informational objects* that *signify* a *UoD construct*. Involving may mean creating, modifying, utilizing, or deleting *informational objects*.

performs relationship
A **performs relationship** is an *inter-domain contextual relationship* meaning that an *action* is performed by a *tool.*

uses relationship
A **uses relationship** is an *inter-domain contextual relationship* meaning that an *action* consumes certain *resources.*

usedToMake
A **usedToMake relationship** is an *inter-domain contextual relationship* meaning that a certain *facility*, i.e. a *tool* or a *resource*, is used to produce an *object.*

situatedIn relationship
A **situatedIn relationship** is an *inter-domain contextual relationship* meaning that a *human actor*/an *object*/a *facility* is situated in a *location*.

existsAt relationship
An **existsAt relationship** is an *inter-domain contextual relationship* defining when a *purpose*/an *actor*/an *object*/a *facility*/a *location* exists, has existed, or will exist.

III. Layer ontology

The **layer ontology** provides *concepts* and *constructs* to conceive, understand, structure and represent the static and dynamic features of information processing at four layers. It is composed of two *parts*. The first *part* provides the *concepts* and *constructs* related to information processing in general. The second *part* of the ontology shows how information processing is structured and related onto four layers according to the predefined system of layers.

Knowledge
Knowledge is a relative stable and sufficiently consistent *set* of (*conceptual*) *informational objects* owned by single *human actors*.

Explicit knowledge
Explicit knowledge is a body of *knowledge* that can be articulated in a natural or *formal language*.

Tacit knowledge
Tacit knowledge is a body of *knowledge* that is embedded in personal experience and therefore cannot be (easily) represented externally.

Data
Data is *knowledge* represented in a *language*.

Information
Information is the *knowledge* increment brought about by receiving *data*, by observing *reality*, or by inner thought *processes* by which a *person* organizes, compares, assesses his/her *knowledge*.

Information processing
Information processing means *action*(s) by which *informational objects* are collected, stored, processed, disseminated and interpreted.

System
A **system** is a *conceptual construct* through which phenomena in *reality* can be conceived as a *whole* (system), contained in the environment, characterized by emergent *predicates*, and composed of *parts* (elements).

Computerized information system
A **computerized information system** (CIS) is a *system* in which all *information processing* is automated, that is to say, performed by one or more computer *systems*.

Human information system
A **human information system** (HIS) is a *system*, in which *human actors* play the only role in the accomplishment of *actions* to process *information* in a structured way.

Information system

An **information system** is a *system*, composed of *actors*, *information/data*, *facilities* and *locations*, collecting, storing, processing and distributing *information* about the relevant *parts* of *reality*, called the object system, in order to enable and/or improve *actions* in the other *context*, called the utilizing system.

Utilizing system

A **utilizing system** (US) is a *system*, which exploits information services, provided by the *information system*, in its decision making or operational actions, in order to make *plans* and execute changes (i.e. *state transitions*) in the controlled system.

Information service

Information service is a *service* that is composed *informational objects*.

Controlled system

A **controlled system** is a *system*, which the *utilizing system* has control over.

User

A **user** of IS is an *actor* who potentially increases his/her *knowledge* about some phenomena in the object system with the help of the *IS*.

End-user

A **end-user** is an *actor*, who increases his/her *knowledge* by interacting directly with the *CIS*.

Indirect-user

An i**ndirect user** is an *actor*, who increases his/her *knowledge* by getting results from the *CIS* through other *users* of the *information system*.

Object system

An **object system** (OS) means a *system* about which the *IS*, due to the interests of the *US*, collects, stores, processes and disseminates *information* (*services*) to the *US*.

Primary action

A **primary action** means routine-like *information processing*.

Development action

A **development action** means an *action* to make changes in routines of the *primary action*.

Micro-level development

A **micro-level development** means a *development action* carried by individual *persons* as part of their personal work.

Mid-level development

A **mid-level development** means a *development action* carried out in group works, organized often informally when necessary.

Macro-level development

A **macro-level development** means a pre-planned, controlled, and coordinated *development action* that involves several individual *persons* with various skills, takes weeks or months, sometimes years, and may costs a lot of money.

Processing layer

A **processing layer** is composed of *information processing actions*, which share the similar *goals* and the same target (i.e. *object*) of *action*.

System of layers

A **system of layers** is a *system* that is composed of *processing layers*, which constitute a hierarchical structure, in which *actions* at a higher *layer* produce *informational objects* to be used as *prescriptions* for the *actions* at the next lower *layer*.

IV. Perspective ontology

The **perspective ontology** provides the *concepts* and *constructs* for conceiving, understanding, structuring and representing *things* in *information processing contexts* with a system of pre-defined perspectives[180].

Perspective

A **perspective** is a strictly defined *point of view*.

System of perspectives

A **system of perspectives** is a (static) *system*, which is composed of *perspectives* and *relationships* between them.

Systelogical perspective

The **systelogical perspective** is a *perspective* according to which the *IS* is seen in relation to its *utilizing system* (US).

Infological perspective

The **infological perspective** is a *perspective* according to which the *IS* is seen as a functional structure of *information processing actions* and *informational objects*, independent from any representational and implementational features.

Conceptual perspective

The **conceptual perspective** is a *perspective* according to which the *IS* is considered through the semantic contents of *information* it processes.

Datalogical perspective

The **datalogical perspective** is a *perspective* according to which the *IS* is viewed through *representation*-specific *concepts* as a *context*, in which *IS actors* work with *IS facilities* to process *data*.

Physical perspective

The **physical perspective** is a *perspective*, which ties the *datalogical concepts* and *constructs* to the particular organizational and technical environment, showing how the *IS* looks like and behaves when it is implemented.

180 The definitions of the perspectives below are expressed in relation to the IS. The notion of the IS should be here understood generally; It can mean information processing on any processing layer.

V IS ontology[181]

The **IS ontology** provides *concepts* and *constructs* to conceive, understand, structure and represent the contextual features of *information processing* at the *IS layer.*

V.1 IS domains

IS purpose
An **IS purpose** means an *IS goal* for the *IS* and/or s *reason* for setting up a *goal.*

IS goal
An **IS goal** is a desired *state* of affairs in the *IS*.

IS reason
An **IS reason** can be a *functional* or *non-functional requirement* for *information processing*, a *problem* in prevailing *information processing*, *strength* and *weakness* in, and an *opportunity* and a *threat* for, existing or planned *information processing*.

IS actor
An **IS actor** is an *actor* working in and for the *IS*.

Human IS actor
A **human IS actor** is an individual *person* or a *group* of *persons* working in and for the *IS*.

IS role
An **IS role** is a collection of responsibilities, stipulated in terms of *HIS actions*.

IS position
An **IS position** is a *position*, composed of the defined *IS roles* and occupied by a *human IS actor*.

IS organization
An **IS organization** is an *organization* whose main responsibility is to develop, manage and/or carry out *information processing* in a business organization.

IS organizational unit
An **IS organizational unit** is composed of *IS positions*.

IS action
An **IS action** is an *action* that strives for one or more *IS purposes*.

IS object
An **IS object** is an *informational object signifying* one or more *OS_{IS} constructs*. It is an input to and an output from one or more *IS actions*.

181 The concepts of this ontology are categorized in the following way. The concepts that are directly related to the IS are presented in the first part, the IS domains. The definitions of the IS perspectives as well as the concepts of the utilizing system (US) and the object system (OS) are included in the second part, the IS perspectives.

Transient IS object

A **transient IS object** is an *IS object*, which lasts only a short time (e.g. a reply to a routine request).

Permanent IS object

A **permanent IS object** is an *IS object* that is valuable enough to "live" longer (e.g. personnel information, vehicle information).

Data object

A **data object** is an *IS object* represented in some *language*. It can be in a digital or non-digital form.

Non-digital data

Non-digital data means an *IS object* that is presented in a *language* that can be interpreted by a human being.

Digital data

Digital data is an *IS object* that is in a digital form and can be read by a computer.

IS rule

An **IS rule** governs one or more *IS actions*. It is composed of four *parts*: IS event(s), IS condition(s), thenISAction(s), and elseISAction(s).

Human information system

A **human information system** (HIS) means a *system* in which human beings have the only *role* in the accomplishment of the *IS actions*.

HIS purpose

A **HIS purpose** is a *IS purpose* which concerns the *HIS* as a *whole*, or *parts* thereof.

HIS action

A **HIS action** is an *IS action* carried out by a *human IS actor*, to attain one or more *HIS purpose*.

HIS rules

A **HIS rule** is an *IS rule* governing one or more *HIS action*.

User interface

A **user interface** is a *part* of the *CIS* which facilitates the interaction between the *users* and the *CIS*.

Dialog

A **dialog** means an interaction between a *user* and the *CIS*, occurring through windows.

Window

A **window** is a logical *whole* through which a *user* can communicate with the *CIS*. It is composed of UI components.

UI component

A **UI component** is a *part* of a *window*. It can be a UI data component or a UI action component.

UI data component

A **UI data component** is a *UI component* that displays *data* to a *user* or accepts *data* from a *user*.

UI data

UI data is an *IS object* displayed by the *CIS* to a *user*, or got from a *user*.

UI action component

An **UI action component** is a *UI component* intended to the manipulation of the *window* and the control of the *dialog* (e.g. a button, a menu, a slider, etc).

UI state

A **UI state** is a *state* composed of those *UI data*, *UI data components* and *UI action components* that are present at the certain *time*.

UI transition

A **UI transition** is a *transition* from one *UI state* to another, triggered by an *HIS action* or by a *CIS action*.

UI event

A **UI event** means any happening that triggers *UI transitions*.

Computerized information system

A **computerized information system** (CIS) a *system* in which all *data processing* is automated, that is to say, performed by one or more computer systems.

CIS action

A **CIS action** means as *IS action* that is performed by the *CIS*.

Transaction

A **transaction** is composed of logically related *CIS actions*.

CIS rule

A **CIS rule** is an *IS rule* governing one or more *CIS actions*.

Algorithm

An **algorithm** is a *transaction* represented in a *formal language*.

Hardware architecture

A **hardware architecture** consists of interoperable hardware.

Software architecture

A **software architecture** is composed of compatible software.

Application software

An **application software** is composed of SW components.

Layer

A **layer** is a part of the layered *software architecture* in which layers are related to one another with the black box strategy or the while box strategy.

SW component

A **SW component** means an executable unit of code that provides physical black-box encapsulation of related services. Its services can only be accessed through a consistent, published interface.

Node

A **node** is composed of e.g. memory devices, processors, printers and displays.

Communication line

A **communication line** is a line along which *data* messages are sent from one *node* to another.

Protocol

A **protocol** means a set of conventions or *rules* that govern the interactions of *processes* or *software components* through *communication lines* in a *CIS* or between *CIS's*.

Data storage

A **data storage** stands for all kinds of structured *digital data* (e.g. a data file and a database).

Database

A **database** is a *data storage* structured according to some database model (e.g. a hierarchical model, a relational model, an object-relational model, an object model, a document model, XML-native model).

Data file

A **data file** is a *data storage* that is decomposed into records and data fields.

Memory device

A **memory device** is a device for storing *permanent data*.

V.2 IS Perspectives

IS perspective

An **IS perspective** is a *perspective* with which features of the *IS*, relevant to the *problem* or the situation at hand, can be considered.

IS systelogical perspective

The **IS systelogical perspective** is an *IS perspective* from which the *IS* is seen in relation to its *utilizing system* (US_{IS}).

US purpose

A **US purpose** means a *goal* for business processes and/or a *reason* for setting up *goal*(s).

US organization

A **US organization** is an *organization* (i.e. an enterprise, a department or some other administrative arrangement), which utilizes, or is going to utilize, an *IS*. It is composed of US organizational units.

US actor

A **US actor** is an *actor* working in and for the *US context*.

US role

A **US role** is a collection of responsibilities, stipulated in terms of US actions.

US position

A **US position** is a *position* composed of the defined *US roles* and occupied by a *US human actor*.

US organizational unit

A **US organizational unit** *is composed of US positions.*

US action

A **US action** is an *action*, which strives for one or more *US purposes*.

US rule

A **US rule** governs one or more *US actions*.

US object

A **US object** is a *material* or *informational object* that is an input to and/or an output from one or more *US action*.

US tool

A **US tool** is a *tool* designed, built, installed, etc. to serve or perform *US actions*.

US resource

A **US resource** is a *resource*, like money, energy, goods, manpower, etc. that is used in the *US context*.

IS infological perspective

The **IS infological perspective** is an *IS perspective* from which the *IS* is seen as a functional structure of *information processing* and *informational objects*.

IS conceptual perspective

The **IS conceptual perspective** is an *IS perspective*, which reveals the semantic contents of the *IS objects*.

Entity

An **entity** means any perceivable *thing* in the *object system* with an independent existence.

OS relationship

An **OS relationship** means some relevant connection, association or like (i.e. a *relationship*) between two or more *entities.*

Attribute value

An **attribute value** identifies and characterizes a particular *entity* or *OS relationship*.

OS_{IS} construct

An **OS_{IS} construct** in the OS_{IS} means a *conceptual construct* composed of specific *entities* related to one another through *OS relationships* and characterized by specific *attribute values*.

OS_{IS} state

An **OS_{IS} state** means a *state* of the *object system* (OS_{IS}), or its *parts*, composed of *OS_{IS} constructs*.

OS_{IS} transition

An **OS_{IS} transition** means a *transition* from one *OS_{IS} state*, called the pre-state, to another *OS_{IS} state*, called the post-state.

OS_{IS} event

An **OS event** means an *event*, which may trigger an *OS_{IS} transition* from the pre-state to the post-state, and which may be caused by another *OS_{IS} transition.*

IS datalogical perspective

The **IS datalogical perspective** is an *IS perspective* from which the *IS* is viewed through representation-specific *concepts* as a *context*, in which *IS actors* work with *IS facilities* to process *data.*

IS physical perspective

The **IS physical perspective** is an *IS perspective,* which considers the *IS* with all its physical aspects.

VI. Model Level Ontology

The **model level ontology** provides *concepts* and *constructs* for conceiving, understanding, structuring, and presenting *things* in models within a system of model levels.

Model
A **model** as a *thing* that is used to help or to enable the understanding, communication, analysis, design, and/or implementation of some other *thing* to which the model refers.

Concept model
A **concept model** is a *model* that is composed of *concepts* and conceptual *constructs* referring to certain *things* in *reality*.

Model denotation
A **model denotation** is a precise and unambiguous representation of a *concept model* in some *language*.

Physical model
A **physical model** is a *model* that consists of physical parts, which, as an organized *whole*, resemble some other *thing*(s) (e.g. small copies of airplanes or ships).

Model constructing
Model constructing is an *action* by which a *physical model* is produced from concrete *things* by e.g. moulding, building or crafting.

Modeling
Modeling *is an* action *with which a model is produced.*

Model conceptualizing
Model conceptualizing means a *modeling action* with which a *concept model* is produced by perceiving and conceptualizing the relevant features of the concrete *thing*(s).

Model transforming
Model transforming means a *modeling action* with which a *concept model* is produced by transforming it from some other *concept model(s).*

Model representing
Model representing means a *modeling action* with which a *model denotation* is produced by representing *concept model(s)* by *signs* of some *language.*

Model translating
Model translating means a *modeling action* with which a *model denotation* is produced by translating it from some other *model denotation*.

Model implementing
Model implementing means an *action* with which a *model denotation* is implemented into a *physical model*.

Modeling context
A **modeling context** means a *context* the *purpose* of which is to produce a *model* for a model utilizing *context*.

Informal model

An **informal model** is a *model* that is restricted in its structure by the modeler's imagination.

Semi-formal model

A **semi-formal model** is a *model* that is constrained by the *syntax* of the *language(s)* (e.g. diagrams, tables, matrices and structured texts).

Formal model

A **formal model** is a *model* that is represented in a *formal language* according to the rigorously defined *syntax* and *semantics*.

Subjective model

A **subjective model** is a *model* that reflects the modeler's subjective conception about the subject matter.

Inter-subjective model

An **inter-subjective model** is a *model* that reflects sharing conceptions within a community.

Objective model

An **objective model** is a *model* that reflects "objective truth" (e.g. a *formal model* of the Euclidean space).

Modeled context

A **modeled context** is a *context* about which the *model* is.

Structural model

A **structural model** is a *model*, which is composed of *concepts* that refer to static phenomena in the *modeled context*.

Dynamic model

A **dynamic model** is a *model*, which is composed of *concepts* that refer to the *behavior* in or the *evolution* of the *modeled context.*

Instance model

An **instance model** is a *model*, which is mainly composed of the *concepts* that are *instances* of the *concepts* of another *model*, called the type model.

Type model

A **type model** is a *model*, which is composed of the *concepts* that are *types* of the *concepts* of another *model*, called the *instance model*.

Model utilizing context

A **model utilizing context** is a *context* for which the *model* is produced.

Descriptive model

A **descriptive model** is a *model* that is used to portray or predict the relevant features of the *modeled context*, in order to support the analysis of the existing *reality* or the design of the future *reality*.

Prescriptive model

A **prescriptive model is** a *model* that is conceived as normative statements, which specify what is permitted, forbidden or obliged in certain situations.

Technique

A **technique** is a *prescriptive model* that guides the *behavior* in the *modeled context.* A technique may be composed of procedures and guidelines.

Description technique

A **description** technique is a *technique* to create a *model* and represent it as a *model denotation* (e.g. diagramming techniques).

Processing technique

A **processing technique** is a *technique* to create, transform, translate, analyze, validate and/or verify one or more *models*.

Procedure

A **procedure** is an explicitly specified manner of proceeding in a *process*.

Guideline

A **guideline** is any advice or guide to reach a *goal*.

Meta concept

A **meta concept** is a *concept* an *instance* of which is a *type concept* for some *instance concepts.*

Meta level

A **meta level** is composed of *meta concepts.*

Concept level

A **concept level** is composed of *concepts* among which there are no *instanceOf relationships*.

System of concept levels

A **system of concept levels** is composed of *concept levels* in such a way that the *concepts* on a certain concept level have the *instanceOf relationships* with the *concepts* on the higher *concept level*.

Model level

A **model level** is composed of *models* that comprise *concepts* on the same *concept level.*

System of model levels

A **system of model level** is composed of *model levels* in such a way that the models on a certain *model level* have the *instanceOf relationships* with the *models* on the next higher *model level*.

Meta model

A **meta model** is a *model* that is composed of *meta concepts* .

Metamodeling

Metamodeling is an *action* by which a *meta model* is produced.

Deliverable model

A **deliverable model** is a *model* that describes/prescribes the structure and *presentation* of *informational objects* (e.g. a relational scheme with dta types).

Data model

A **data model** is a *model* that describes/prescribes the *conceptual* contents of *informational objects* (e.g. an ER schema).

VII. ISD Ontology

The **ISD ontology** provides *concepts* and *constructs* for conceiving, understanding, structuring, and representing contextual phenomena of *ISD*.

Paradigm

A **paradigm** means the most fundamental set of assumptions adopted by a professional community, which allow it to share similar perceptions and engage in commonly shared practices.

ISD approach

An **ISD approach** means a generic way of conceiving certain aspects of ISD, or a generic way of working in ISD.

ISD approach in Category A

An **ISD approach in Category A** is a kind of schools of though with identifiable founders and scientific community as institutionalizations. It is a set of *goals*, guiding principles, fundamental *concepts*, and principles for the ISD process that drive interpretations and *actions* in the *ISD*.

ISD approach in Category B

An **ISD approach in Category B** has a specific *view* of *ISD* as a *context*.

ISD approach in Category C

An **ISD approach in Group C** has a particular view of some specific contextual domain(s) of ISD.

Transformation approach

The **transformation approach** is an *ISD approach in Category B* according to which ISD is seen as sequential steps of transforming ISD deliverables on one level of *abstraction* into the ISD deliverables on the next lower level of *abstraction.*

Decision making approach

The **decision making approach** is an *ISD approach in Category B* according to which *ISD* is seen as a decision making *process* in which *knowledge* is acquired, options are specified, and the "best" options are selected.

Problem solving approach

The **problem solving approach** is an *ISD approach in Category B* according to which *ISD* is seen as a *problem* solving *process* in which *problems* at several levels of details are identified and solved.

Learning approach

The **learning approach** is an *ISD approach in Category B* according to which *ISD* is seen as a learning *process* by which *knowledge* on application domain, technology and ISD work is acquired, elaborated and disseminated.

Political approach

The **political approach** is an *ISD approach in Category B* according to which *ISD* is seen as a cooperative *process* composed of negotiations, bargaining, power and social interactions.

Knowledge work approach

The **knowledge work approach** is an *ISD approach in Category B* according to which *ISD* is viewed as *knowledge* work.

IS data-oriented approach

The **IS data-oriented** approach is an *ISD approach in Category C* that regards *data* as the fundamental *parts* of an *IS*.

IS process-oriented approach

The **IS process-oriented** approach is an *ISD approach in Category C* that views *information processing actions* or *processes* as the most essential *parts* of the *IS*.

IS user-oriented approach

The **IS user-oriented** approach is an *ISD approach in Category C* that puts the major emphasis on human beings, their needs, *views* and interactions in the *IS* and in the *US*.

Life cycle approach

The **life cycle approach** is an *ISD approach in Category C* that decomposes the *ISD work* into discrete *phases* to be accomplished in an order that is comparable to sequential waterfalls. Each *phase* should be satisfactorily completed before the next one begins

Prototyping approach

The **prototyping approach** is an *ISD approach in Category C* that aims, through prototypes, to increase the understanding of those issues on which there exists some uncertainty, and thus to decrease risks related to the *ISD process* or its outcome.

Incremental approach

The **incremental approach** is an *ISD approach in Category C* that means the *process* of constructing a partial *implementation* of a total *system* and slowly adding increased functionality or performance.

Evolutionary approach

The **evolutionary approach** is an ISD approach in *Category C* according to which an *information system* is an incremental outgrowth of evolution and learning and it continues to evolve over time owing to new learning experiences.

Information system development

Information development system is a *context* in which ISD actors carry out ISD actions, ranging from requirements engineering to implementation and evaluation of an *IS*, to produce ISD deliverables contributing to a renewed or a new *IS*, by means of ISD facilities in a certain organizational and spatio-temporal *context*, in order to satisfy ISD goals set by ISD stakeholders.

VII.1 ISD Domains

ISD purpose domain

The **ISD purpose domain** embraces all those *concepts* and *constructs* that refer to *goals*, motives, or intentions of someone or something in the *ISD context*. The *concepts* may show a direction toward which it is due to proceed, a *state* to be attained or avoided, and *reasons* for them.

ISD goal

An **ISD goal** expresses is a desired *state* or *event* with qualities and quantities, related to an *ISD context* as a *whole* or to some of its *parts*.

Hard ISD goal

A **hard ISD goal** is an *ISD goal* with pre-specified *criteria* for the assessment of the fulfilment.

Soft ISD goal

A **soft ISD goal** is an *ISD goal* with no pre-specified *criteria* for the assessment of the fulfilment.

ISD requirement

An **ISD requirement** is some quality or performance demanded in and for the *ISD context*. It is a statement about the future.

ISD problem

An **ISD problem** is the distance or mismatch between the prevailing *ISD state* and the *state* reflected by the *ISD goals.*

IS criterion

A **IS criterion** is a standard of judgment presented as an established rule or principle for evaluating some feature(s) of the I*S*.

IS requirement

An **IS requirement** means a condition or capability of the *IS* needed by an IS client or an IS worker to solve a *problem* or achieve a *goal.*

Functional IS requirement

A **functional IS requirement** is an *IS requirement* that specifies what the *IS* should do and for whom.

Non-functional IS requirement

A **non-functional IS requirement** is an *IS requirement* that constraints or sets some quality attributes upon the *services* or functions offered by the *IS.*

IS systelogical requirement

An **IS systelogical requirement** is an *IS requirement* that concerns (e.g. the benefits and costs of) *information services* the *IS* should provide to its *utilizing system*.

IS infological requirement

An **IS infological requirement** is an *IS requirement* that expresses demands on the type and quality of *information* needed as well as *actions* with which the *information* is to be processed.

IS conceptual requirement

An **IS conceptual requirement** is an *IS requirement* that pertains to the contents of *information* to be processed in the *IS*.

IS datalogical requirement

An **IS datalogical requirement** is an *IS requirement* that concerns e.g. how to present *information*, how to divide *information processing* between *persons* and *computers*, and how to organize *responsibilities* for *information processing* into *IS roles* and *IS positions.*

IS physical requirement

An **IS physical requirement** is an *IS requirement* that expresses detailed demands on physical structures and behavior of the *HIS* and the *CIS*.

ISD actor domain

The **ISD actor domain** consists of all those *concepts* and *constructs* that *refer to* human and active part of an *ISD context*.

ISD actor

An **ISD actor** is an ISD human actor or an administrative actor that is, one way or another, involved in an *ISD context*.

ISD human actor

An **ISD human actor** means an individual *person* or a *group* of *persons* contributing to *ISD work*.

ISD role

An **ISD role** is a collection of ISD responsibilities and authorities, stipulated in terms of *ISD actions*.

ISD position

An **ISD position** is a *position*, composed of the defined *ISD roles* and occupied by a *human ISD actor*.

IS owner

An **IS owner** is an *ISD role* in which an *ISD actor* has financial interest in the *IS* and, thereby, the *responsibility for*, and the authority of, making decisions on the *IS* as though it would be his/her property.

IS client

An **IS client** is an *ISD role* for whom the *IS* is to be developed. He/she is a beneficiary or a 'victim' of the *IS*.

IS worker

An **IS worker** is an *ISD role*, in which an *ISD actor* is working with the current *IS* and/or is going to work with the new *IS*.

IS developer

An **IS developer** is an *ISD role*, in which an *ISD actor* is engaged in meeting the needs and *requirements* put forward by *ISD actors* in the other *roles*.

Project manager

A **project manager** is an *ISD role*, in which an *ISD actor* makes *plans* on how to organize an *ISD* effort. He/She also participates in making decisions on the *execution* of the *plans*.

Vendor/consultant

A **vendor / consultant** is an *ISD role*, which is played by a *person* from outside the *organization*.

ISD stakeholder

An **ISD stakeholder** means an *ISD actor* who is potentially affected by the IS or ISD and therefore is invited to act in some of the *ISD roles*.

ISD project

An **ISD project** is a temporary effort with the well-defined *objectives* and constraints, the established *organization*, the budget and the schedule, launched for the accomplishment of *ISD*.

ISD project organization

An **ISD project organization** is a composition of *ISD positions* and ISD teams wherein the responsibility, authority and communication *relationships* are defined.

ISD organizational unit

An **ISD organisational unit** is a composition of *ISD positions* with a coherent set of organizational *goals*, authorities and responsibilities.

Steering committee

A **steering committee** is a *group*, which carries the responsibility of the overall *management* of the *ISD project*.

Project team

A **project team** is a *group* that is collected for the *execution* of an *ISD* effort.

Leader

A **leader** of a *project team* is an *ISD position* devoted to the *management* of the *team* and to ensure the proper communication between the *members*, as well as between the *team* and the other *teams*

IT expert

An **IT expert** is a *person* whose education, skills, experience as well as his/her former *position* is related to the information technology and/or ISD methods.

Business expert

A **business expert** is a *person*, who is knowledgeable in business strategies, policies, markets, competition, trends, legislation, etc., shortly in how to make business, in general or in the *organization*.

Work expert

A **work expert** is a *person*, who masters daily routines, e.g. in making orders, invoicing, production planning, inventory control, goods deliveries, etc.

ISD action domain

The **ISD action domain** comprises all those *concepts* and *constructs* that refer to deeds or *events* in an *ISD context*.

ISD action

An **ISD action** is an *action* carried out to *manage* and/or *execute* a *part* of an *ISD* effort.

ISD rule

An **ISD rule** governs one or more *ISD action.*

ISD process

An **ISD process** is an *instance* of an *ISD action.*

ISD management-execution structure

The **ISD management–execution structure** is a functional and behavioral unity, composed of two kinds of *actions*, ISD management actions and ISD execution actions.

ISD management action

An **ISD management action** aims to plan, organize, staff, direct, and/or control *ISD* work.

ISD planning

ISD planning means all those *ISD management actions* that specify the *goals* of an *ISD project* and the strategies, policies, programs and procedures for achieving them.

ISD organizing

ISD organizing means all those *ISD management actions* that are needed to design a formal structure of ISD execution actions and authority *relationships* between them.

ISD staffing

ISD staffing means all those *ISD management actions* that are needed to fill and keep filled the *ISD positions* of the *ISD project organisation*.

ISD directing

ISD directing means all those *ISD management actions* that are needed for clarifying the assignments of ISD personnel, assigning *actions* to *organisational units*, *teams* and *individuals*, motivating and inspiring personnel, resolving disagreements between personnel and between the *ISD project* and outer *stakeholders*.

ISD controlling

ISD controlling means all those *ISD management actions* that are needed for ensuring that actual *ISD actions* are executed according to the *plans*.

ISD execution action

An **ISD execution action** aims to produce the required ISD deliverables under the guidance and control of *ISD management*.

ISD workflow structure

The **ISD workflow structure** is composed of various ISD workflows.

ISD workflow

An **ISD workflow** is a coherent composition of *ISD actions*, which are organised to accomplish some *ISD process*, which share the same target of *action*, and which produce valuable results for *stakeholders.*

ISD task

An **ISD task** is a *part* of an *ISD workflow*.

IS requirements engineering

IS requirements engineering is an *ISD workflow*, which aims to identify and elicit *IS clients'* and *IS workers' requirements* on an *IS*, as well as to establish and maintain, at least to some extent, agreement on what the *information system* should do and why.

IS analysis

IS analysis is an *ISD workflow*, which *models* the problem domain.

IS design

IS design is an *ISD workflow*, which *models* the solution domain.

IS implementation

IS implementation is an *ISD workflow*, which fleshes out the architecture and the *system* as a *whole*, by carrying IS design *models* into effect.

IS evaluation

IS evaluation is an *ISD workflow*, which aims at the assessment of an existing *system*, as well as the evaluation of all the specifications, designs and implementations made for the future *system.*

ISD phase structure

The **ISD phase structure** is composed of sequential ISD phases.

ISD phase

An **ISD phase** means an *ISD action*, executed between two milestones, by which a well-defined set of *goals* is met, ISD deliverables are completed, and decisions are made on to move or not to move into the next *phase*.

Milestone

A **milestone** is a synchronization point where *ISD management* makes important business decisions and ISD deliverables have to be at a certain level of completion.

IS inception

The **IS inception phase** is an *ISD phase* where the focus is on the understanding of the overall *requirements* and determining the scope of the development endeavor.

IS elaboration

The **IS elaboration phase** is an *ISD phase* where the focus is on detailed *requirements engineering*, but some *systems design* and *implementation actions* aimed at prototyping can also be done.

IS construction

The **IS construction phase** is an *ISD phase*, which focuses on *design* and *implementation* of the *system*.

IS transition

The **IS transition phase** is an *ISD phase* into which it is entered when at least some part of an ISD baseline is mature enough to be deployed.

ISD problem solving structure

The **ISD problem solving structure** is a result of seeing *ISD* as a series of interrelated decisions, which involve the identification and articulation of *problems*, alternative solutions, decisions and justifications.

Intelligence

Intelligence means *ISD actions* that search the environment for conditions calling for a decision.

Design

Design consists of *ISD actions* of inventing, shaping and specifying alternatives for possible courses of *action* in *ISD* work.

Choice

Choice means the evaluation and comparison of each alternative design option and the selection among them.

IS modeling

IS modeling is an *ISD action* which aims to produce an *IS model*.

IS modeling structure

The **IS modeling structure** is composed of *IS modeling actions.*

Elementary modeling structure

The **elementary modeling structure** comprises *IS modeling actions* that are always present in *IS modeling*.

Conceptualizing

Conceptualizing is an *ISD action* by which relevant perceptions of the existing reality and conceptions of the imagined reality are interpreted, abstracted and structured according to some conceptual *model*.

Representing

Representing is an *ISD action* by which conceptions are made "visible" and proper to communicate about them. Representing yields a *model denotation* from a *concept model*.

Single-model action structure

The **single-model action structure** comprises *IS modeling actions* that involve a single *model* at a time.

Creating

Creating means an *ISD action* by which an IS *model* is conceptualized and represented for some specific use.

Refining

Refining means an *ISD action* by which an IS *model* is corrected, modified, and/or enlarged.

Testing

Testing is an *ISD action* by which a *concept model* or a *model denotation* is checked against the given quality *criteria.*

Multi-model action structure

The **multi-model action structure** comprises *IS modeling actions* that involve, some way or another, two or more IS *models* at the same time.

Transforming

Transforming is an *ISD action* by which conceptions structured according to one IS *model* are transformed into conceptions structured according to another IS *model.*

Translating

Translating is an *ISD action* by which an IS *model denotation* represented in some *language* is translated into another *language.*

Relating

Relating is an *ISD action* by which two or more IS *models* are mapped to one another by finding common *concepts* within the IS *models* or defining some "bridging" *relationships* between the *concepts* of the IS *models*.

Integrating

Integrating is an *ISD action* by which a new *model* is made by assembling their *concepts* and *relationships* of two or more IS *models.*

ISD object domain

The **ISD object domain** comprises all those *concepts* and *constructs* that *refer to* something, to which an *ISD action* is directed.

ISD deliverable

An **ISD deliverable** is an *ISD object* at which *ISD actions* are targeted.

OS_{ISD} construct
An **OS_{ISD} construct** is a *part* of the *object system* of *ISD*.

ISD management deliverable
An **ISD management deliverable** is an *ISD deliverable* produced by *ISD management actions*.

ISD execution deliverable
An **ISD execution deliverable** is an *ISD deliverable* produced by *ISD execution actions.*

IS model
An **IS model** is a *model* describing/prescribing certain aspects of an *IS*.

IS implementation
IS implementation is an ISD deliverable resulting from the implementation of one or more *IS models* (e.g. a software module, a prototype).

ISD baseline
An **ISD baseline** is a set of reviewed and approved *ISD deliverables* that represents an agreed basis for further evolution and development, and can be changed only through a formal procedure such as configuration and change management.

VII.2 ISD Perspectives

ISD perspective
An **ISD perspective** is a *perspective* with which the features of the *ISD context*, specific to the *problem* or the situation at hand, can be considered.

ISD systelogical perspective
The **ISD systelogical perspective** is an *ISD perspective*, which reveals the support the *ISD context* provides to its *utilizing system* (US_{ISD}).

ISD service
An **ISD service** means all those material or immaterial *ISD deliverables* that are produced in the *ISD context* and delivered to be exploited in the intended *IS contexts*.

ISD infological perspective
The **ISD infological perspective** is an *ISD perspective* according to which the *ISD context* is seen as a functional structure of *information processing actions* and *informational objects.*

ISD conceptual perspective
The **ISD conceptual perspective** is an *ISD perspective,* which designates the *things* the *ISD deliverables* signify.

Entity type
An **entity type** is a *generic concept* corresponding to the intensional specification of all those features that are shared by the *entities* that are regarded as *instances* of the entity type.

OS relationship type

The **OS relationship type** is a *generic concept* corresponding to the intensional specification of all those features that are shared by the *OS relationships* that are regarded as *instances* of the OS relationship type.

Entity role

An **entity role** means a particular *role*, which an *entity type* connected by an *OS relationship type* plays in that *relationship*.

Attribute

An **attribute** is a relevant *predicate* used to characterize an *entity* or an *OS relationship*.

Single-valued attribute

A **single-valued attribute** is an *attribute* that has a single *value* for a particular *entity* or *OS relationship*.

Multi-valued attribute

A **multi-valued attribute** is an *attribute* that may have many *values* for a particular *entity* or *OS relationship*.

Composite attribute

A **composite attribute** is an *attribute* that can be divided into smaller *parts* that still have independent meanings.

Atomic attribute

An **atomic attribute** is an *attribute* that is not divisible.

Derived attribute

A **derived attribute** is an *attribute* the *value* of which can be calculated from the *values* of other *attributes*, or derived in some other way from the existing *entities* and / or *OS relationships.*

OS_{IS} construct type

An **OS_{IS} construct type** means here a conceptual *construct* composed of specific *entity types* related to one another through *OS relationship types* and characterized by *attributes*.

OS_{IS} state type

An **OS_{IS} state type** means a *state type* of the *object system* or its *parts*, composed of *OS_{IS} construct types*.

OS_{IS} transition type

An **OS_{IS} transition type** is a *generic concept* corresponding to the specification of all those features that are shared by *OS_{IS} transitions*.

OS_{IS} event type

An **OS_{IS} event type** means a *generic concept* corresponding to the specification of all those features that are shared by *OS_{IS} events*, which may trigger an *OS_{IS} transition* and which may be caused by another *OS_{IS} transition*.

OS_{IS} constraint

An **OS_{IS} constraint** specifies allowed *OS_{IS} states* (static constraint) and/or allowed *OS_{IS} transitions* (dynamic constraints) between the *OS_{IS} states.*

ISD datalogical perspective

The **ISD datalogical perspective** is an *ISD perspective* according to which the *ISD context* is considered through representation-specific *concepts*, involving, besides *ISD purposes*, *ISD actors*, *ISD actions*, and *ISD deliverables*, also *ISD actors* and *ISD facilities* on a general level.

VIII. ISD Method Ontology

The **ISD method ontology** provides *concepts* and *constructs* for conceiving, understanding, structuring and representing *contextual* aspects of the ISD methods.

ISD method

An **ISD method** is an artefact anchored on certain historical, intentional and functional backgrounds and aimed to be applied and deployed as a *prescription* in the intended kinds of *ISD contexts*, in order to make organizational and technical changes in *IS's* possible or more productive. The ISD method, presented and materialized in certain forms, contains four kinds of *knowledge* bringing out how *ISD actors* carry out *ISD actions* to produce *ISD deliverables,* by means of *ISD facilities,* in an organizational and spatiotemporal *context*, in order to satisfy *ISD goals* set by *ISD stakeholders*. The ISD method is composed of descriptive and prescriptive *parts* with a large variety.

Knowledge of ISD process

Knowledge of ISD process means all the *knowledge* that concerns how to accomplish an *ISD work.*

Knowledge of application domain

Knowledge of application domain means all the *knowledge* that concerns an *information system* to be designed, its *utilization system* and its *object system*.

Knowledge of IC technology

Knowledge of IC technology means all thhe *knowledge* that concerns the search, acquirement, installation, and deployment of *hardware* and *software* for an *IS*, as well as for *ISD*.

Knowledge of human and social issues

Knowledge of human and social issues means all the *knowledge* that concerns human characteristics and behavior as well as social and organizational aspects that should be taken into account in building an *IS* and in organizing *ISD work.*

Generic ISD method

A **generic ISD method** is an *ISD method* that provides general support, such as general *approaches*, principles, *models* and guidelines, to conduct an *ISD* effort in a wide range of *ISD contexts.*

Domain-specific ISD method

A **domain-specific ISD method** is an *ISD method* that provides more domain-specific support to conduct an *ISD* effort in a specific application domain.

Organization-specific ISD method

An **organization-specific method** is an *ISD method* that provides customized support to conduct an *ISD* effort in a specific *organization*.

Project-specific ISD method

A **project-specific ISD method** is an *ISD method* that provides configured and instantiated support to conduct an *ISD* effort in a specific *project* in an instantiated manner.

Methodical view

A **methodical view** is a *point of view* from which particular aspects of an *ISD method* can be considered.

Historical view

The **historical view** is a *methodical view*, which enlightens the backgrounds of and experiences from the engineering and use of the *ISD method*. It involves both the prior ME contexts and the prior ISD contexts.

Application view

The **application view** is a *methodical view*, which outlines where and how the *ISD method* can be applied.

Generic view

The **generic view** is a *methodical view*, which provides the general understanding of the nature of the *ISD method*.

Contents view

The **contents view** is a *methodical view*, which reveals the conceptual contents of the *ISD method*.

Presentation view

The **presentation view** is a *methodical view* from which an *ISD method* is seen as a set of *expressions* presented in some *language*(s).

Physical view

The **physical view** is a *methodical view*, which reveals the appearance(s) of the *ISD method*, that is to say, the media on which the *ISD method* is made visible or "functioning".

Structural view

The **structural view** is a *methodical view* from which the *ISD method* is seen as a modular structure of *parts* with a large variety: of e.g. paradigmatic assumptions, *ISD approaches*, *ISD principles*, background and application knowledge, *concepts*, notations, ISD models, ISD techniques, *ISD rules*, and ISD guidelines.

Prior ME context

A **prior ME context** is an *ME context* that has contributed to the creation and engineering of the *ISD method*.

Prior ISD context

A **prior ISD context** is an *ISD context* in which the *ISD method* has been deployed.

Target ISD context

A **target ISD context** is an *ISD context* for which the *ISD method* is intended.

Target ME context

A **target ME context** is an *ME context* in which the *ISD method* is to be customized and instantiated for the use of a particular *organization* or *project*.

ISD model

An **ISD model** is a *model* that prescribes/describes structural and/or behavioral features of the *ISD context*(*s*).

ISD technique

An **ISD technique** is a *technique*, which guides the accomplishment of specific *actions* in the *ISD context(s)*.

ISD purpose model

An **ISD purpose model** is an *ISD model* that prescribes/describes *problems* in, *requirements* for, and/or *goals* of, the intended[182] *ISD context*, or some *part*(s) thereof.

ISD actor model

An **ISD actor model** is an *ISD model* that prescribes/describes *ISD roles*, *ISD positions*, *ISD organizational units*, *persons* and/or *groups* participating one way or another in the intended *ISD context*.

ISD action model

An **ISD action model** is an *ISD model* that prescribes/describes *ISD actions* and their *relationships* in the intended *ISD context*.

ISD deliverable model

An **ISD deliverable model** is an *ISD model* that prescribes/describes the structure and presentation of *ISD deliverables* and how they are related in the intended *ISD context*.

ISD data model

An **ISD data model** is an *ISD model* that prescribes/describes the conceptual contents of the *ISD deliverables* in the intended *ISD context*.

ISD facility model

An **ISD facility model** is an *ISD model* that prescribes/describes *resources* and *tools* available and used in the intended *ISD context*.

ISD location model

An **ISD location model** is an *ISD model* that prescribes/describes the nature, structure and features of *locations*, whether *physical* or *logical*, involved in the intended *ISD context*.

ISD time model

An **ISD time model** is an *ISD model* that prescribes/describes the *time system* used in the intended *ISD context*.

ISD ID model

An **ISD ID model** is an *ISD model* that prescribes/describes *inter-domain* (ID) features of the intended *ISD context*.

---

[182] An intended ISD context means or a target ISD context ('prescribe') or a prior ISD context ('describe').

ISD systelogical model

An **ISD systelogical model** is an *ISD model* that describes/prescribes the support the intended *ISD* provide or should provide to its *utilizing system* (US_{ISD}), as well as the assumptions on the *target IS's* and their *utilizing systems* (US_{IS}).

ISD infological model

An **ISD infological model** is an *ISD model* that describes/prescribes the *purposes*, *actions* and *deliverables* of the intended *ISD context*.

ISD conceptual model

An **ISD conceptual model** is an *ISD model* that describes/prescribes the conceptual contents of the *deliverables* of the intended *ISD context*.

ISD datalogical model

An **ISD datalogical model** is an *ISD model* that describes/prescribes the *purposes*, *actors*, *actions*, *deliverables* and *tools* of the intended *ISD context,* last two on a general level.

ISD physical model

An **ISD physical model** is an *ISD model* that describes/prescribes, besides the features mentioned above, yet on a more concrete level, also spatial and temporal features of the intended *ISD context*, and all as being *instantiated* into a particular *ISD context*.

ISD IP model

An **ISD IP model** is an *ISD model* that describes/prescribes features of the intended *ISD context* from multiple *ISD perspectives*.

ISD methodical framework

An **ISD methodical framework** is composed of *IS meta models* and/or *ISD meta models*.

ISD methodical skeleton

An **ISD methodical skeleton** is a normative *prescription* for the *ISD context*, structuring and guiding the I*SD process* on a general level.

Methodical tool kit

A **methodical tool kit** is a collection of more or less unrelated methodical *parts*, which do not, as such, constitute any coherent and concrete *method* for ISD.

ISD method component

An **ISD method component** is a well-defined *part* of the *ISD method* that can be integrated to other *ISD method components* to form a coherent and consistent *ISD method*.

Contextual ISD method component

A c**ontextual ISD method component** is an *ISD method component* that contains *descriptions/prescriptions* of features of *ISD* within several *contextual domains*.

Domain-based ISD method component

A **domain-based ISD method component** is an *ISD method component* that contains *descriptions/prescriptions* of features of *ISD* with one or at most two *contextual domains*

Ontological component

An **ontological component** is an *ISD method component*, which provides *concepts* and *constructs* for conceptual *modeling*.

Notational component

A **notational component** is an *ISD method component*, which provides a set of *symbols* (without any predefined *semantics*).

Action-based component

An **action-based component** is an *ISD method component*, which mainly describes/prescribes *ISD actions*.

Actor-based component

An **actor-based component** is an *ISD method component*, which describes/prescribes *ISD actors* and how they are related (e.g. an *organisational structure*).

Tool-based component

A **tool-based component** is an *ISD method component*, which describes/prescribes elements and architecture of a *computerized information system*.

Construct component

A **construct component** is an *ISD method component*, which cannot be decomposed into smaller *parts* without loosing some of its meaningfulness and integratability.

Contextual interface

A **contextual interface** of an *ISD method component* means a white-box like description of those *contextual relationships* through which an *ISD method component* can be integrated into other *ISD method components*. The *contextual relationships* are *inter-domain relationships* and/or *intra-domain relationships.*

IX. ME Ontology

The **ME ontology** provides *concepts* and *constructs* to conceive, understand, structure, and represent *contextual* features of *method engineering*.

Method engineering

Method engineering (ME) means all those *actions* by which an *ISD method* is developed, and later possibly customized and configured to fit the needs of an *organization* and/or an *ISD project*.

ME strategy

An **ME strategy** is a generic way of accomplishing an *ME* effort.

Creation

Creation means an *ME strategy*, also known as the "greenfield" or "from scratch" strategy, that is applied in a situation where no *method* is available to be used as a basis for *ME*.

Integration

Integration means an *ME strategy* according to which an *ISD method* is engineered by assembling *components* of existing *ISD methods*.

Adaptation

Adaptation means *an ME strategy* according to which an *ISD method* is engineered by dropping off or modifying some *part*(s) of an existing *ISD method*, or extending an existing *ISD method* with some new *part*(s).

Customization

Customization means an *ME process* by which an *organization-specific ISD method* is derived from some *generic ISD method* (or *domain-specific ISD method)* by adjusting it with organizational features that fit the traditions, culture, infrastructure, management policies, etc. of the target *organization*.

Configuration

Configuration means an *ME process* by which a *project-specific ISD method* is derived from an *organization-specific ISD method*.

Realization

Realization means an *ME process* by which a *project-specific ISD method* is put into action.

Decustomization

Decustomization is an *ME process* by which a *generic ISD method* is engineered by clearing an *organization-specific ISD method* from the *knowledge* specific to a certain *organization*.

Deconfiguration

Deconfiguration means an *ME process* by which an *organization-specific ISD method* is engineered by abstracting project-specific *knowledge* from an existing *ISD method*.

Method engineering context

A **method engineering context** is a *context* in which *ME actors* carry out *ME actions* of *(de)customization*, *(de)configuration*, *realization*, and/or *abstraction* to produce a new or improved *ISD method*, with *ME facilities* in a certain organizational and spatiotemporal *context*, in order to satisfy *ME goals* set by *ME stakeholders*.

Method development context

A **method development context** is an *ME context* that aims to engineer a *generic ISD method* or a *domain-specific ISD method*.

Method customization context

A **method customization context** is an *ME context* that aims to attain an *organization-specific ISD method*.

Method configuration context

A **method configuration context** is an *ME context* that aims to engineer a *project-specific ISD method*.

Prior ME context

A **prior ME context** means a *context*, which has contributed to the *ISD method* that is under consideration/engineering in the *ME context* at hand.

Target ME context

A **target ME context** means a *context* in which the *ISD method* under engineering is later to be customized, configured and/or realized for the use of certain *ISD contexts*.

Prior ISD context

A **prior ISD context** means a *context* in which the *ISD method(s)* interested by the *ME context* at hand have been applied.

Target ISD context

A **target ISD context** means a *context* for which the *ME* effort at hand has been launched.

IX.1 ME Domains

ME purpose domain

The **ME purpose domain** embraces all those *concepts* and *constructs* that refer to *goals*, motives, or intentions of someone or something in the *ME context.*

ME goal

An **ME goal** expresses a desired *state* or *event* with qualities and quantities related to the *ME context* as a *whole*, or to some *part* of it.

Hard ME goal

A **hard ME goal** has pre-specified *criteria* for the assessment of the fulfilment.

Soft ME goal

A **soft ME goal** has no pre-specified *criteria* for the assessment of the fulfilment.

ME requirement

An **ME requirement** is some quality or performance demanded from the *ME context,* or from some *part(s)* thereof.

ME problem

An **ME problem** is a perceived deviation from a desired *state* or way of doing, which may lead to specifying one or more *ME requirements* and set up one or more *ME goals.*

ISDM purpose

An **ISDM purpose** is an *ME goal* or an *ME reason* pertaining to an *ISD method.*

ME actor domain

The **ME actor domain** consists of all those *concepts* and *constructs* that refer to human and active *part* of the *ME context*.

ME actor

An **ME actor** is a human thing or an administrative *thing* that is, one way or another, involved in the *ME context*.

Human ME actor

A **human ME actor** means an individual *person* or a *group* of *persons* contributing to the *ME* work.

ME role

An **ME role** is a collection of *ME* responsibilities and *ME* authorities.

ME position

An **ME position** is a *position*, composed of the defined *ME roles* and occupied by a *human ME actor.*

ME engineer
A **method engineer** is an *ME role* in which a *human ME actor* has the main responsibility for *ME actions* in an *ME* effort.

ME project manager
An **ME project manager** is an *ME role* in which a *human ME actor* makes *plans* of and *decisions* on how to organize an *ME* effort.

ME stakeholder
An **ME stakeholder** is an *ME actor* who plays in any *ME role.*

Method expert
A **method expert** is a *person* who has a deep understanding of *methods* generally, and of some specific *method(s)* in particular.

Tool expert
A **tool expert** is a *person*, who has familiarized oneself with *tools* used in *method engineering* (i.e. CAME tools and/or MetaCase tools) and in ISD (i.e. CASE tools).

Theory expert
A **theory expert** is a *person*, who has special *knowledge* on theoretical and methodological issues of *method engineering*.

ME organization
An **ME organization** is a composition of *ME positions* with a coherent set of organizational *goals*, authorities and responsibilities.

ME action domain
The **ME action domain** comprises all those *concepts* and *constructs* that refer to deeds or *events* in the *ME context.*

ISD modeling structure
The **ISD modeling structure** is composed of *ME actions* modeling *ISD* on two levels.

Metamodeling
Metamodeling is a *modeling process*, which takes place on one *level* of abstraction and logic higher than the standard *modeling process*.

ME workflow structure
The **ME workflow structure** is composed of various *ME workflows.*

ME workflow
An **ME workflow** is coherent composition of *ME actions*, (a) which are organized to accomplish some *ME process*, (b) which share the same target of *action*, and (c) which produce results valuable for *ME stakeholders.*

ME task
An **ME task** is a *part* of an *ME workflow.*

ISDM requirements engineering
ISDM requirements engineering means an *ME workflow*, which aims to identify and elicit *ME stakeholders' requirements* on the nature, contents and structure of the *ISD method.*

ISDM analysis
ISDM analysis means an *ME workflow*, which aims to produce high-level descriptions of the *ISD method*, meaning that the *ISD method* is considered from the *ISD infological perspective* and the *ISD conceptual perspective.*

ISDM design

ISDM design means an *ME workflow,* which aims to produce more elaborated descriptions of the *ISD method,* meaning that the *ISD method* is considered from the *ISD datalogical perspective.*

ISDM implementation

ISDM implementation means an *ME workflow,* which aims to produce concrete descriptions/prescriptions of the *ISD context* from the *ISD physical perspective.*

ISDM evaluation

ISDM evaluation means an *ME workflow,* which aims to produce assessments of one or more *ISD methods* according to the defined *criteria.*

ME object domain

The **ME object domain** comprises all those *concepts* and *constructs* that refer to something, to which *ME actions* are targeted.

ME deliverable

An **ME deliverable** is an *object* to which *ME actions* are targeted.

ME management deliverable

An **ME management deliverable** is an *ME deliverable* that is produced by *ME management actions.*

ME execution deliverable

An **ME execution deliverable** is an *ME deliverable* that is produced by *ME execution actions.*

OS_{ME} construct

An **OS_{ME} construct** is a *part* of the *object system* of *ME.*

ME baseline

An **ME baseline** is a set of reviewed and approved *ME deliverables.*

IX.2 ME Perspectives

ME perspective

An **ME perspective** is a *perspective* with which the features of the *ME context*, specific to the *problem* or the situation at hand, can be considered.

ME systelogical perspective

The **ME systelogical perspective** is an *ME perspective* that reveals the support *method engineering* provides to its *utilizing system* (US_{ME}).

ME service

An **ME service** means a material and immaterial *ME deliverable* that is produced in the *ME context* and delivered to be utilized in the *target ISD contexts.*

ME infological perspective

The **ME infological perspective** is an *ME perspective* from which the *ME context* is seen as a functional structure of *information processing* and *informational objects.*

ME conceptual perspective

The **ME conceptual perspective** is an *ME perspective* that addresses the conceptual contents of the *ME deliverables.*

OS_{ISD} construct type

An **OS_{ISD} construct type** in OS_{ISD} means a conceptual *construct* composed of specific *entity types* related to one another with *OS relationship types* and characterized by *attributes*.

OS_{ISD} state type

An **OS_{ISD} state type** means a *state type* of the *object system* or its *parts*, composed of *OS_{ISD} construct types*.

OS_{ISD} transition type

An **OS_{ISD} transition type** is a *generic concept* corresponding to the specification of all those features that are shared by *OS_{ISD} transitions*.

OS_{ISD} event type

An **OS_{ISD} event type** means a *generic concept* corresponding to the specification of all those features that are shared by *OS_{ISD} events*, which may trigger an *OS_{ISD} transition* and which may be caused by another *OS_{ISD} transition*.

OS_{ISD} constraint

An **OS_{ISD} constraint** specifies allowed *OS_{ISD} states* (static constraint) and/or allowed *OS_{ISD} transitions* (dynamic constraint) between the *OS_{ISD} states*.

OS_{is} construct meta type

An **OS_{is} construct meta type** in OS_{ME} is composed of classes related through associations and class roles to one another.

ME datalogical perspective

The **ME datalogical perspective** is an *ME perspective* from which *ME* is seen as a *context* in which *ME deliverables*, represented in some *language*, are processed by *ME actors* for certain *purposes* with some computer-aided *ME tools*.

X. ME Method Ontology

The **ME method ontology** provides *concepts* and *constructs* for conceiving, understanding, structuring and representing *contextual* aspects of the *ME methods*.

ME method

An **ME method** is an artifact anchored on historical, intentional and functional backgrounds and aimed to be applied and deployed as a *prescription* in the intended kinds of *ME contexts*, in order to make organizational and technical changes in *ISD contexts* possible or more productive. The ME method, presented and materialized in several forms, contains *knowledge* bringing out how *ME actors* carry out *ME actions* to produce *ME deliverables*, by means of *ME facilities*, in an organizational and spatiotemporal *context*, in order to satisfy *ME goals* set by *ME stakeholders*. The ME method is composed of descriptive and prescriptive *parts* in a large variety.

Generic ME method

A **generic ME method** is an *ME method*, which provides general *approaches*, principles, *models* and guidelines to conduct *ME* efforts in a wide range of *ME contexts*.

Domain-specific ME method

A **domain-specific ME method** is an *ME method*, which provides more domain-specific support to conduct *ME* efforts in a specific application domain.

Organization-specific ME method

An **organization-specific ME method** is an *ME method*, which provides customized support to conduct *ME* efforts in a specific *organization*.

Project-specific ME method

A **project-specific ME method** is an *ME method*, which provides configured and instantiated support to accomplish a particular *ME* effort.

Prior RW context

A **prior RW context** is an *RW context*, which has contributed to the creation and engineering of the *ME method*.

Target RW context

The **target RW context** means an *RW context* in which the *ME method* is to be elaborated, customized, configured and/or instantiated for the use of a particular *organization* or ME *project*.

ME strategy

An **ME strategy** means a generic way of accomplishing an ME effort, or a part thereof.

ME approach

An **ME approach** means a generic way of perceiving certain aspects of *ME* and/or a way of working in *ME*.

Main ME principle

A **main ME principle** expresses essential aspects of a specific way to structure, accomplish, and/or manage the *ME process.*

ME method component

An **ME method component** is a well-defined *part* of the *ME method* that can be integrated to other *ME method components* to form a coherent and consistent *ME method*.

ME model

An **ME model** is a *model* that describes/prescribes structural, functional and/or behavioral features of the *ME context*.

ME technique

An **ME technique** is a *technique*, which guides the accomplishment of specific *actions* in the *ME context.*

ME contextual models

The **ME contextual models** mean *ME models* that can be classified into eight categories according to which *ME domain(s)* they address.

ME perspective models

The **ME perspective models** mean *ME models* that can be classified into five categories according to which *ME perspective(s)* they address

APPENDIX 2: ONTOLOGY REPRESENTATION LANGUAGE

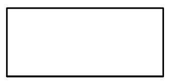
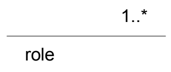
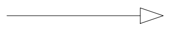
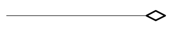
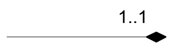
This thesis provides a large number of meta models for describing component ontologies in OntoFrame. In this appendix we specify the ontology representation language, in which the meta models are presented.

The ontology representation language is closely based on UML version 1.4 (OMG 2001)[183]. We deploy only a small part of the UML language. Following the strategy of MOF (OMG 2002) we have selected only those features of UML, which are feasible in describing ontologies. That means that even from the set of the concepts and constructs of the MOF we have excluded some concepts as unnecessary for our purposes. Compared to UML, we have also made changes in the semantics of the following concepts:

- *Association:*
  In UML an association is defined as a structural relationship between two or more classes. In the ontology representation language an association is defined as a binary relationship. This limitation is made in concordance with MOF.
- *Aggregation* and *composition*:
  In UML an aggregation is defined as an association that "specifies a whole-part relationship between the whole and its part" (p. Booch *et al.* 1999, 458). It represents "a 'has-a' relationship, meaning that an object of the whole has objects of the part" (Booch *et al.* 1999, 67). A composition is defined "a form of aggregation with strong ownership and coincident lifetime of the parts by the whole" (Booch *et al.* 1999, 460). We define these concepts to stand for the relationships between the whole and its part in such a way that the parts are assumed to be inter-related. This assumption about the internal structure of the parts within the whole is also included in the notion of an aggregation in OML (Firesmith *et al.* 1997).

In the table below we define the UML-based concepts and notation for the ontology representation language. The definitions are based on OMG(2001) and OMG(2002).

[183] Nowadays, there is also the OMG standard for UML 2.0 (OMG 2003), but it was not available when we started our work.

Ontology representation language		
Concepts	**Definition**	**Notation**
Class	A description of a set of concepts that share the same predicates.	
Association	An semantic connection between two classes, each one associated in a specific role (ClassRole). For each association end a range of allowed cardinalities is specified with the multiplicity.	
Generalization	A taxonomic association between a more general class and a more specific class.	
Aggregation	A special form of association that specifies a whole-part relationship between a whole and its part in such a way that the parts in the whole are inter-related.	
Composition	A special form of aggregation which requires that a part instance is included in at most one whole at a time, and that the lifetimes of the parts are coincident with the lifetime of the whole.	

YHTEENVETO (FINNISH SUMMARY)

Tietojärjestelmien suunnittelumenetelmiä kehitetään, räätälöidään ja sovitetaan usein intuitiivisesti ja improvisoiden. Näin toteutettuna menetelmäkehityksellä on vaarana tuottaa vaikeasti ymmärrettäviä, huonosti soveltuvia ja tehottomasti sovellettavia suunnittelumenetelmiä. Improvisointiin nojaavalle menetelmäkehitykselle on usein ominaista myös suurempi resurssitarve. Merkittävänä syynä vallitsevalle tilanteelle on se, että menetelmäkehitykseltä itseltään puuttuu kunnollinen menetelmätuki. Kirjallisuudessa on kyllä esitetty laaja kirjo menetelmäkehityksen strategioita, lähestymistapoja, tekniikoita ja askeltasoisia proseduureja, mutta nämä ovat sittenkin vain osaratkaisuja. Kokonaisvaltaista menetelmää menetelmäkehitykseen ei ole tarjolla. Toisaalta ehdotetut ratkaisutkin perustuvat suuressa määrin olemassa olevien käytäntöjen kirjaamiseen. Niiltä puuttuu lähes kokonaan laaja-alainen ja teoreettinen käsiteperusta.

Tämän väitöskirjatyön tavoitteena on ollut kehittää ensiksikin käsitteellinen perusta, joka auttaa tunnistamaan, ymmärtämään, jäsentämään ja esittämään tietojärjestelmien suunnittelumenetelmien luonteeseen, sisältöön, rakenteeseen ja kehittämiseen liittyviä piirteitä ja ilmiöitä. Toisena tavoitteena on ollut rakentaa teoriapohjaista menetelmätukea menetelmäkehitykselle. Erityisenä pyrkimyksenä on ollut hyödyntää kontekstuaalisia piirteitä esiin nostavia teorioita (esim. semantiikka, pragmatiikkaa, toimintateoria) ja luoda niiden pohjalta lähestymistapoja, käsitteistöä, malleja ja ohjeistoja menetelmäkehitykseen.

Väitöskirjassa esitetään kaksi konstruktiota, ontologinen kehys ja menetelmärunko. Ontologinen kehys, nimeltään OntoFrame, on moniulotteinen käsitteellinen kehys, joka koostuu eri tasoisista ontologioista. Yleisluonteisin ontologia on rakentunut ”universaaleista” käsitteistä kuten olevaisesta, suhteesta, roolista, ominaisuudesta ja näkökulmasta. Generatiivisen lähestymistavan mukaisesti yleisemmistä ontologioista on johdettu spesifisempien ontologioiden käsitteet. Mitä spesifisemmälle tasolle edetään, sitä kontekstuaalisempia käsitteitä ontologiat sisältävät. Spesifisimmät ontologiat koskevat menetelmäkehitystä ja sitä tukevia menetelmäkomponentteja. OntoFrame on rakennettu yhtäältä relevanttien teorioiden päälle (deduktiivinen lähestymistapa) ja toisaalta käyttämällä hyväksi olemassa olevia kehyksiä, viitemalleja, metamalleja ja ontologioita (induktiivinen lähestymistapa). Kehykseen sisältyvien käsitteiden määritelmät esitetään yhtenäisenä sanastona tutkimuksen liitteessä. Kunkin ontologian käsitteistä ja käsiterakenteista esitetään myös yksityiskohtaiset UML-pohjaiset metamallit, jotka auttavat muodostamaan kokonaiskäsityksen ontologioista ja lisäävät esityksen täsmällisyyttä.

Menetelmärunko, MEMES, on tarkoitettu menetelmien yleistasoisten määritysten tekemiseen ja arviointiin. Se jäsentää menetelmäkehityksen viiteen tehtäväkokonaisuuteen ja tarjoaa kolmelle niistä lähestymistapoja, periaatteita ja askeleita. Nämä tehtäväkokonaisuudet ovat: menetelmävaatimusten määrittäminen, menetelmäanalyysi (so. karkean tason suunnittelu) ja menetelmäarviointi. Vaatimusten määrittämisessä tehdään ensin ratkaisu kontingenssiviitekehyksen käytöstä ja valitaan sovellettava lähestymistapa. Sen jälkeen karakterisoidaan ja analysoidaan aiempia tietojärjestelmien suunnittelutilanteita, käsillä olevaa menetelmäkehitystilannetta sekä olemassa olevia menetelmiä. Analyysien tulosten perusteella määritellään vaatimukset ja tavoitteet menetelmäkehitykselle. Seuraavassa vaiheessa mallinnetaan konstruoitavan menetelmän kohteena olevaa tietojärjestelmän suunnittelukontekstia infologisesta ja käsitteellisestä näkökulmasta. Tämä työ sisältää muiden muassa tietojärjestelmää koskevien ontologioiden valinnan ja sovittamisen menetelmää varten. Menetelmäarvioinnissa valitaan tai määritellään arviointikriteerit, arviointitekniikka ja suoritetaan itse arviointi annettuja askeleita soveltaen.

Tutkimuksessa on sovellettu konstruktiivista tutkimusotetta ja suunnitteluteoreettista paradigmaa. Ontologista kehystä ja menetelmärunkoa arvioidaan vertailemalla niitä ja niiden osia laajasti olemassa olevaan kirjallisuuteen. Menetelmäkehystä arvioidaan myös empiirisin menetelmin.

Väitöskirjan tuloksia voivat hyödyntää menetelmäkehitystä tekevät ja tutkivat organisaatiot. Ontologinen kehys tarjoaa käsitteellisen perustan arvioida ja vertailla olemassa olevia tietojärjestelmien suunnittelumenetelmiä sekä konstruoida uusia. Laajana kehyksenä se edesauttaa myös siltojen rakentamista eri tieteenalojen, lähestymistapojen ja aikakausien käsitteiden ja käsitysten välille. Metodirunkoa voidaan käyttää tarkennettuna ja sovitettuna käytännön menetelmäkehityksessä. Sen pohjalta voidaan räätälöidä erilaisiin strategioihin, lähestymistapoihin ja organisaationalisiin tilanteisiin sopivia menetelmäkehityksen menetelmiä ja tekniikoita.