Ari Peltoniemi

TOWARDS A META-METHOD FOR THE ENGINEERING OF SITUATIONAL EVALUATION METHODS FOR DOMAIN-SPECIFIC MODELING TOOLS

UNIVERSITY OF JYVÄSKYLÄ
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS
2015

ABSTRACT

Peltoniemi, Ari
Towards a Meta-Method for the Engineering of Situational Evaluation Methods for Domain-Specific Modeling Tools
Jyväskylä: University of Jyväskylä, 2014, 116 p.
Information Systems Science, Master's Thesis
Supervisor: Leppänen, Mauri

Domain-Specific Modeling (DSM) is an approach to Information Systems Development (ISD) in which the abstraction level of development is raised from the solution domain to the problem domain. DSM enables the automation of ISD, particularly in narrow and well-established domains, in which the domain concepts, rules and semantics can be meaningfully specified as constructs of DSM languages. DSM tools provide facilities for DSM language specification and application as well as model transformation. DSM tools are typically evaluated by the industry for the justification of tool acquisitions. DSM tools are also evaluated for research purposes. In order to assure the validity of the results, an evaluation method must address the situational context of the evaluation as well as its multi-disciplinary dimensions. The current literature provides very limited support for the engineering of situational evaluation methods for DSM tools. The primary objective of the study is to investigate how to methodically support the engineering of situational evaluation methods for DSM tools. A practical need for the method support was identified in a case study, in which DSM tools were evaluated in an industrial context. The premise of the study suggests that the application of Situational Method Engineering (SME) principles to the evaluation of DSM tools would provide a potential solution. The Design Science Research (DSR) approach was applied as the research framework for the study. Two artifacts were designed and evaluated, according to the principles of DSR: 1) an evaluation criteria checklist for DSM tools, and 2) a baseline method for the engineering of situational evaluation methods for DSM tools. The checklist is designed for evaluators, to be used as a practical guideline in the situational formulation of the evaluation criteria for DSM tools. The application of the checklist also promotes the commensuration of the evaluation results. The conceptual baseline method is designed to be instantiated by method engineers in the engineering of situational evaluation methods for DSM tools. The main contribution of the study is a design theory or a Meta-Method for the engineering of situational evaluation methods for DSM tools. Meta-Method is conceptually and empirically evaluated. Future research is required to confirm the findings and further elaborate Meta-Method.

Keywords: domain-specific modeling, DSM tools, evaluation criteria, situational evaluation method, method engineering, design science research, case study

TIIVISTELMÄ (FINNISH ABSTRACT)

Peltoniemi, Ari
Kohti metamenetelmää sovellusaluemallinnuksen välineiden tilannekohtaisten arviointimenetelmien kehitykseen
Jyväskylä: Jyväskylän yliopisto, 2014, 116 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Leppänen, Mauri

Sovellusaluemallinnus (Domain-Specific Modeling, DSM) on eräs ohjelmistotuotannon lähestymistavoista, jossa sovelluskehityksen abstraktiotasoa nostetaan ohjelmoinnista sovellusaluekeskeiseen mallinnukseen. DSM mahdollistaa sovelluskehityksen automatisoinnin erityisesti kapeilla ja vakiintuneilla sovellusalueilla, joiden käsitteet, säännöt ja merkitykset soveltuvat DSM-kielten konstruktioiksi. DSM-välineet tarjoavat työkaluja DSM-kielten määrittelyyn ja käyttöön sekä sovellusmallien transformaatioihin. Teollisuudessa DSM-välineiden arviointeja tehdään tyypillisesti välinehankintojen valmistelun yhteydessä. Arviointeja suoritetaan myös tieteellisen tutkimuksen näkökulmasta. Arviointitulosten validiteetin varmistamiseksi DSM-välineiden arviointimenetelmän tulee ottaa huomioon arvioinnin tilannekohtainen konteksti sekä sen monitieteiset dimensiot. Kirjallisuudessa esitetään hyvin rajoitetusti menetelmiä DSM-välineiden tilannekohtaiseen arviointiin. Tämän tutkimuksen ensisijaisena tavoitteena on selvittää, miten DSM-välineiden arviointimenetelmien kehitystä tilannekohtaisessa kontekstissa voidaan tukea menetelmällisesti. Menetelmätuen käytännön tarve todettiin teollisuusalan yritykselle suoritetun tapaustutkimuksen yhteydessä. Tutkimus esittää ratkaisun lähtökohdaksi tilannekohtaisen menetelmäkehityksen (Situational Method Engineering, SME) periaatteiden soveltamista DSM-välineiden arviointiin. Tutkimusviitekehyksenä käytettiin suunnittelutieteellistä lähestymistapaa, jonka mukaisesti on muodostettu ja arvioitu kaksi artefaktia: 1) DSM-välineiden arviointikriteerien tarkistuslista, ja 2) lähtökohtamenetelmä DSM-välineiden arviointimenetelmien tilannekohtaiseen kehitykseen. Tarkistuslista on suunniteltu arvioijien praktiseksi ohjesäännöksi DSM-välineiden arviointikriteeristöjen tilannekohtaiseen muodostamiseen sekä arviointitulosten yhteismitallisuuden edistämiseen. Käsitteellinen lähtökohtamenetelmä on suunniteltu menetelmäkehittäjien käyttöön, erityisesti DSM-välineiden tilannekohtaisten arviointimenetelmien kehitykseen. Tutkimuksen pääkontribuutio on esitettyjen artefaktien pohjalta muodostettu nk. suunnitteluteoria eli Metamenetelmä DSM-välineiden tilannekohtaisten arviointimenetelmien kehitykseen. Metamenetelmä arvioitiin käsitteellisesti ja empiirisesti. Metamenetelmän varmentamiseen ja kehittämiseen tarvitaan jatkotutkimusta.

Asiasanat: sovellusaluemallinnus, DSM-väline, arviointikriteeristö, tilannekohtainen arviointimenetelmä, menetelmäkehitys, suunnittelutiede, tapaustutkim.

ACKNOWLEDGEMENTS

This research was conducted in the SCOPE project of the Department of Mathematical Information Technology, operating under the Faculty of Information Technology at the University of Jyväskylä. SCOPE was funded by Tekes (the Finnish Funding Agency for Technology and Innovation) as well as the University of Jyväskylä and Airbus Defense and Space. The research was also supported by Research and Training Institute ReTI Foundation. I extend my sincere gratitude and appreciation to these organizations and the many people who made this thesis possible.

First, I would like thank my supervisor, Dr. Mauri Leppänen, who has patiently, insightfully, and proficiently guided me throughout the study. Second, I would like to express my gratitude to Dr. Erkki Kurkinen and Mr. Antti Raunio for facilitating the active industry-academia collaboration that was essential in the conduct of the research. Third, I would like to thank Dr. Tommo Reti, who facilitated the finalization of the thesis. Fourth, I would like to express my gratitude to Dr. Juha-Pekka Tolvanen for evaluating the thesis. Finally, I would like to thank my family for the continuous support during the research process.

This thesis was prepared in various geographical locations such as Jyväskylä and Kankaanpää, Finland, and Geneva, Switzerland. I am grateful for the insights and support of my colleagues, friends, and acquaintances, who have inspired and guided me in my academic and professional endeavors. I have had a privileged opportunity to learn from exceptional individuals who have profoundly contributed to my view and understanding of our discipline, life, and the world. Thank you.

Jyväskylä
April 2015

Ari Peltoniemi

FIGURES

TABLES

CONTENTS

1 INTRODUCTION

In this chapter the background and motivation for the thesis are first discussed. Second, the research questions and objectives for the research are defined. Third, the methodology for the research is discussed, including the research framework and research process. Finally, we describe the structure of the thesis.

1.1 Background and Motivation

Domain-Specific Modeling (DSM) is an approach to Information Systems Development (ISD) in which the abstraction level of development is raised from the solution domain to the problem domain (Kelly & Tolvanen, 2008, p. 3). The approach aims to respond to challenges in productivity and quality in industrial settings of ever increasing software system complexity and decreasing time to market. DSM is applied in varied, often narrow and well-established domains in which the concepts, rules, and semantics of the domain can be appropriately captured in the specifications of DSM Languages (DSML), and where the automation of software artifact generation from models is viable (Kärnä et al., 2009).

DSM tools differ from conventional ISD tools by providing an additional abstraction layer for DSML construction and the facilities for applying DSMLs (Achilleos et al., 2007; Langlois et al., 2007). Furthermore, model transformation facilities are utilized in the generation of software as well as other artifacts of persistent quality from models (Czarnecki & Helsen, 2003). Some of the most established DSM tools available are MetaEdit+ (Tolvanen et al., 2007), Eclipse Modeling Tools (Gronback, 2009), and Obeo Designer (Obeo, 2014). Many of the current DSM tools are based on open source components, upon which commercial solutions are built by providing production-ready quality, comprehensive customer support, and a streamlined user experience.

In practice, a diverse set of ISD tools can be considered when selecting a DSM tool for a given situation, as there are various approaches to DSM tool implementation (Atkinson & Kühne, 2005; Mohagheghi & Haugen, 2010; Saraiva & da Silva, 2008). DSM tools are provided with versatile architectures, capabilities, and maturity levels, delivered as open source and/or commercial software products. Ultimately, DSM tools must provide graphical facilities for language specification and application as well as model transformation facilities (El Kouhen et al., 2012). In addition, a robust architecture, adequate user support, usability, reliability, and development life-cycle support are of essence (Kelly & Tolvanen, 2008, p. 61).

The suitability of a specific DSM tool to a given situational context is a multifaceted and non-trivial issue, as it is dependent on multiple factors, such as interoperability and maturity of the tool, initial and operational costs as well as various other technical and organizational domain requirements (Lukman & Mernik, 2008). The wide range of heterogeneity among DSM tools and application domains makes the selection of the optimal tool candidate challenging. A method support that addresses the situational characteristics of the evaluation context as well as appropriately formulated evaluation criteria are required for the optimal execution of the evaluation effort (Lundell & Lings, 2002).

Evaluation is the process of determining the merit, worth, or significance of things (Scriven, 2003). Evaluation criteria are a selected subset of properties of things, by which the things are evaluated (Scriven, 2001). DSM tools are typically evaluated by the industry for the purpose of justifying the acquisition of a tool. DSM tools are also evaluated for research purposes. To our knowledge, there is no commonly accepted unified set of evaluation criteria or a method to be considered in DSM tool evaluations. This implies that DSM tool studies have no standard to be used as a reference for criteria formulation, nor a proven method for measuring the selected criteria. Hence, the study results are often rendered incommensurate. Many studies have been reported of using considerably divergent means of evaluation (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Sivonen, 2008; Vasiljević et al., 2013).

Evaluation efforts conducted in situational contexts face various facets of complexity, propagated from the interactions of people, organizations, and technology (Lundell & Lings, 2004b; Hevner et al., 2004). According to the widely accepted CCP model (Stockdale, Standing, Love & Irani, 2008), evaluation of Information Systems (IS) can be observed via three dimensions: Content, Context, and Process. Lundell and Lings (2004) suggest three dimensions of interest in situational evaluations of ISD tools: Stakeholders, Context, and Activity. Furthermore, Kitchenham (1996) suggests that the following sociological dimensions affect the participants of such evaluation efforts: novelty effects and expectation effects. As the evaluation of DSM tools is conducted in a situational context, the aforementioned socio-technical dimensions affect the effort. A few evaluation methods have been proposed that, to a varying extent, regard some of these dimensions as issues of concern, such as ISO 14102 (ISO, 2008), 2G (Lundell & Lings, 2003), RUP (Kruchten, 2004), and DESMET (Kitchenham, 1996).

The main motivation of our work is to design artifacts that enable the evaluation of DSM tools in a situational context. We seek the foundations for a potential solution from the areas of Situational Method Engineering (SME) (Henderson-Sellers et al., 2014,), existing evaluation methods for ISD tools (ISO, 2008; Kruchten, 2004; Lukman & Mernik, 2008; Wheeler, 2011; Morera, 2002; Lundell & Lings, 2003; Kitchenham, 1996), and previous studies on the evaluation of DSM tools (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Sivonen, 2008; Vasiljević et al., 2013). We seek the potential of SME to address the socio-technical dimensions of evaluation in the method construction/tailoring stages as situational factors. Our premise is that various proven characteristics that SME provides for the engineering of ISD methods, such as flexibility, adaptability, modularity, reusability, and reference to situational aspects (Bucher et al., 2008), will also provide a useful foundation for the engineering of situational evaluation methods for DSM tools. According to the principles of SME, such an endeavor requires a reusable baseline method, from which the situational methods are derived and enacted in the evaluations of DSM tools.

1.2 Research Questions and Objectives

The research problem of the thesis is: *How to methodically support the engineering and enactment of situational evaluation methods for DSM tools?* The research problem is many-fold, as it addresses various facets of a baseline method: a method base, situational factors, and construction guidelines. The *method base* includes the method elements that represent the activities, outcomes, and roles that are provided for the engineering of the situational evaluation methods. Furthermore, the preparation of the guidance for the activity of evaluation framework development is one of the key concerns in our work. The *situational factors* are the characteristics of the specific context in which the evaluation is conducted, affecting the engineering of the situational evaluation methods. The *construction guidelines* are the technical principles and techniques that are employed in the construction/tailoring of the methods. The research problem can be decomposed into the following research questions:

- RQ1: What are Domain-Specific Modeling and DSM tools?
- RQ2: Which evaluation criteria are proposed in the literature for DSM tools and how to classify them?
- RQ3: What are the situational factors affecting the engineering and enactment of the situational evaluation methods for DSM tools?
- RQ4: Which method elements are proposed in the literature for the evaluation of ISD tools and how to classify them?
- RQ5: How to engineer situational evaluation methods for DSM tools?

The main objective of the study is *to conceptually and empirically investigate the engineering and enactment of situational evaluation methods for DSM tools, and based on the findings, present a design theory or a Meta-Method that addresses such phenomena.* In order to develop Meta-Method, two artifacts are designed and evaluated: Artifact I and Artifact II. *Artifact I* is an evaluation criteria checklist for DSM tools. *Artifact II* is a baseline method for the engineering of situational evaluation methods for DSM tools, including a method base, situational factors, and construction guidelines.

The design goal of Artifact I is to construct a checklist that provides practical guidance for the formulation of the situational evaluation criteria for DSM tools. The main design goal of Artifact II is to provide a conceptual method base for the evaluation of DSM tools in a situational context. The functional goal of Artifact I is to provide guidance for the activity of evaluation framework development in the instantiations of Artifact II. The functional goal of Artifact II is to provide method support for the engineering of situational evaluation methods for DSM tools. The goal for the discussion of the situational factors is to present the dimensions of IS and ISD tool evaluations, and to align them with the corresponding knowledge in SME. The goal for the discussion of the construction guidelines is to present the available approaches on a general level. The ultimate goal is to combine and present the designed artifacts as Meta-Method and provide evidence for methodical progress in comparison to the ISO 14102 standard. The method support provided by ISO 14102 for the evaluation of DSM tools in a situational context, or the lack thereof, was the initial indication of the need for this study.

1.3 Research Methodology

The research methodology consists of an adapted research framework and process, according to Design Science Research. Furthermore, the applied research methods are discussed.

1.3.1 Research Framework

The traditional behavioral research paradigm, i.e. developing and verifying theories that explain or predict the behavior of humans or organizations, has to a large extent characterized the previous research conducted in the field of Information Systems (IS) (Hevner, March, Park & Ram, 2004). Research methods such as survey (Kitchenman & Pfleeger, 2002a, 2002b, 2003; Pfleeger & Kitchenham, 2001) and case study (Robson, 2002; Yin 2003; Benbasat et al., 1987) are commonly known examples of the behavioral paradigm. However, the behavioral approaches are not optimal for the development of new and innovative artifacts that seek to extend the boundaries of the capabilities of humans and organizations (Hevner et al., 2004).

The Design Science Research (DSR) approach is a relatively recent specialization of the scientific method in the field of IS, in which the design and application of the artifacts in a practical context propagate knowledge and understanding of the problem domain and its solution (Hevner et al., 2004). Since the DSR paradigm addresses the design, application, evaluation, and theorizing of such artifacts, i.e. constructs, models, methods, and instantiations, it is selected as the high-level research framework for this study. Case study is however utilized in the empirical evaluation of the created artifacts, as it provides established means to evaluate the utility of the artifacts in a practical context.

Wieringa (2014) suggests that the practical context has elements such as people, values, desires, fears, goals, norms, and budgets, which must be investigated to fully understand the context. Furthermore, he states that the artifact itself does not solve any problem. Rather, it is the interaction between the artifact and the context that contributes to the solving of the problem (Wieringa, 2014). DSR incorporates a rigorous process for the design of the artifacts to solve relevant organizational problems, to evaluate the designs, and to communicate the results for appropriate audiences (Peffers, Tuunanen, Rothenberger & Chatterjee, 2007). The empirical and conceptual evaluation of the artifacts is emphasized in this thesis. Furthermore, a novel design theory (Gregor & Jones, 2007; Walls et al., 2004) is derived from the combination of the artifacts.

The widely accepted general guidelines for DSR by Hevner et al. (2004) are presented in TABLE 1.

TABLE 1 Design Science Research Guidelines (Hevner at al., 2004, p. 83)

Guideline	**Description**
Design as an Artifact	DSR must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Problem Relevance	The objective of DSR is to develop technology-based solutions to important and relevant business problems.
Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Research Contributions	Effective DSR must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methods.
Research Rigor	DSR relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Communication of Research	DSR must be presented effectively both to technology-oriented as well as management-oriented audiences.

The DSR framework by Hevner et al. (2004) is presented in FIGURE 1. DSR requires relevance from the application environment of the artifacts, i.e. the artifacts must address real business needs that originate from the interactions of people, organizations and technology. On the other hand, DSR requires rigor for the artifacts from the scientific knowledge base, i.e. the artifacts must have solid theoretical foundations and they must be designed using scientific methods. Moreover, the utility of the artifacts must be evaluated by studying their usage in the application environment with proven methods. The iterative DSR process should produce artifacts that provide potential solutions to real-world problems in the application domain as well as scientific contributions to the knowledge base. (Hevner et al., 2004.)

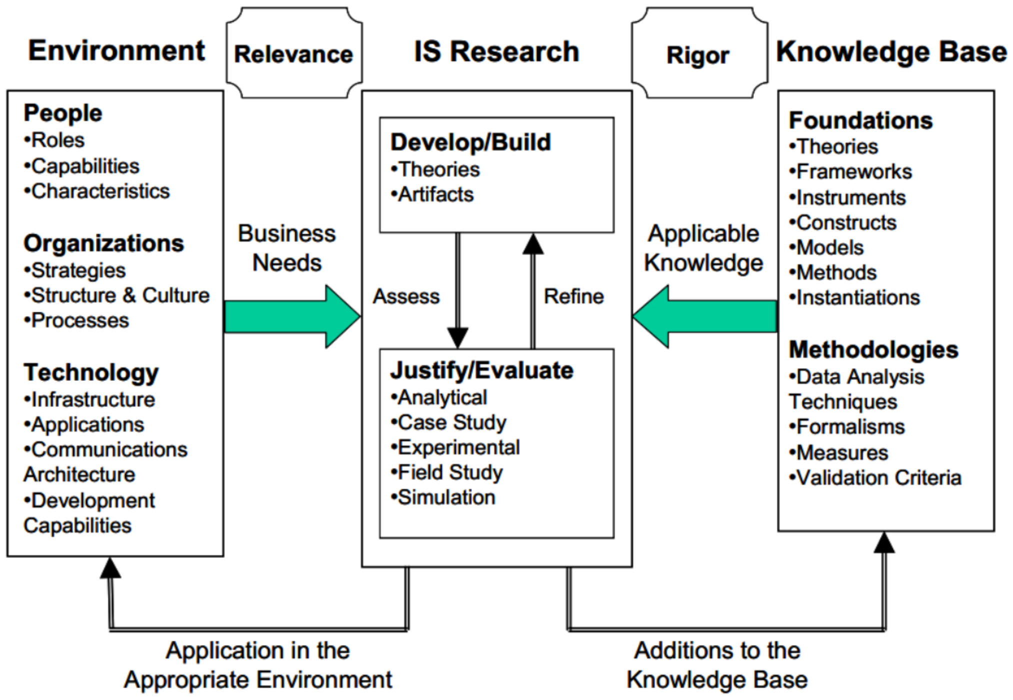


FIGURE 1 Design Science Research Framework (Hevner et al., 2004, p. 80)

The instantiation of the DSR framework in our study is described as follows. In *Environment*, People comprise roles such as researcher, method engineer, evaluator, project leader, and chief engineer. Organizations include a research lab located in University of Jyväskylä as well as a case study company that operates in the industry of professional mobile radio (PMR) networks and devices. The case study company provides the concrete business need for the evaluation of the state-of-the-art of DSM tools. The evaluation is required for the purpose of selecting the optimal tooling for the re-engineering effort of a software product line for PMR devices. University of Jyväskylä provides the research resources to conduct the evaluation. We are the researcher, who iteratively designs and evaluates the artifacts as well as derives the design theory presented in this thesis. Technologies such as ISD, DSM and CAME tools as well as communication and project management tools are utilized. In *IS Research*, artifacts such as Artifact I and Artifact II are designed, which are finally combined as Meta-Method. Meta-Method is empirically evaluated in the case study as well as conceptually evaluated by design theory componentization and the criteria of progress for IS design theories. In *Knowledge Base*, Foundations such as DSM, evaluation methods for ISD tools, and SME are utilized in the design of the artifacts. Also, Methodologies such as literature review, semantic analysis, and conceptual modeling are applied in the design process. The contributions of the study are finally added to Knowledge Base as publications, such as this thesis.

1.3.2 Research Process

The DSR process (Peffers et al., 2007) has six activities: problem identification and motivation, define objectives of a solution, design and development, demonstration, evaluation, and communication. *Problem identification and motivation* includes the definition of the research problem and justification of the value of the solution. *Define objectives of a solution* refers to the inference of the objectives of the solution using the problem definition and the knowledge of what is possible and feasible. *Design and development* represents the creation of the artifact. *Demonstration* refers to the use of the artifact to solve one or more instances of the problem. *Evaluation* includes the observation and measurement of how well the artifact supports a solution to the problem. *Communication* refers to the communication of the problem and its importance as well as to the artifact and its utility and novelty, the rigor of its design, and its effectiveness to the relevant audiences, such as the researchers and practitioners of the field. The DSR process doesn't impose a specific order for the activities, thus the process can be initiated from any of the following activities: problem identification and motivation, define objectives of a solution, design and development, and demonstration, and iterated over to address the issues related to the other activities. (Peffers et al., 2007.)

The DSR process adapted to this study is *motivated* by the case study company's business need for an evaluation of the state-of-the-art of DSM tools from the perspective of their specific business unit. The implemented DSR process is illustrated in FIGURE 2, in which the process and its relation to the research framework are presented. We stipulate the aforementioned business need in the terms of the DSR process as: the need for the demonstration of Meta-Method. From this delineation we derive the research problem presented in Section 1.2, in which we also define the *objectives* for the research. Thus, the *design and development* activities focus on the iterative design of the Artifact I and artifact II, from which Meta-Method is derived, considered from both conceptual and empirical points of view. In the *demonstration* activities, Meta-Method is instantiated by SME practices, producing situational methods that are enacted as the iterations of the evaluation effort for DSM tools in the context of the case company. Ultimately, the enacted evaluation methods produce reports for the decision-making of the case company. In the *evaluation*, the instantiations of Meta-Method are empirically evaluated in the case study. Furthermore, Meta-Method is conceptually evaluated through the comparison against ISO 14102, using the criteria of progress for IS design theories. Finally, the study is *communicated* via forums such as this thesis.

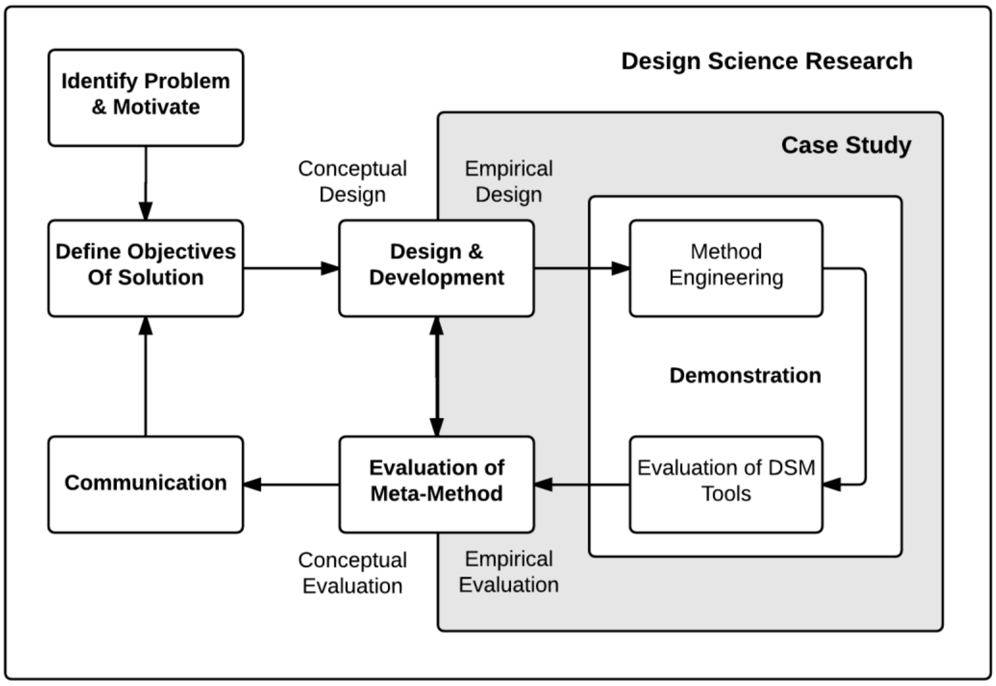


FIGURE 2 Research Process

Further discussions of the evaluation methodology, the case study, and the design theory development as well as the empirical and conceptual evaluation are presented in Chapter 6.

1.4 Structure of the Thesis

The thesis is organized into seven chapters. In Chapter 2, DSM is introduced and motivated as well as DSML development, model transformation, and DSM tools discussed. Chapter 3 discusses the evaluation and requirements of DSM tools, existing evaluation criteria, classification and analysis of the criteria as well as presents the design of Artifact I. Chapter 4 discusses the socio-technical dimensions of situational evaluation as well as the existing evaluation methods and their classification in the Activity dimension. Chapter 5 discusses the design of Artifact II, introducing the SME foundation as well as the core elements of Artifact II: a method base, situational factors, and construction guidelines. Chapter 6 discusses the evaluation methodology for Meta-Method as well as its application in the conceptual and empirical evaluation of Meta-Method. Finally, in Chapter 7 a summary, a conclusion, and limitations of the study are presented as well as future research is outlined.

2 DOMAIN-SPECIFIC MODELING

This chapter defines the basic concepts of domain-specific modeling (DSM), motivates the DSM approach, and discusses DSM languages, model transformation, the development of DSM languages, as well as DSM tools. Finally, a summary of the chapter is presented.

2.1 Basic Concepts

A *domain-specific language (DSL)* is "a programming language or an executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain" (van Deursen et al., 2000). Sánches-Ruiz et al. (2006) define *domain-specific modeling (DSM)* as the process of building a model for a specific domain with a graphical DSL, which in this thesis, is called a domain-specific modeling language (DSML). A DSML is defined within a metamodel, which is a model of the DSML (Favre, 2005). A *metamodel* is "a model of the conceptual foundation of a language, consisting of a set of basic concepts, and a set of rules determining the set of possible models denotable in that language" (Falkenberg et al., 1996, p. 58). A metamodel is an output artifact of a process of *metamodeling*, often considered synonymous with building a DSML (Atkinson & Kühne, 2003). A *meta-metamodel* is the metamodel of a metamodel, i.e. it describes the concepts that are available for metamodeling (Stahl & Völter, 2006, p. 57). A *domain* is "an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area" (Booch et al., 1998). A *model* is "a simplified, stylized representation of system, abstracting the essence of the system's problem studied" (Wijers, 1991, p. 6). A model also helps or enables understanding, communication, analysis, design and/or implementation of something to which the model refers to (Leppänen, 2005, p. 57). *Domain modeling* is the process of identifying, documenting, and specifying the objects and their relationships that are relevant in the context of a given problem (Sánchez-Ruíz et al., 2006). DSM conforms to *model-driven development (MDD)* paradigm. MDD focuses on models in software development, rather than computer programs (Kent, 2002; Selic, 2003). MDD is also referred to as *Model-Driven Engineering (MDE)* (Schmidt, 2006).

2.2 Motivation for DSM Approach

Ever since the introduction of computers into society, scientists and practitioners have been constantly seeking ways to improve productivity of software engineering by reducing complexity and increasing abstraction (Saraiva & da Silva, 2008). A major leap was the transition to third generation languages (3GL), like C or Java, from Assembler, which resulted in drastic, even 450 % productivity gains (Jones, 2006).

DSM aims to provide similar benefits: "DSM raises the level of abstraction beyond current programming languages by specifying the solution directly using problem domain concepts" (Kelly & Tolvanen, 2008, p. 3). Application of DSMLs in software development eliminates the need for mapping the problem domain concepts to solution domain concepts. Industrial experiences consistently report DSM providing 5 to 10 times higher productivity rates than other current development approaches (Kärnä et al., 2009; Kelly, 2007). FIGURE 3 illustrates the approaches on how the abstraction gap between an idea in domain terms and its implementation has been bridged in software engineering (Kelly & Tolvanen, 2008, p. 16).

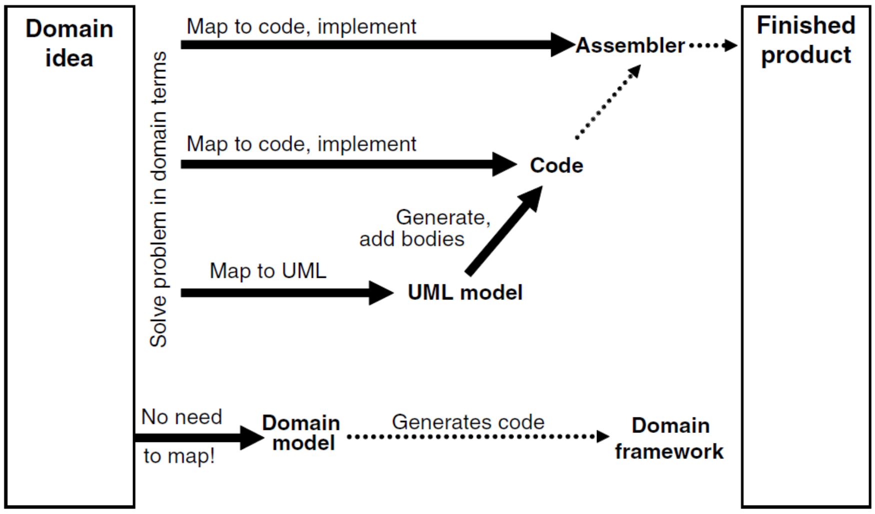


FIGURE 3 Bridging the Abstraction Gap (Kelly & Tolvanen, 2008, p. 16)

Models are utilized e.g. in designing systems, understanding them better, specifying required functionalities, creating documentation, and as universal teaching and learning tools (Ludewig, 2003). Commonly in software engineering the specification models that form the base of application code end up obsolete as customer requirements change and evolve. This is simply due to the fact that the cost of maintaining the same up-to-date information manually in two places is too high. In DSM this changes entirely since the models are the primary development artifacts. Executable code can be generated from the models. DSMLs utilize concepts and rules from the problem domain, as opposed to general modeling languages, such as Unified Modeling Language (UML), which was designed for describing object-oriented (OO) software constructs in the solution domain (Booch et al., 1998, p. 20). In FIGURE 4 different approaches to software development and their code-model alignment are presented. (Kelly & Tolvanen, 2008, p. 16)

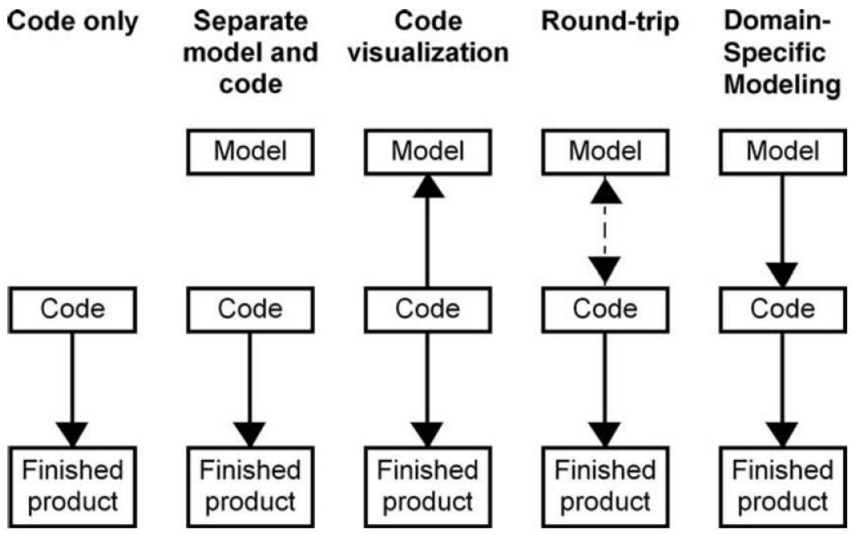


FIGURE 4 Aligning Code and Models (Kelly & Tolvanen, 2008, p. 5)

The code only approach represents programming without design specifications, which works well on small scale development tasks. The second approach is currently the most utilized, in which the software systems are designed separately from the code with general-purpose modeling languages, such as UML, and in the programming phase the mappings of model concepts to coding concepts are made manually by developers. The code visualization approach implements reverse engineering to derive model concepts from the finished code, e.g. for documentation generation purposes. Reverse engineering is the process of comprehending software and producing a model of it at a high abstraction level, suitable for documentation, maintenance, or re-engineering (Rugaber & Stirewalt, 2004). Round-trip approach utilizes engineering and reverse engineering to keep convergent information up-to-date both in models and code. In theory the development can then be executed in either media, but in practice their limitations in expressivity for the generation of either one restricts the functionality to class skeleton generation (Antkiewicz, 2006). (Kelly & Tolvanen, 2008, p. 5)

The goal of DSM is that application code is entirely generated from models constructed with customized DSMLs, employing the concepts and rules of a specific domain. This enables raise of abstraction and hides unnecessary complexity. Model-to-code transformations are automated via customized code generators, analogous to compilers translating code e.g. from C++ to Assembler. Generated code is complete and executable within a domain framework of a given application environment. A *domain framework* consists of everything below the code generator: hardware, operating system, programming languages and software tools, libraries and additional components or code on top of these, split into domain-specific and platform parts. Thus, a domain framework provides an interface between the generated code and the underlying target environment (Kelly & Tolvanen, 2008, p. 86). (Kelly & Tolvanen, 2008, p. 15)

Domain-specific languages and tools developed for a particular task will always perform better than general-purpose ones. Therefore DSM should be considered whenever applicable. DSM is suited where the domain and application are well known, thus it is not ideal for unique projects. DSM is optimal for software product families having development tasks of repeatable nature and an established solution history. As in code-driven development, reuse of libraries, components and services increases productivity, the application of DSM requires similar principles, as it offers a way to find a balance between writing code manually and generating it. (Kelly & Tolvanen, 2008, p. 18.)

2.3 Domain-Specific Modeling Languages

The focus of a DSML needs to be in a narrow area in order to enable transformations that produce executable artifacts, requiring minimal to no manual patching. The components of a DSML are Concrete Syntax, Abstract Syntax, and Semantics. Concrete Syntax is mapped to Abstract Syntax, and Abstract Syntax is mapped to Semantics. The structure enables well-formed models to be created. The components and their relationships are illustrated in FIGURE 5. (Cho, 2013, p. 23.)

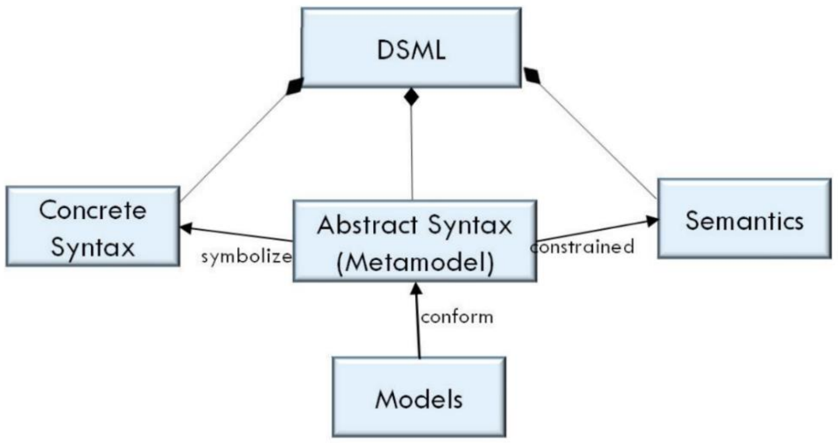


FIGURE 5 DSML Components and their Relationships (Cho & Gray, 2011, p. 2)

Abstract Syntax is the description of the concepts of a DSML, the structural relationships between the concepts, and the constraints that define how the language elements can be combined to describe specific domains. Abstract Syntax is the metamodel of a DSML. *Concrete Syntax* defines the visual notation of a DSML, utilized in DSML application. Concrete Syntax can be e.g. textual, graphical, mixed or matrix representation. Concrete Syntax elements must be mapped via rules to Abstract Syntax elements. Typically, Abstract Syntax elements can be mapped to one or more Concrete Syntaxes, for different usage purposes, e.g. graphical model notation for human use and XMI specification for model exchange between tools. *Semantics* are typically utilized in the specification of structural and behavioral properties of Abstract Syntax elements, and in the governance of the syntax and semantics of Concrete Syntax and the values of properties. (Cho et al., 2012; Cho, 2013, p. 23)

FIGURE 6 demonstrates the Abstract Syntax (metamodel) for a simple Finite State Machine (FSM) DSML, applicable to modeling the states of a system and the transitions between the states. FSM DSML includes two elements, State and Transition, which are connected by Incoming and Outgoing relationships. The State and Transition elements are mapped to respective Concrete Syntax elements, demonstrated in FIGURE 7. (Cho, 2013, p. 24)

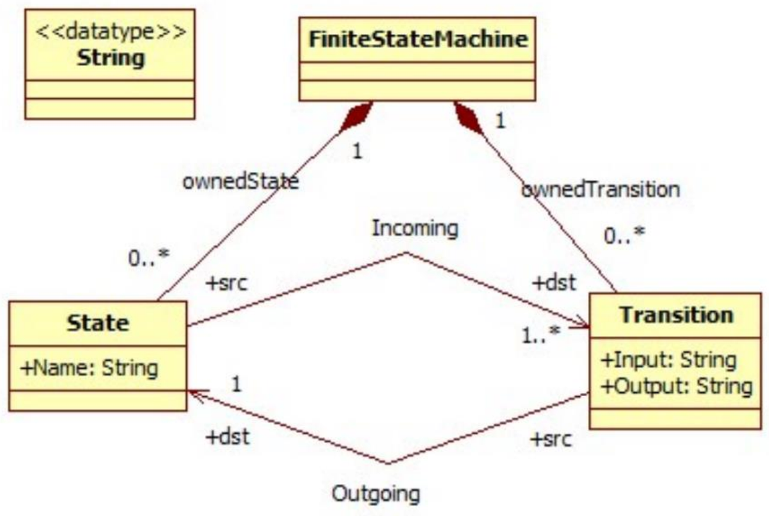


FIGURE 6 Abstract Syntax of a Finite State Machine DSML (Cho, 2013, p. 24)

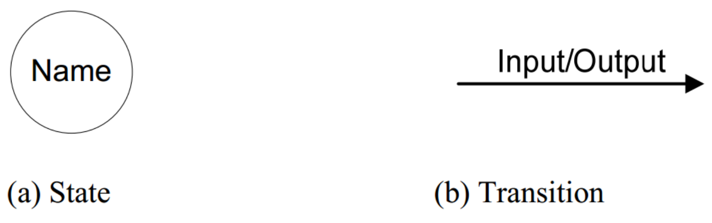


FIGURE 7 Concrete Syntax of a Finite State Machine DSML (Cho, 2013, p. 24)

After the mappings are specified, semantics can be utilized in fine tuning e.g. the behavior of the elements and the values the element properties can have. Then, a modeling tool providing the FSM DSML can be generated and applied in the modeling of FSMs.

2.4 Model Transformation

There are two common types of model transformation: Model-to-Model (M2M), and Model-to-Text (M2T) (Czarnecki & Helsen, 2003). M2M transforms models into other types of models and M2T transforms models into textual artifacts, such as application code. FIGURE 8 illustrates the basic concepts of model transformation in terms of M2M. The source models, conformant to the source metamodel are transformed into target models, conformant to the target metamodel, utilizing transformation definitions, executed on a transformation engine. In the case of M2T, which is the primary type of transformation discussed in this thesis, the target is a textual representation of the source model. (Czarnecki & Helsen, 2006)

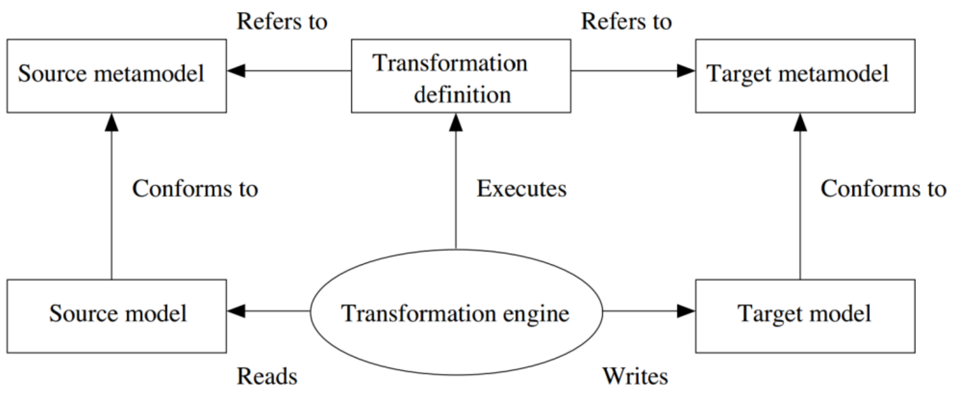


FIGURE 8 Concepts of Model Transformation (Czarnecki & Helsen, 2006, p. 3)

In DSML application e.g. novice developers or non-technical personnel can utilize the DSML to produce models, transform them to code, and execute the code as is, or within a domain framework. For example, an FSM model created using the DSML described in the previous section, can be transformed into FSM code for a given system, using M2T. The quality of the generated artifacts is consistent and corresponds to the capabilities of the developers that defined the transformation. Template-based transformations are widely used in M2T (Czarnecki & Helsen, 2006).

Typically, M2T templates consist of two types of code. There is code for accessing and selecting model data by traversing the model structure specified in the metamodel. There is also code for expanding and wrapping the selected model data into chunks of strings, ultimately forming the structure of the application code generated. There are multiple ways of implementing the templates, such as tree-based intermediate representation and DSLs for M2T. (Hoisl et al., 2013)

Hoisl et al. (2013) propose that the templates are first-class modeling elements and suggest an abstraction model from the implementation details. They consider the templates as instances of a conceptual M2T template metamodel, defined in the MOFM2T specification (OMG, 2008a). This promotes the portability of the template approach to the modern M2T languages, such as Eclipse Xpand, JET, and Acceleo. FIGURE 9 illustrates the approach by utilizing the MOF four-layer architecture, discussed in detail in Chapter 2.6. (Hoisl et al., 2013)

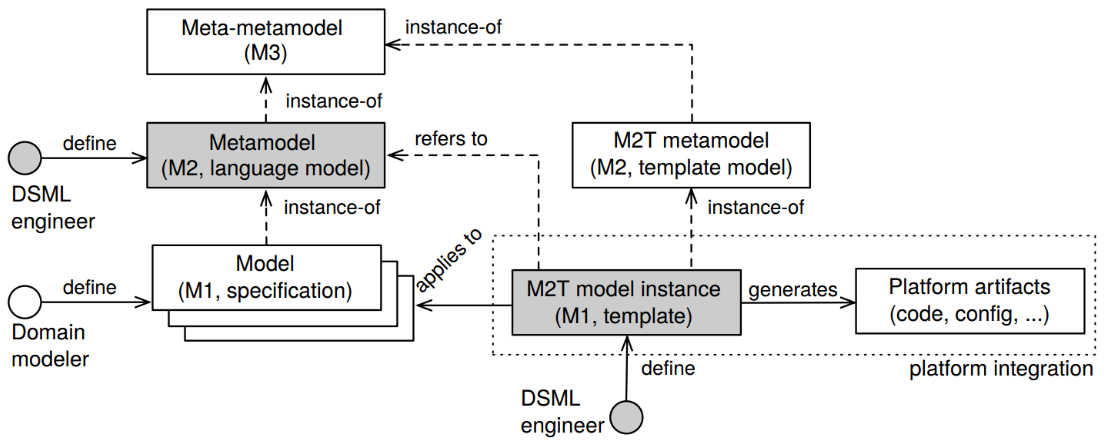


FIGURE 9 Model-to-Text Template Model Transformation (Czarnecki & Helsen, 2006, p. 3)

A DSML engineer defines a domain-specific template in level M1, using concepts and rules defined in level M2 metamodel, which in turn is defined in level M3, the MOFM2T specification. Then, a domain modeler utilizes a DSML to generate models in level M1 and applies the M1 template to generate artifacts of level M0, such as application code. (Hoisl et al., 2013)

2.5 Domain-Specific Modeling Language Development Process

The development of a DSML can be carried out by utilizing any of the available software development methods, like the waterfall model or agile methods. The DSML development is distinct in that it has the three interrelated components of Concrete Syntax, Abstract Syntax, and Semantics. The development of the components has to be considered independently as well as the mapping of them together in a unified way. (Cho, 2013, p. 28.)

Typically, the development requires the collaboration of two types of experts: domain experts and DSML engineers. A domain expert has profound knowledge and expertise within a given domain, and has the capability to describe the DSML requirements as well as validate the DSML for a release. A DSML engineer builds the language by analyzing the requirements, developing the components, mapping them together, and performing tests. The metamodeling language, mapping mechanisms, and DSML editor generation facilities provided by a DSM tool are utilized in the development process. (Cho, 2013, p. 30)

A demonstration-based DSML development process is illustrated in FIGURE 10. The process starts with the capturing of the requirements. The goals of the requirements engineering are to identify stakeholders of the domain, define the domain scope, and to identify the notation typical to the domain. The concrete syntax is often specified next, as it promotes communication between the stakeholders via use of symbols and concepts of the domain, thus helping to explore the specific problem domain. As the concepts and rules of the domain unfold, the logic of a DSML is captured into the abstract syntax design, and mapped to the concrete syntax. After the syntax is designed, semantics should be specified and associated to the abstract syntax. In the specification of the semantics, three activities are involved: understanding of the designed syntax of the DSML, choosing a semantic domain, which is the formalism used to define the DSML, and mapping from the syntax to semantic domain. Finally, the language is verified by testing, validated by the domain expert, and released. On a side note, the demonstration-based approach promotes the definition of the concrete syntax before the abstract syntax, which is contrary to the traditional model. (Cho, 2013.)

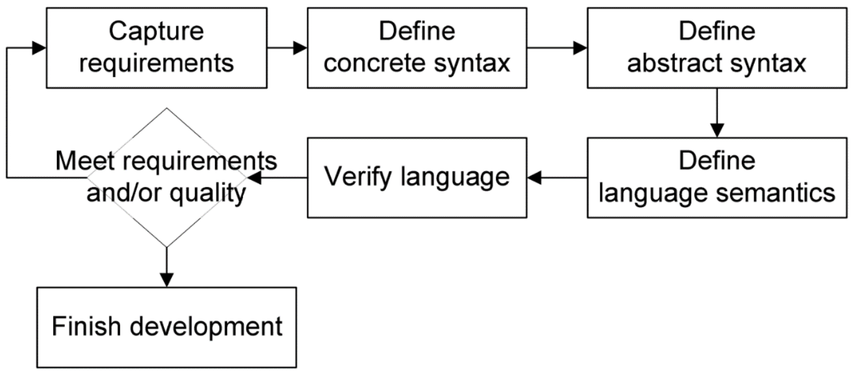


FIGURE 10 Demonstration-Based DSML Development Process (Cho et al., 2012, p. 1)

In order to utilize the DSML in the generation of artifacts such as application code, executable within a domain framework, a model transformation definition is specified. The DSML engineer defines the transformation using a language supported by a transformation engine, provided by a DSM tool. Models defined by domain modelers (the DSML users) are transformed into code utilizing M2T transformation artifacts called code generators. The code generator is specified by the DSML engineer by analyzing the domain requirements and existing codebase from the problem domain. The roles of the DSML engineer and the domain modeler are illustrated in FIGURE 9 in template-based transformation. DSM tools may provide a fixed mechanism for transformations or the functionality can be added as a module or plugin. (Hoisl et al., 2013)

2.6 Domain-Specific Modeling Tools

In model-driven development the importance of modeling tools is emphasized since the models are the main development artifacts, not just throwaway sketches of systems (Favre, 2004; Seidewitz, 2003). Modeling tools allow e.g. creating, checking, verifying, reusing, integrating and sharing of design specifications. In traditional Computer-Aided Software Engineering (CASE) tools the modeling languages are hard-coded into the tools as fixed metamodels and developers are restricted to using them (Kelly & Tolvanen, 2008, p. 59). *A CASE tool* is a software development tool that aids in software engineering activities, including, but not limited to, requirements analysis and tracing, software design, code production, testing, document generation, quality assurance, configuration management, and project management (IEEE, 1995). In the context of this thesis, CASE tools are referred to as *ISD tools*, denoting a general-purpose tool that supports ISD activities. DSM tools can be considered as a specific type of ISD tools, also called meta-CASE tools, which enable the design and generation of customized ISD tools, i.e. modeling tools that implement DSMLs.

A *DSM tool* provides facilities for DSML specification and application as well as model transformation (Kirchner and Jung, 2007). A *DSML specification facility* provides tool support for the specification of the components of a DSML. The specification of a DSML metamodel is governed by the meta-metamodel provided by the DSM tool (Stah & Völter, 2006, p. 57). There is a number of DSM tool-specific meta-metamodels available that provide divergent metamodeling concepts, such as OMG's Meta Object Facility (MOF) (OMG, 2006), Ecore, the implementation of essential subset of MOF, EMOF in Eclipse Modeling Tools (Steinberg et al., 2009), and Graph-Object-Property-Port-Role-Relationship (GOPPRR) in MetaEdit+ (Kelly et al., 2013; Tolvanen et al., 2007). The heterogeneity of the meta-metamodels leads to issues such as the selection of a meta-metamodel and interoperability between DSM tools (Kern et al., 2011). A *DSML application facility* refers to an ISD tool that is generated by a DSM tool, implementing the DSML specification. A *model transformation facility* provides the means to transform the models specified in the generated ISD tools to other artifacts, such as program code. (Kirchner and Jung, 2007)

Atkinson and Kühne (2005) propose three main architectures for modeling tools: four-layer architecture, two-level cascading architecture, and orthogonal classification architecture. In practice, the four-layer architecture is extensively used e.g. in Eclipse Modeling Tools, and considered a prominent architecture for DSM tool design (Atkinson & Kühne, 2005; Karagiannis & Kühn, 2002). The two-level cascading architecture is employed by the commonly used DSM tools such as MetaEdit+ and by the Software Factories approach. The orthogonal classification architecture is based on level compaction and uses the library metaphor, e.g. in ConceptBase. (Atkinson & Kühne, 2005)

A typical example of the four-layer architecture is MOF, which is presented in FIGURE 11, along with its alignment with DSM tools and ISD tools. MOF is a language adapted to the domain of OO approach to modeling (Atkinson & Kühne, 2005), while UML is a language adapted to the domain of OO programming languages (Saraiva & da Silva, 2008). The four-layer architecture employs four distinct logical modeling layers M3, M2, M1, and M0. M3 is the MOF meta-metamodel layer, which defines UML metamodel in M2, which defines models in M1 that define the application instances of those models in M0. A meta-metamodel is written in a meta-metamodeling language, a metamodel is written in a metamodeling language, and a model is written in a modeling language. A DSM tool has a fixed meta-metamodel, which defines a metamodeling language used in the development of a DSML. Thus, a metamodeling language is used in the construction of a metamodel of a DSML. In a traditional ISD tool the metamodel is fixed, so only the modeling language, such as UML, defined by the metamodel, is provided.

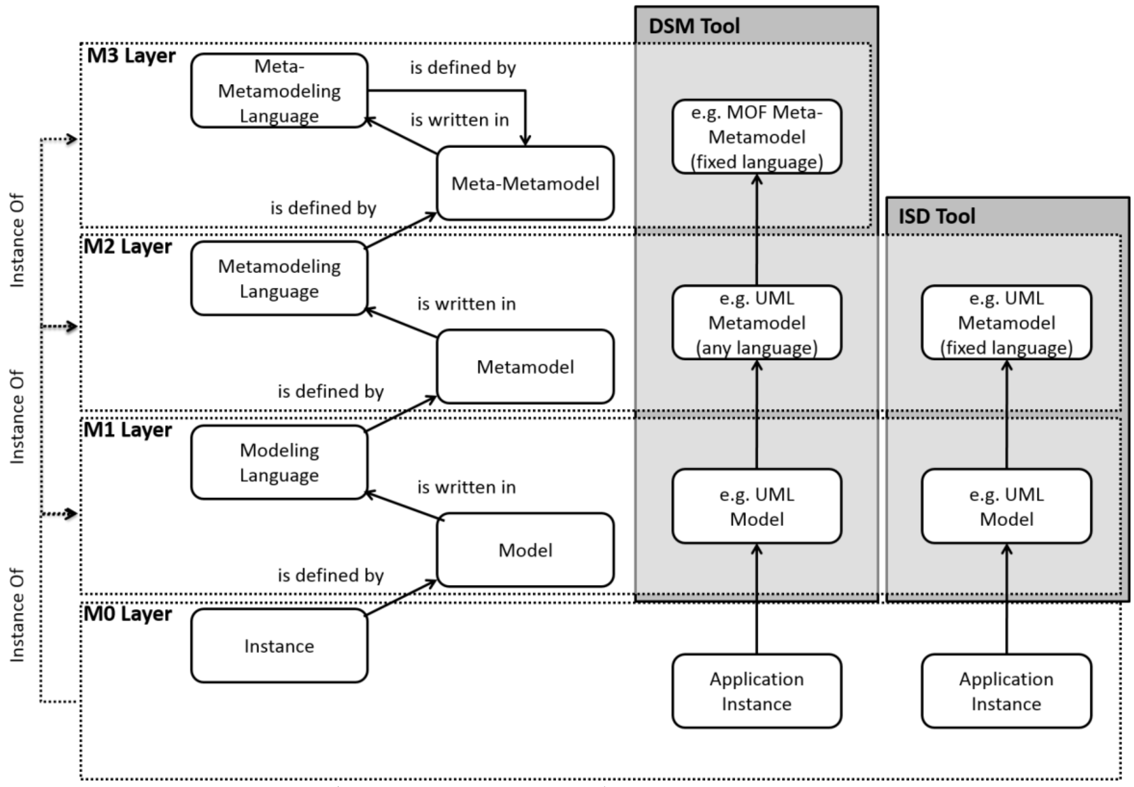


FIGURE 11 Four-Layer Architecture in DSM Tools

Essentially, DSM tools need to have the facilities to construct modeling languages and transformations to enable increased automation. They also need to be able to provide adequate tool support, usability, and reliability for an entire DSM solution life-cycle. A DSM solution is a production application utilized in a domain framework, generated from a model, created with a DSML. It is important that a DSML can be quickly developed and easily maintained, since the rationale for using this approach is the productivity increase via automation. If the cost of a DSM solution is greater than the cost of a manually programmed solution, there's no point in using DSM. The value of having automation in use as early as possible is salient. Optimally the development of a DSM solution would only require the construction of a DSML and a transformation, along with a domain framework to support the generated code, and the tool should provide the rest. During the evolution of a DSM solution the safety of tool customization becomes crucial. All modifications to the language should propagate to all specifications without deleting or corrupting them. Integration with compilers, debuggers and testing tools is also needed. In summation, DSM tools should guide and support developers during the construction and maintenance of DSM solutions. (Kelly & Tolvanen, 2008, p. 61)

2.7 Summary

This chapter defined the basic concepts of domain-specific modeling (DSM), motivated the DSM approach, and discussed DSM languages, model transformation, the development of DSM languages, as well as DSM tools.

Domain-Specific Modeling was delineated as an approach to software engineering in which models are the primary development artifacts, typically applied in narrow and well-established domains, for the purpose of increasing productivity by automation. DSM raises the level of abstraction beyond current programming languages by specifying the solution directly using problem domain concepts. In the DSML development process, the concepts, rules, and semantics of application domains are captured in the specifications of DSMLs that are utilized in the production of models, from which software and other artifacts are generated via model transformations.

DSM tools were defined as tools which, in addition to the functionality of conventional ISD tools, provide facilities for DSML specification and application as well as model transformation. In addition, a robust architecture, adequate user support, usability, reliability, and development life-cycle support are essential characteristics. DSM tools provide dynamic metamodels for modeling tool specification, as opposed to ISD tools, which provide fixed metamodels for modeling languages such as UML. In summary, DSM tools should guide and support developers in the construction and maintenance of DSM solutions.

3 EVALUATION CRITERIA FOR DSM TOOLS

The topic of this chapter is the evaluation of DSM tools. It is considered from four perspectives. First, the evaluation of DSM tools is discussed on a general level. Second, four sets of requirements for DSM tools are presented based on four previous studies. Third, evaluation criteria for DSM tools, presented in eight previous studies, are outlined. Fourth, a comprehensive classification of evaluation criteria for DSM tools, derived from existing criteria, is introduced. The classified evaluation criteria are adapted into a checklist with unified data types, ranges, and examples of criteria values, effectively forming Artifact I.

3.1 Evaluation of DSM Tools

Evaluation is the process of determining the merit, worth, or significance of things (Scriven, 2003). Evaluation is practiced when quality, value, and/or importance of things are assessed (Scriven, 2001). Evaluations are conducted by evaluators using evaluation criteria against a set of standards. *Evaluator* is the practitioner of an evaluative study. *Evaluation criteria* are a selected subset of properties of things, governed by stakeholder values. A criterion may consist of one or more metrics that define the value of the criterion. The criteria may be weighted and/or prioritized, as well as the standards set, according to the requirements in question. Evaluation can be employed by internal and/or external evaluators before, during, and/or after the lifecycle of a thing, in generic or context-specific settings, in novel or supplemental capacity. (Scriven, 2003.)

Evaluations of software tools are typically conducted for the justification of acquisition or for research purposes (Lundell & Lings, 2004b). DSM tools are often evaluated by the industry for the purpose of investigating the opportunities to implement DSM in software production, or to upgrade the current DSM tools in use. DSM tools are also studied by researchers with the aim of producing objective knowledge by e.g. comparing the tool features and capabilities in a case study or lab setting. Furthermore, Kelly (2013, p. 1) identifies the following types of approaches for the evaluation of DSM tools:

- comparing DSM tools as different ways of producing an ISD tool for the same DSML
- comparing the effort to update the resulting ISD tool when the DSM tool, problem or solution domain evolves
- comparing the productivity of the resulting ISD tool and transformation against hand-writing the same code
- comparing the productivity of different DSMLs made for the same domain with different DSM tools
- comparing the performance of the resulting ISD tool: how long the user has to wait for the tool to open a model, generate code, show model changes etc.

This chapter discusses the evaluation criteria utilized in scientific comparative studies. The compiled criteria findings may be utilized as a guideline in the criteria formulation of future studies and also as a reference for industrial evaluations.

3.2 Requirements for DSM Tools

In this section a set of definitive features and requirements for DSM tools by Lukman and Mernik (2008), Achilleos et al. (2007), Nytun et al. (2006), and Kelly and Tolvanen (2008) are presented. The features and requirements applying to DSM tools are collected and unified to eliminate repetition. The basic features of DSM tools are the facilities for the specification and application of DSMLs as well as the model transformation (Kirchner & Jung, 2007). The requirements presented in the following provide aspects of the facilities in more detailed manner, building the foundation for the analysis of Artifac I.

Lukman and Mernik (2008) propose a set of minimal features for MDE tools: Modeling Environment and Artifacts Generator. Additionally, they suggest additional useful features, which could increase developer adaptation of MDE. The features are presented in TABLE 2.

TABLE 2 Features of MDD Tools (Lukman & Mernik, 2008)

Feature	Description
Modeling Environment	Enables the creation and editing of visual models. This environment must also include a way of defining and enforcing constraints on the build models.
Artifacts Generator	A model-to-code transformation engine, which enables the generation of source code, documentation and other development artifacts based on the given models.
Model Debugger	The development of today's complex and extensive software is hardly imaginable without debugging capabilities. Debugging capabilities should also be available on the modeling level.
Model Validation	Models are validated with the constraints that are present in the domain they belong to.
Model-to-Model Transformation Engines	To enable advanced development tasks on the available models a mode-to-model transformation engine is needed. Such tasks are e.g. model refactoring and exploration of design alternatives.
Test Suite	Enables testing on the modeling level.
Model Analysis Tools	Enable analysis of the constructed models in various ways e.g. assessing the quality of models (via model metrics).
Model Simulators	In some domains, e.g. embedded software, code execution on the actual platform is not rational or possible. Therefore simulation capabilities on the modeling level are much desired.

Achilleos et al. (2007) propose additional requirements for MDD tools, presented in TABLE 3.

TABLE 3 Requirements for MDD Tools (Achilleos et al., 2007)

Requirement	Description
Abstract Syntax	Any DSML shall be specified as a M2 meta-model using a semantic meta-metamodeling language, such as MOF. An effective DSM tool must ensure completeness of the new modeling language through its meta-metamodeling language.
Concrete Syntax	A DSML shall additionally specify a graphical notation, to allow the concrete representation of its abstract concepts. This will enable better understanding of the language and will make its use easier in developing models.
Metamodel constraints	Precision in the DSML semantics shall be provided by the specification of constraints onto the M2 metamodel (abstract syntax) to ensure correctness of the language.
Modeling tools generation	One to one mapping must be enabled between the DSML abstract concepts and their corresponding concrete representation, which shall lead to the generation of a modeling tool. The tool will be used for the specification and management of M1 models.
Text-based generation	A DSM tool shall generate text-based output from M1 models. This can lead to code generation in a programming language, such as Java, or a markup language, such as XML.
Accelerated adoption	Generated tools should be easy to use for the modelers.

Nytun et al. (2006) suggest high-level requirements for metamodeling tools, presented in TABLE 4.

TABLE 4 Requirements for Metamodeling Tools (Nytun et al., 2006)

Requirement	Description
Generativeness	When discussing tools that produce modeling tools, the most important requirement is that they are able to automatically produce the tool. This refers to the mapping from the metamodel to the tool code.
High-Level Description	The descriptions are more easily handled when they are given in a high-level notation. This means that a tool should provide high-level notations for the different modeling language aspects.
Completeness	The coverage of aspects related to metamodel structure, constraints, representation and behavior. A good metatool will allow the expression of all important aspects of a modeling language.
Conformance to Standards	The tools are produced automatically from the corresponding standards documents. For this to be possible the standards documents have to be given in a formal way.

Kelly and Tolvanen (2008) propose requirements for DSM tools, addressing more specific, functional aspects of metamodeling and modeling (Kelly & Tolvanen, 2008, p. 365). The requirements are presented in TABLE 5.

TABLE 5 Requirements for DSM Tools (Kelly & Tolvanen, 2008, p. 365)

Metamodeling Requirements
Specify the object and relationship types declaratively
Specify declaratively a list of properties for each object or relationship type, with support for at least string and Boolean property data types
Specify basic rules for how objects can be connected by relationships
Specify symbols for types, whether graphically, declaratively or in code
Ability for a generator to access the models
From specifications defined by metamodeler, create a modeling tool
Modeling Requirements
Store and retrieve a model from disk
Create new instances in models by choosing a type and filling in properties
Link objects via relationships
Lay out the objects and relationships, either by dragging or automatic layout
Edit properties of existing object relationships
Delete objects and relationships

The definitive features and requirements for DSM tools presented in this section may be used as a reference in requirements engineering of real-world DSM tool evaluations. The requirements should be mapped to the corresponding evaluation criteria, some of which are reviewed in the next section.

3.3 Evaluation Criteria for DSM Tools in Previous Studies

There are some studies that report on evaluations of DSM tools against defined evaluation criteria. Saraiva and da Silva (2008) evaluated a set of DSM tools specifically from the metamodeling perspective, focusing on architectural and practical aspects. They included both theoretical and practical evaluation criteria. Nevertheless, they pointed out that the criteria were subjective and to establish a higher degree of consensus on the criteria, more work was required in the future. The tools evaluated were Enterprise Architect, MetaSketch, MetaEdit+, and Microsoft DSL Tools.

Vasiljević et al. (2013) evaluated five DSM tools by analyzing their advantages and disadvantages from the viewpoint of the user and the applicability to both academic and industrial contexts. They didn't discuss the background of the criteria formulation or analyze the criteria application, but on the general note they kept the criteria relative simple and didn't go into fine grain details of DSM tools. The evaluated tools were Visual Studio DSL Tools, Poseidon for DSL, Spray, Magic Draw Standard, and AToM^3. (Vasiljević et al., 2013)

The evaluation criteria utilized by Saraiva and da Silva (2008) and Vasiljević et al. (2013) are presented in the TABLE 6.

TABLE 6 Criteria by Saraiva & da Silva (2008) and Vasiljević et al. (2013)

Saraiva & da Silva (2008)	**Vasiljević et al. (2013)**
Supported Standard Exchange Formats	Documentation
• Metamodels	Storage format
• Models	Operating System
Model Transformation Framework	License
Usage of the Level Compaction Technique	Price
Usage of the Language Metaphor	Version
Usage of the Library Metaphor	Deployment Model
Number of Levels the User Can Manipulate	Meta-metamodel
Support for Specifying Metamodel Syntax	Concrete Syntax Representation
• Supports Specification	Abstract to Concrete Syntax Mapping
• Languages Used	Multiple Concrete Syntax
Support for Specifying Metamodel Semantics	Constraint Language
• Supports Specification	Model to Model
• Languages Used	Model to Text
The Size of The Hard-Coded Meta-metamodel	Extensibility
	Stability

Pelechano et al. (2006) compared Eclipse Modeling Plugins and Microsoft DSL Tools in a controlled experiment in which 48 undergraduate computer science students, divided into two groups, developed a demonstrative DSML using the tools. Afterwards the test subjects answered a survey with multiple questions about the task. The survey feedback data was utilized in the statistical comparison of the tools. The survey questions were abstracted as evaluation criteria.

Langlois et al. (2007) proposed a feature model for DSL tools, including both textual and graphical tools, and conducted a DSM tool evaluation between three tools. The feature model was constructed for the purpose of formalizing DSL and DSL tool variants, and the evaluation criteria were derived from the model. The tools evaluated were Eclipse GMF, Microsoft DSL Tools, and Xactium XMF-Mosaic.

The evaluation criteria by Pelechano et al. (2006) and Langlois et al. (2007) are presented in TABLE 7.

TABLE 7 Criteria by Pelechano et al. (2006) and Langlois et al. (2007)

Pelechano et al. (2006)	**Langlois et al. (2007)**
Documentation Availability	DSL Metamodel
Metamodeling Language Understandability	Abstract/Concrete Syntax Representation
Metamodeling Language Expressivity	Concrete Syntax Style
Language (Metamodel) Designer Usability	Abstract to Concrete Syntax Mapping
Graphical Designer Usability	Problem to Solution Mapping Expression
Quality of the Resulting Graphical Modeler	Existence of Viewpoints
Graphical Designer Completeness	Destructive/Incremental Asset Update
Extensibility of the Graphical Designer	Asset Integration Support
Comparing Generated Editors	Model/Text Techniques
Maturity and Robustness	Internal/External Execution Environment
Complexity in Defining the Code Generator	Abstraction: Intrusive/Seamless
Implementing the Code Generator	Providing Adaptive Tool Assistance
Utility of the Employed Tools	Providing Step/Workflow Process Guidance
Industrial Application	DSL Checking
Fidelity to the Tool	Usability
	Portability
	Interoperability (Part of Functionality)
	Maintainability

Amyot et al. (2006) evaluated a set of DSM tools by performing a DSML development task, with the criteria derived from the practical point of view. In the evaluation a particular emphasis was put on the generation of graphical editors, which was reflected on the criteria formulation. The DSM tools evaluated were Generic Modeling Environment (GME), Xactium XMF-Mosaic, and Eclipse EMF with GEF. Also the evaluation of the UML profiling capabilities of Telelogic Tau G2 and of Rational Software Architect (RSA) were included.

Sivonen (2008) evaluated DSM tools for the purpose of selecting a tool for a DSML development study. The task had the objective of developing a DSML for creating repository-based Eclipse plugins, although tools from other vendors were evaluated as well. One could argue the nature of the task might have biased the criteria formulation towards Eclipse. The tools evaluated were Microsoft DSL Tools, GME, Eclipse GEMS, and MetaEdit+.

The evaluation criteria utilized by Amyot et al. (2006) and Sivonen (2008) are presented in TABLE 8.

TABLE 8 Criteria by Amyot et al. (2006) and Sivonen (2008)

Amyot et al. (2006)	**Sivonen (2008)**
Graphical Completeness	Tool Provider
Editor Usability	Supported Platforms
Effort	License
Language Evolution	Documentation and Support
Integration with Other Languages	• User's Guide for the Tool
Analysis Capabilities	• Tutorials
	• Instructions for the Code Generator Definition
	• E-Mail Support
	Metamodeling Language
	Constraint Definition Possibilities
	Code Generation Possibilities
	• Generator Definition Language
	• Generator Output Language

De Smedt (2011) compared three DSM tools in a case study, which included a DSML development task. The study comprises of a practical comparison and a technical comparison, in which the undertaking of the task was evaluated. The tools evaluated in the study were AToM^3, MetaEdit+, and Poseidon for DSL.

El Kouhen et al. (2012) propose criteria for the evaluation of the adaptability of a DSM tool by observing how well they can be used to customize graphical editors for a sample DSML in a case study. The evaluation criteria are specified in detail, including various quantitative metrics that comprise the singular criteria. The tools evaluated in the study were IBM Rational Software Architect (RSA), GME, MetaEdit+, Obeo Designer, and Eclipse GMF.

The evaluation criteria by De Smedt (2011) and El Kouhen et al. (2012) are presented in TABLE 9.

TABLE 9 Criteria by de Smedt (2011) and El Kouhen et al. (2012)

De Smedt (2011)	**El Kouhen et al. (2012)**
Speed of Development	Customization Level
Documentation	Graphical Expressiveness
Repository	Graphical Completeness
Platform	Tool Openness
Price	• Tool Building Approaches
Integration	• Extensibility
Transformation Tool	• Reusability
API	• Maintainability
Abstract Syntax	Tool Usability
Concrete Syntax	• Efficiency
Abstract to Concrete Syntax Mapping	• Task Visibility
Relationships	• Visual Coherence
Constraints	Required Resource (Man-Day)
	License Nature
	Artefact Quality Level
	Artefact Format

The eight studies outlined above found over a hundred single criteria for DSM tool evaluations. In the following, the discussed criteria are analyzed, classified, unified, and adapted to establish a coherent checklist to be used as a guideline for future criteria formulations of DSM tool evaluations.

3.4 A Classification of Evaluation Criteria

Kirchner and Jung (2007) propose an evaluation framework for metamodeling tools, in which they divide the evaluation criteria into two main categories and the latter one further into three subcategories:

- General Evaluation Criteria for Software Tools
- Evaluation Criteria for Metamodeling Tools
  - Tool Architecture
  - Modeling Language Specification
  - Modeling Language Application.

The *general evaluation criteria* includes the criteria of interest to almost any projected acquisition of a software tool, such as initial and operational costs, software ergonomics and documentation, along with the installation and removal of the tool (Kirchner & Jung, 2007). These types of general criteria are described at length by e.g. Rivas et al. (2010) and the ISO 14102 standard document (ISO, 2008, 15). They are not in the core area of this thesis.

The *evaluation criteria for metamodeling tools* contains the criteria of particular interest to aspects dealing with the evaluation of DSM tools. The *tool architecture* subcategory comprises of criteria related to the overall tool architecture, such as modularity, model management, extensibility, and integration. The architecture determines the performance and flexibility of the tool. The *modeling language specification* subcategory deals with the criteria related to DSML specification tasks, such as the general approach to metamodeling and the definition of the language concepts. The *modeling language application* subcategory includes criteria dealing with the aspects of the application of DSMLs, such as the generation of modeling tools, model transformation, simulation, metrics and model documentation. (Kirchner & Jung, 2007.)

The categories are utilized in the classification of the criteria into a unified checklist, presented in the next section.

3.5 Artifact I - Evaluation Criteria Checklist for DSM Tools

The evaluation criteria found in the literature review in Section 3.3 are here classified, unified, and given data types, ranges, and example values for better representation of their semantics. Forty-five representative criteria were adapted to the checklist (see APPENDIX 1). In the adaptation, coverage, compactness, and practicality of the criteria were the key principles, which resulted in the abstraction of some of the criteria into a more general form, the semantic normalization of the criteria in terms of eliminating repetition, and the elimination of some of the more rare criteria (see APPENDIX 2). The data types, ranges, and examples are adapted from the literature and unified to form a coherent representation. In the following, the criteria included in the checklist are described according to the classification presented in Section 3.4.

3.5.1 General Evaluation Criteria for Software Tools

The General Evaluation Criteria category includes the following criteria: Documentation, Customer Support, Licensing Model, Price, Vendor, Version, Stability, Usability, Utility, and Effort. The criteria are presented as a part of the checklist in TABLE 10.

TABLE 10 General Evaluation Criteria for Software Tools

General	**Type**	**Range (or none)**	**Example**
Documentation	String	Set of Documentation Types	Tutorials
Customer Support	String	Set of Support Mechanisms	E-Mail
Licensing Model	String	Set of Licensing Models	Open Source
Price (in Currency X)	Int	Natural Numbers	1000
Provider	String	Set of Provider Names	Microsoft
Version	String	Set of Version Identifications	Beta 1.2
Stability	Enum	Low, Medium, Good	Low
Usability	Enum	Low, Medium, Good	Medium
Utility	Enum	Low, Medium, Good	Good
Effort (e.g. in Man Days)	Int	Natural Numbers	2

Documentation includes all the types of literature, examples, and learning aids provided with the tool. Automatic generation of the documentation of a DSML model should also be available. *Customer Support* includes the support mechanisms provided by the tool vendor. *Licensing Model* includes the terms by which the tool is provided to the customer. *Price* includes the acquisition and operational costs of the tool. *Provider* is the company or community who is providing the tool. *Version* is the identification of a software revision released as the tool. *Stability* is the assessment of the functional reliability of the tool, based on the experiences during the evaluation period. *Usability* applies to all tool components with user interfaces and determines the ease of use and learnability of the tool. *Utility* is the assessment of the practicality and usefulness of the tool. *Effort* is the time and resources required to learn and use the tool to undertake a specific task. (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Pelechano et al., 2006; Sivonen, 2008; Vasiljević et al., 2013)

3.5.2 Evaluation Criteria for Tool Architecture

The Tool Architecture category includes the following criteria: Storage Mechanism, Platform Support, Deployment Model, Extensibility, Integration, Maturity, Meta-metamodel Architecture, Language Validation, Interoperability, Maintainability, Language Evolution, Customizability, and Reusability. The criteria are presented as a part of the checklist in TABLE 11.

TABLE 11 Evaluation Criteria for Tool Architecture

Tool Architecture	Type	Range (or none)	Example
Storage Mechanism	String	Set of Storage Mechanisms	Repository
Platform Support	String	Set of Platform Names	Linux
Deployment Model	Enum	Embedded, Standalone	Standalone
Extensibility	Enum	Low, Medium, Good	Low
Integration	String	Set of Integration Mech.	API
Maturity	Enum	Low, Medium, Good	Low
Meta-Metamodel Architecture	String	Set of Tool Architectures	Four-layer
Language Validation	Bool	Yes, No	Yes
Interoperability	String	Set of Standards	XMI
Maintainability	Enum	Low, Medium, Good	Low
Language Evolution	Enum	Low, Medium, Good	Low
Customizability	Enum	Low, Medium, Good	Low
Reusability	Enum	Low, Medium, Good	Low

Storage Mechanism includes the ways the tool handles the storage of models and other artifacts. *Platform Support* includes the operating systems that the tool can be run on. *Deployment Model* includes the ways in which the tool is available, e.g. as a plugin or a standalone package. *Extensibility* includes the assessment of the extent to which the tool functionality can be augmented by a developer, via e.g. modular architecture design. *Integration* includes the mechanisms by which the tool can be integrated to the development environment. *Maturity* includes the assessment of how evolved and established the tool is in terms of user adaption and functionality. *Meta-Metamodel Architecture* is the type of the architecture of the meta-metamodel fixed in the tool. *Language Validation* is the indication of whether or not the tool has a mechanism for checking the validity of artefacts. *Interoperability* includes the standards adopted in the serialization of language descriptions, defining probability of using the artifacts in another context. *Maintainability* includes the assessment of the level the tool architecture supports maintenance updates and fixes while providing consistent processing of language artifacts. *Language Evolution* includes the assessment of how well the tool handles language artefacts as they are updated. *Customizability* is the assessment of how well the tool architecture supports customization. *Reusability* is the assessment of how well the tool architecture supports reusable artifacts. (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Sivonen, 2008; Vasiljević et al., 2013)

3.5.3 Evaluation Criteria for Modeling Language Specification

The Modeling Language Specification category includes the following criteria: Metamodeling Language, Mutable Logical Levels, Metamodel Syntax Specification, Abstract Syntax (AS) Representation, Concrete Syntax (CS) Representation, Concrete Syntax Style, AS to CS Mapping, Semantics Specification, Constraint Language, Graphical Completeness, Context Adaptive Assistance, Workflow Guidance, and Relationships. The criteria are presented as a part of the checklist in TABLE 12.

TABLE 12 Evaluation Criteria for Modeling Language Specification

Language Specification	**Type**	**Range (or none)**	**Example**
Metamodeling Language	String	Set of Languages	Ecore
Mutable Logical Levels	Int	Natural Numbers	2
Metamodel Syntax Specification	Bool	Yes, No	Yes
Abstract Syntax Representation	Enum	Tree, Graph	Tree
Concrete Syntax Representation	Enum	Text, Graphic, Matrix	Graphic
Concrete Syntax Style	Enum	Declarative, Imperative	Declarative
AS to CS Mapping	String	Set of Mapping Approaches	Model-Based
Semantics Specification	Bool	Yes, No	Yes
Constraint Language	String	Set of Constraint Languages	OCL
Graphical Completeness	Enum	Low, Medium, Good	Low
Context Adaptive Assistance	String	Set of Assistance Techniques	Tooltips
Workflow Guidance	String	Set of Guidance Techniques	Wizard
Relationships	Enum	Binary, N-Ary	Binary

Metamodeling Language is the metamodeling language fixed in the tool. *Mutable Logical Levels* is the number of the logical levels of the meta-metamodel architecture the tool exposes for user manipulation. *Metamodel Syntax Specification* is the indication whether or not the tool allows to specify DSML abstract and concrete syntaxes. *Abstract Syntax Representation* includes the ways the abstract syntax is represented in the tool. *Concrete Syntax Representation* includes the ways the concrete syntax is represented in the tool. *Concrete Syntax Style* is the paradigm the tool utilizes in the definition of CS. *AS to CS Mapping* includes the approaches by which the correspondence between the elements of AS and CS is defined in the tool. *Semantics Specification* is the indication whether or not the tool allows the definition of language semantics. *Constraint Definition* includes the mechanisms for the definition of model constraints the tool supports. *Graphical Completeness* is the assessment of the capability of the tool to represent all desired visual elements of a concrete syntax. *Context Adaptive Assistance* includes the techniques the tool provides to help the user by adapting the form and content of the assistance to the given task at hand. *Workflow Guidance* includes the set of techniques the tool provides to help the user through the steps of the specification process. *Relationships* is the indication whether the metamodeling language support binary or n-ary relationships between objects in metamodels. (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Sivonen, 2008; Vasiljević et al., 2013)

3.5.4 Evaluation Criteria for Modeling Language Application

The Modeling Language Application category includes the following criteria: Model Transformation Capability, Problem to Solution Mapping, Transformation Definition Language, Transformation Output Language, Generated Editor Quality, Artefact Quality, Output Update Mechanism, Viewpoints, and Analysis Capabilities. The criteria are presented as a part of the checklist in TABLE 13.

TABLE 13 Evaluation Criteria for Modeling Language Application

Language Application	**Type**	**Range (or none)**	**Example**
Model Transformation Capability	Bool	Yes, No	Yes
Problem to Solution Mapping	String	Set of Mapping Techniques	Template
Transformation Def. Language	String	Set of Languages	MERL
Transformation Output Language	String	Set of Languages	Java
Generated Editor Quality	Enum	Low, Medium, Good	Low
Artefact Quality	Enum	Low, Medium, Good	Low
Output Update Mechanism	Enum	Destructive, Incremental	Destructive
Viewpoints	String	Set of Viewpoints	DSL Explorer
Analysis Capabilities	Enum	Low, Medium, Good	Low

Model Transformation Capability is the indication of whether or not the tool supports model transformations. *Problem to Solution Mapping* includes the techniques the tool provides on how transformations are specified. *Transformation Definition Language* includes the languages supported by the tool in transformation specification. *Transformation Output Language* includes the languages (or any) that can be generated with transformation facility of the tool. *Generated Editor Quality* is the assessment of the quality of the modeling editor generated by the tool from the DSML specifications. *Artefact Quality* is the assessment of the quality of the artefacts produced by the generated editor. *Output Update Mechanism* is the indication whether consecutive transformations of the same models reflect on the output destructively or incrementally. *Viewpoints* includes the set of viewpoints the tool provides to the transformation design. *Analysis Capabilities* is the assessment of the capability of the tool to provide analytics of the models described with the generated editor. (Amyot et al., 2006; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Vasiljević et al., 2013)

3.6 Summary

This chapter discussed the evaluation of DSM tools. First, the evaluation of DSM tools was considered on a general level. Second, four sets of requirements for DSM tools were presented based on four previous studies. Third, evaluation criteria for DSM tools, presented in eight previous studies, were outlined. Fourth, a comprehensive classification of evaluation criteria for DSM tools, derived from existing criteria, was introduced. Finally, the classified evaluation criteria were adapted into a checklist with unified data types, ranges, and examples of criteria values, effectively forming Artifact I.

After all the existing evaluation criteria were combined into Artifact I, it can be concluded that the various characteristics of DSM tools were covered quite adequately in the previous studies as a whole. No single study was comprehensive in terms of evaluation criteria. Nevertheless, during the analysis it was remarked that some relevant DSM tool characteristics that were indicated as requirements for DSM tools, were not covered in previous studies as evaluation criteria. Such characteristics are tool support for testing and debugging of DSML artifacts as well as mapping mechanisms for abstract syntax to semantics. These characteristics have not been considered as evaluation criteria, although they came up in the literature. Criteria mapped to those requirements may be considered as potential additions to the checklist.

4 SOCIO-TECHNICAL DIMENSIONS AND METHODS OF EVALUATION

This chapter discusses the socio-technical dimensions of IS and ISD tool evaluations as well as the existing methods for the evaluation of ISD tools. As DSM tools are utilized by human beings in a situational context, the socio-technical dimensions of evaluation should be considered. Furthermore, in Section 2.6 it was outlined that DSM tools are a specific type of ISD tools that enable the design and generation of customized ISD tools. As the literature provides very limited methodical support for the evaluation of DSM tools, existing evaluation methods for ISD tools are discussed in this chapter for the purpose of investigating the baseline concepts for the method elements of situational evaluation methods for DSM tools discussed in Section 5.2.

First, the socio-technical dimensions of ISD tool evaluations are discussed. The dimensions are aligned with the situational factors of SME in Section 5.3. Second, existing evaluation methods and a classification of their activity elements are outlined. The classification is utilized in the synthesis of the method elements of Artifact II in Section 5.2. Finally, a summary of the chapter is presented.

4.1 Socio-Technical Dimensions of Evaluation

Evaluation of IS artifacts has been extensively studied during the last three decades (Song & Letch, 2012). (Paul, 2007, p. 194) defines the information system (IS) as "… what emerges from the usage that is made of the IT delivery system by the users (whose strengths are that they are human beings not machines)". IS evaluation is a social endeavor, which provides versatile feedback to managers and assists with organizational learning processes (Irani & Love, 2002). Land (2001) argues that IS evaluation is difficult and that decision-makers consider IS evaluation problematic, because the prediction of costs, benefits, risks, impact, and lifetime of IS is challenging. Jones (2008) contrasts between mechanistic and interpretive IS evaluation. Mechanistic IS evaluation refers to formal, mainly economic evaluation of IS, primarily concerned with costs and benefits of IS. Interpretive IS evaluation is regarded as a socially embedded process, which appreciates the value of the views and opinions of the organizational IS users for IS evaluation purposes (Jones, 2008).

It can be argued that the application of the interpretive approach, at least in a complementary capacity to the mechanistic evaluation, could potentially be beneficial to the evaluation of DSM tools in a situational context, as the involvement and the views and opinions of the stakeholders would provide informed data for decision-makers, to be considered as a valuable part of the evaluation. The significant human component, embedded in the concept of IS, underpinning the appropriate and successful use of IS, should be a notable factor in assessing the potential benefits, costs and risks of the situational context (Irani & Love, 2008). Thus, the three key components, also pivotal in business case building (see APPENDIX 3), benefits, costs, and risks, form the cornerstones of an evaluation process (Irani & Love, 2008). Song and Letch (2012) describe IS evaluation as:

> "…a multifaceted and complicated phenomenon which can be examined from multiple perspectives. As a domain of study it can be considered to be an interactive social system that is interwoven with different stakeholders, various resources and multiple decision-making processes".

The widely accepted classification of dimensions of IS evaluation is described by the Content, Context, and Process (CCP) framework by Stockdale, Standing, Love and Irani (2008), illustrated in FIGURE 12. The *Content* dimension addresses what is evaluated, i.e. the object of evaluation, the evaluation criteria, and changes caused by IS. The *Context* dimension captures why the evaluation is conducted as well as who is involved in it. Context addresses both the internal and external context of an organization. In the internal context, aspects such as organizational structure and culture, business strategies, management procedures, and social influences are addressed. The external context addresses factors such as technologies, market structures, economic situation, and government policies. As the underlying motivations of the stakeholders for the conduct of evaluations are assessed, the internal and external contexts are effectively captured. The *Process* dimension focuses on when the evaluation is carried out, and how it is conducted. In the Process dimension, evaluation method, i.e. the actions, reactions, and interactions of stakeholders involved in an evaluation are addressed as well as the timeframe of evaluation, i.e. before, during, or after the implementation of IS. (Song & Letch, 2012; Stockdale et al., 2008)

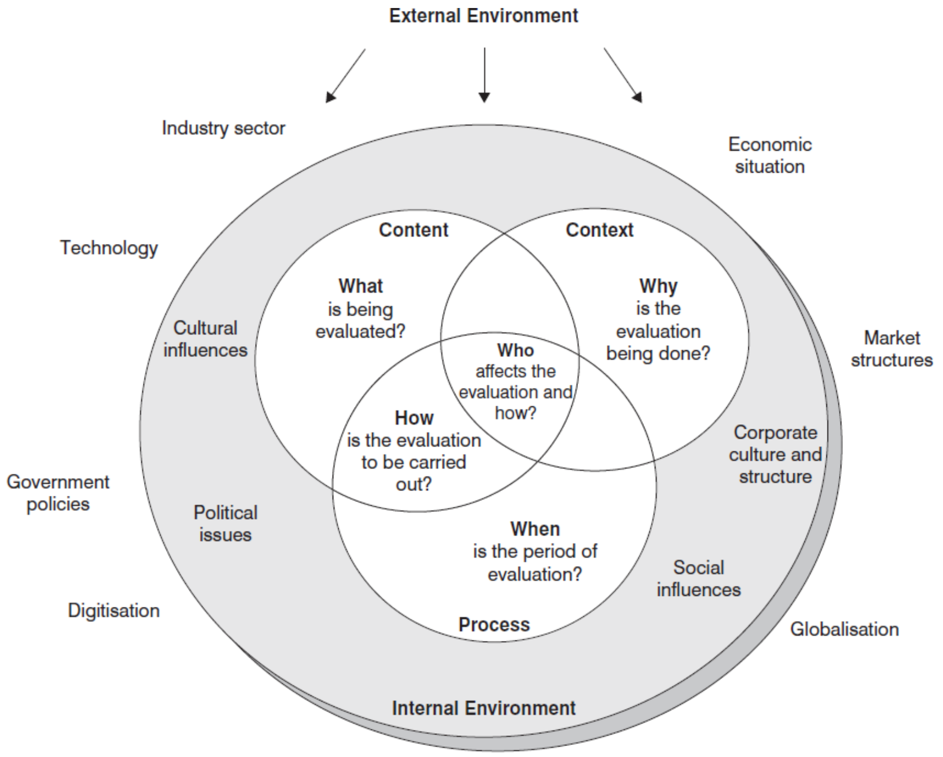


FIGURE 12 Content Context Process (CCP) Framework (Stockdale et al., 2008, p. 43)

Lundell and Lings (2004b) propose facets similar to CPP as the core elements of IS Development (ISD) tool evaluation, further stressing the dimensions of Stakeholders, Context, and Activity, as illustrated in FIGURE 13. They again emphasize that the evaluation of ISD tools, such as DSM tools, is a complex social process, strongly influenced by the motivations and goals of the stakeholders. There is a dynamic between an evaluation activity and the context in which it is taking place, yielding outcomes to the stakeholders and the context. Another central concept is the evaluation framework, which includes the requirements and the criteria for the evaluation process (Lundell & Lings, 2004b). Conceptually, Stakeholders and Context reside in the Context of CPP, and Activity is a container for Content and Process of CPP.

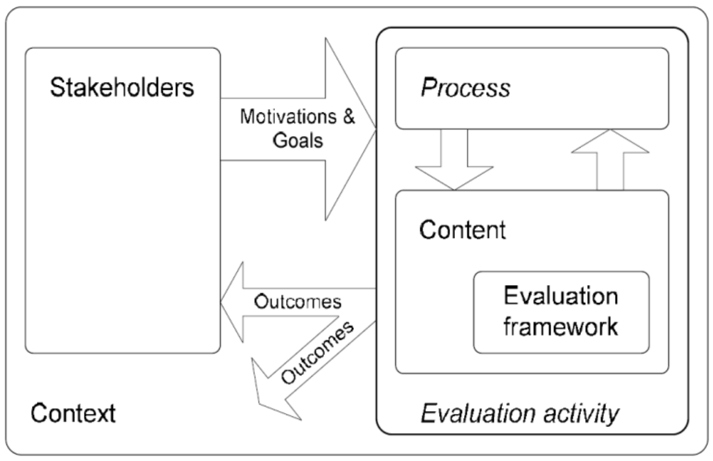


FIGURE 13 Elements of ISD Tool Evaluation (Lundell & Lings, 2004b, p. 40)

The *Stakeholders* dimension addresses the selection of stakeholders for evaluation as well as the social issues related to them. Evaluation is a political activity, with potentially strong impact on the stakeholders, which is why the selection of the stakeholders and the evaluator is a critical issue. The stakeholders must feel able to actively participate and their beliefs and assumptions must be considered, in order to produce valid and acceptable findings. When selecting stakeholders, it is important to have a clear strategy e.g. to cover all the roles related to a process life-cycle within a context. Validity of findings is supported by the representativeness of the stakeholders as well as the multiple perspectives they offer. An ongoing feedback is essential for retaining a sense of process ownership for the stakeholders. Nevertheless, it is important to acknowledge that each stakeholder has its personal motivations and goals, which should be taken into account e.g. in the overall goal setting of evaluations. Evaluator is a special role, which can be assigned to an internal stakeholder or an external agency, depending on the case. The selections of the stakeholders to be included in the evaluation are political decisions that should ideally be resolved transparently, and to the satisfaction of all the stakeholders, before any commenced evaluation activity. Failure to consider the Stakeholder dimension can potentially result in a feeling of lack of process ownership by the stakeholders, leading to poor evaluation results, economic consequences, and political fallout. (Lundell & Lings, 2004b.)

The *Context* dimension is divided into internal and external context, both including a history facet, representing the continuous change of the contexts. The internal context is further divided into usage and organizational context. The direct user of a tool resides in the usage context, whereas the manager of the user that facilitates evaluations resides in the organizational context. Typically, stakeholders of the usage context perceive the impact of the outcomes of an evaluation as strong for them, which affects their motivations and goals. Stakeholders of the organizational context have higher-level goals, which effectively function as the constraints of the evaluation activities. The external context is typically beyond the control of the organization in which the evaluation is conducted, including factors such as national and local government policy, markets and market demands, supplier availability and expertise, and other environmental pressures. Failure to consider the Context dimension may result in poor relevance of the outcomes to the organization, and in the extreme the evaluation might turn into a context-free comparison of tool features. (Lundell & Lings, 2004b.)

The *Activity* dimension addresses the approach and the associated set of activities, i.e. the evaluation method. The suitability of a given method is dependent on the underlying assumptions of the context of the activities and the stakeholders involved in the activities. There is a number of different evaluation methods proposed in the literature, such as the ISO 14102 standard (ISO, 2008). Failure to consider the Activity dimension can result in poorly conducted evaluations with unreliable outcomes, which leads to a lack of stakeholder confidence in the evaluation activities. (Lundell & Lings, 2004b.)

Kitchenham (1996) suggests that the following human factors affect the evaluation efforts of ISD tools: novelty effects and expectation effects. The *novelty effects* arise from the stakeholders' altered behavior, propagating from the unfamiliarity of the usage situation, such as the learning curve effect and the Hawthorne effect. The *learning curve effect* refers to the observation that people's skills and ultimately expertise are developed only as they gain familiarity with the application of the tools, which tends to counteract the perceived positive effects inherent in the new tools evaluated. The *Hawthorne effect* refers to the observation that the stakeholders that are under evaluation, tend to work more conscientiously, because they feel they are under more management scrutiny, which may exaggerate the positive effects inherent in a new tool evaluated. The *expectation effects* refer to the bias resulting from the stakeholders' preconceived expectations about the evaluated tools, such as the placebo effect and doctor effect. The definition of the *placebo effect* suggests that the beliefs of an individual stakeholder may incidentally have a positive effect on some characteristic under evaluation, but such beliefs are not generalizable. The *doctor effect* refers to phenomena such as the unbalanced target effect and intervention effects. The *unbalanced target* effect refers to the scenario in which the stakeholders of the evaluation concentrate their efforts towards the explicit goals of the evaluation, trading-off against the implicit goals, such as documentation quality, which results in unrealistic evaluations. The *intervention effect* refers to the scenario in which the evaluators compromise the normal behavior of the evaluated subjects by encouraging them to modify their behavior or by conveying the initial expectations of the outcomes of the evaluation effort to the subjects. (Kitchenham, 1996)

The following sections emphasize the Activity dimension, along which the evaluation methods are usually structured and disseminated. The other dimensions are highly situational, thus they are meaningful in the context in which the method is utilized. These situational factors are discussed in Section 5.3 from the perspective of SME.

4.2 Existing Evaluation Methods

This section presents seven evaluation methods for ISD tools: ISO (2008), Lundell & Lings (2003), Kitchenham (1996), Kruchten (2004), Lukman & Mernik (2008), Wheeler (2011), and Morera (2002). This set includes all the recent evaluation methods for ISD tools that were found in the literature review. Furthermore, Kitchenham (1996) was included, as it is widely adopted and provides a unique method for the selection of an evaluation approach. The existing evaluation methods address mainly the Activity dimension, leaving the other dimensions largely undiscussed. The Stakeholder and Context dimensions are highly situational, which has likely steered the emphasis on the general evaluation methods towards the Activity dimension, which can be structured and disseminated as process models. The ISO 14102 standard provides the most comprehensive guidance available, describing the reference evaluation activities in detail. The 2G method by Lundell and Lings (2003) is the only method which specifically addresses the Stakeholder and Context dimensions, including exhaustive interviews with the stakeholders and analysis of organizational data. The DESMET method by Kitchenham (1996) is an evaluation method which can be utilized for the selection of the approach for an evaluation method. The rest of the methods are general evaluation methods for ISD tools, except for Lukman and Mernik (2008), which briefly describes an evaluation method for MDE tools.

4.2.1 ISO 14102 Standard

ISO 14102 (ISO, 2008) is an international standard for the evaluation and selection of ISD tools, following the software product evaluation model of ISO 14598-5 (ISO, 1998), and adopting the general model of software quality characteristics defined in ISO 9126-1 (ISO, 2001). Supplemental technical report TR 14471 provides further guidance on the adoption of a selected ISD tool. ISO 14102 stresses objectivity, repeatability, and impartiality as well as careful consideration of both the technical and management requirements in the evaluation activities. The standard is intended as a normative reference from which evaluation methods are to be tailored, according to the organizational needs. The standard evaluation process is divided into four phases: Preparation, Structuring, Evaluation, and Selection, as illustrated in FIGURE 14. The standard also includes an extensive list of predefined ISD tool characteristics, which can be utilized as a reference for situational evaluation framework formulation. (ISO, 2008)

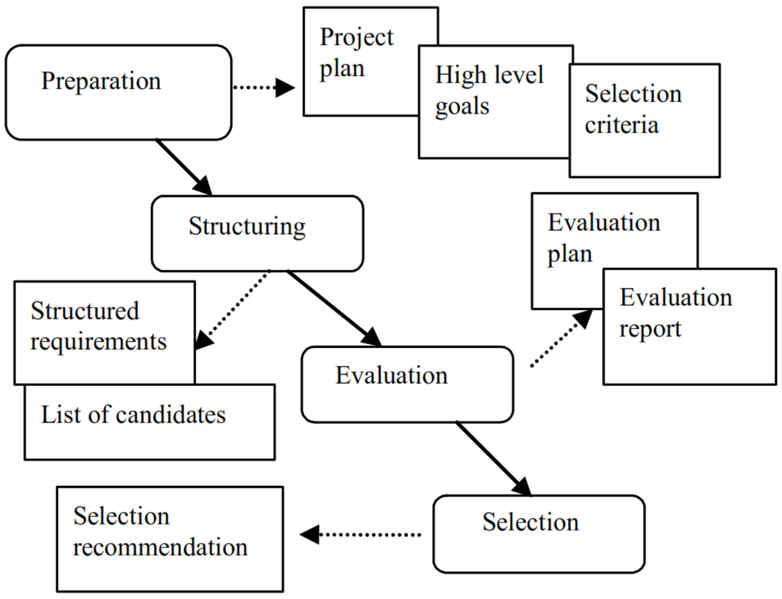


FIGURE 14 Overview of ISO 14102 (ISO, 2008, p. 4)

The *Preparation* phase addresses the organizational needs for an evaluation project, including the following main activities: goal setting, establishing selection criteria, and project planning and control. The goal setting activity produces content such as the high-level goals and also addresses the development of the rationale and general policy for tool acquisition. The establishing selection criteria activity produces content such as tentative selection criteria by the analysis of the high-level goals and addresses issues such as the relative importance of the criteria, data collection methods, and scope of the evaluation. The selection criteria are the high-level criteria utilized finally in the selection phase, which are not to be mixed with the evaluation criteria of the evaluation framework. The project planning and control activity produces content such as a project plan and addresses issues such as the project team setting, structuring of the high-level goals and the selection criteria, scheduling of activities and tasks with resource requirements and cost estimates as well as the means for monitoring and controlling the execution of the plan. The project plan is the primary content produced in the phase, after which the process enters the structuring phase. (ISO, 2008.)

The *Structuring* phase addresses the evaluation framework and includes the following main activities: requirements definition, tool information gathering, and identifying the final candidate tools. The requirements definition activity produces the structured requirements by organizational information gathering, requirements identification, and requirements structuring. The evaluation framework is constructed as a result of mapping the structured requirements to evaluation criteria. Tool information gathering activity is a general search for tool candidates and their characteristics, utilized to quickly identify and exclude candidates for evaluation. Identifying final candidate tools activity produces a list of the final candidates, which is constructed by comparing critical requirements against tool characteristics. In the end of the phase, an evaluation framework, structured requirements and the final list of tool candidates are completed. Next, the evaluation phase is initiated. (ISO, 2008.)

The main activities of the *Evaluation* phase are: preparing for evaluation, evaluating the tools, and evaluation reporting. The preparing for evaluation activity produces content such as an evaluation plan, which includes the metrics, rating levels, and standards for the evaluation criteria as well as the setting and scheduling of all the activities of the phase. The evaluating the tools activity is a process of measurement, rating, and assessment, in which the tools are evaluated against the evaluation criteria. Tasks such as examining the tool documentation, interviewing the stakeholders, and applying the tool to test projects are conducted. The evaluation reporting activity produces the evaluation report, documenting the data collected during the evaluation in detail. The phase ends as the evaluation report is finished, allowing the selection phase to begin. (ISO, 2008.)

The main activities of the *Selection* phase are the following: preparing for selection, applying the selection algorithm, recommending a selection decision, and validating the selection decision. The preparing for selection activity takes the previously structured high-level goals and selection criteria as inputs and addresses issues such as how the generated evaluation data is combined and compared to produce ratings for the tool candidates, setting of the final selection criteria, and the definition of the selection algorithm. The applying the selection algorithm activity takes the evaluation data as input and produces aggregated comparison data about the tools. The recommending a selection decision activity produces the tool selection recommendation as a result of the decision-making of the management, grounded on the comparison data. The recommendation indicates the most appropriate tool for acquisition. Alternatively, the results may indicate needs for additional information, which could implicate a need for another iteration of the previous activities. In the validating the selection decision activity the original goals, selection guidelines and other related data should be reviewed and compared against the recommended selection in order to validate that the high-level goals are met by the selected tool. If no adequate tool exists, a decision between custom tool development, modification of an existing tool, or abandonment of the entire undertaking should be made. Finally, a selection report is produced. (ISO, 2008.)

4.2.2 2G Method

Lundell and Lings (2003) propose the 2G method for the support of the construction of situational evaluation frameworks for ISD tool evaluations. 2G is strongly influenced by Grounded Theory (GT), a qualitative research methodology emphasizing the generation of theory from data in the process of conducting research (Urquhart et al., 2010). 2G provides a systematic method for the construction of situational evaluation frameworks. 2G was developed through field studies in small software development companies, refined in academic environment, and finally validated in a large scale industrial setting. 2G specifically addresses the integration of the "softer" social and organizational requirements addressing the Stakeholder and Context dimensions, with the more detailed technical aspects of technology. Since 2G utilizes GT, the definitions of evaluation concepts evolve, and the structure for interrelating these concepts emerges during analysis, thus enabling context-specificity. In comparison to other methods, 2G can be characterized as data-driven, whereas e.g. ISO 14102 could be classified as concept-driven, having predefined evaluation concepts, such as tool characteristics, organized into hierarchical structures. 2G is presented in FIGURE 15. (Lundell & Lings, 2003.)

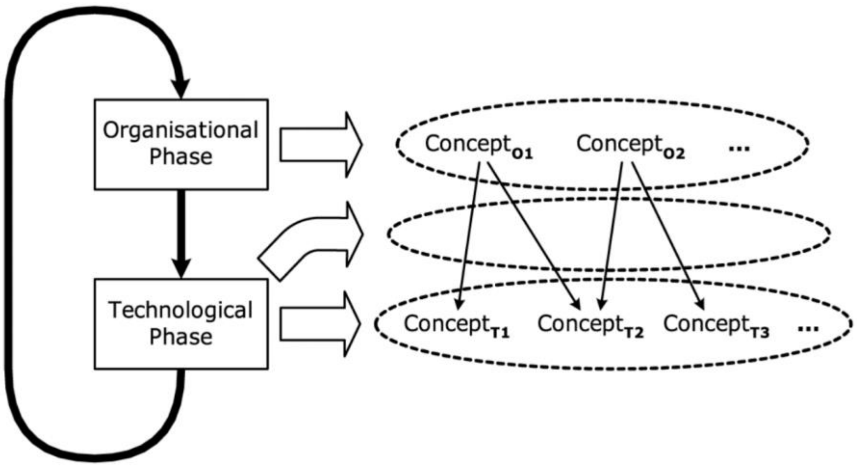


FIGURE 15 2G Method for ISD Tool Evaluation (Lings & Lundell, 2005, p. 200)

In the *organizational phase* of the 2G method, an organizational evaluation framework, grounded in context data, is developed. The data is processed into evaluation concepts by analyzing the data for supporting indicators. Derived concepts are then interrelated by e.g. GT categorization (Lings & Lundell, 2005). In the *technological phase*, a technological evaluation framework is constructed, which is grounded in the current state-of-the-art technology data. Thus, 2G provides double grounding of evaluation concepts. 2G utilizes various data sources as the basis for analysis. At the initiation of 2G, a relevant set of data sources is selected. Some of these will exist, such as organizational manuals, policy documents, and standards. Others will be generated, such as the transcripts of stakeholder interviews. The data sources are analyzed with the goal of developing a set of interrelated concepts with agreed interpretation. (Lundell & Lings, 2003.)

FIGURE 16 represents an example of the two evaluation frameworks and their concept interrelations. As the data is analyzed in the organizational phase, the emerged tool requirements are structured as the organizational framework concepts. In the technological phase the concepts are linked to concrete tool concepts in the technological framework. The links are also assigned prioritizations, which indicate the relevance of the concepts to the context. Thus, the evaluation criteria are derived from the technological framework, on the basis of the prioritizations. (Lings & Lundell, 2005.)

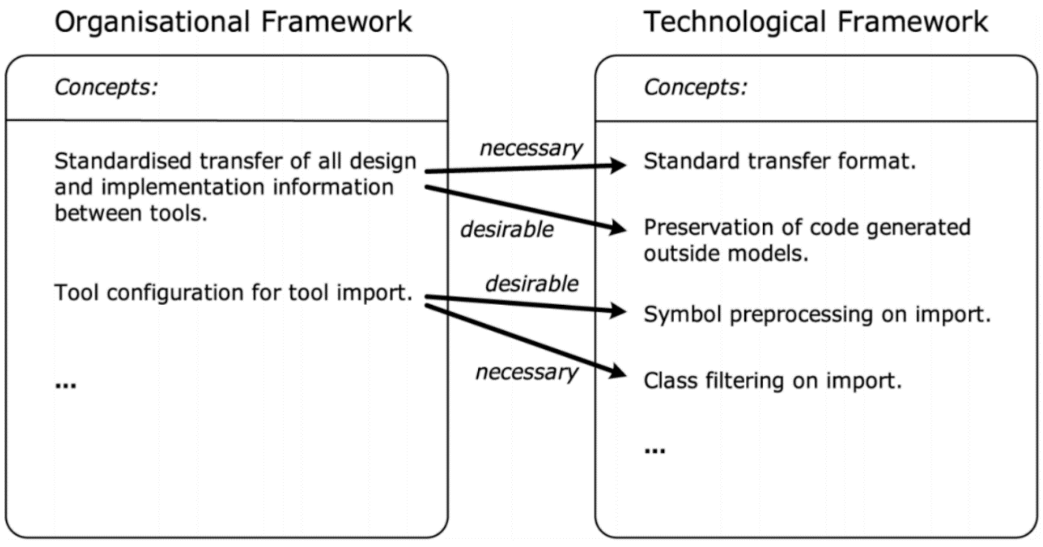

Organisational Framework
Concepts:
Standardised transfer of all design and implementation information between tools.
Tool configuration for tool import.
...
Technological Framework
Concepts:
Standard transfer format.
Preservation of code generated outside models.
Symbol preprocessing on import.
Class filtering on import.
...
necessary
desirable
desirable
necessary

FIGURE 16 Concept Linking between Evaluation Frameworks (Lings & Lundell, 2005, p. 205)

4.2.3 DESMET Method

The DESMET method (Kitchenham, 1996) has been widely adopted and studied in academic and industrial settings since its inception in the mid-nineties. DESMET is an evaluation method for both ISD tools and methods. It provides extensive guidance on various facets of ISD tool evaluation such as the selection of the evaluation approach, human factors affecting evaluations, and activities of evaluation approaches. In the context of our work the selection of the evaluation approach is highly relevant. DESMET is applied by Morera (2002), which is discussed in the next section.

Kitchenham (1996) proposes the following approaches to evaluation: quantitative experiment, quantitative case study, quantitative survey, qualitative screening, qualitative experiment, qualitative case study, qualitative survey, qualitative effects analysis, and benchmarking. *Quantitative experiment* is an investigation of the quantitative impact of the tools organized as a formal experiment. *Quantitative case study* is an investigation of the quantitative impact of the tools organized as a case study. *Quantitative survey* is an investigation of the quantitative impact of the tools organized as a survey. *Qualitative screening* is a feature-based evaluation done by a single individual who not only determines the features to be assessed and their rating scale but also does the assessment. For initial screening, the evaluations are usually based on literature describing the software tools rather than actual use of tools. *Qualitative experiment* is a feature-based evaluation done by a group of potential users who are expected to try out the tools on typical tasks before making their evaluations. *Qualitative case study* is a feature-based evaluation performed by someone who has used the tool on a real project. *Qualitative survey* is a feature-based evaluation done by people who have had experience of using the tool, or have studied the tool.

The difference between a survey and an experiment is that participation in a survey is at the discretion of the subject. *Qualitative effects analysis* is a subjective assessment of the quantitative effect of the tools, based on expert opinion. Finally, *Benchmarking* is a process of running a number standard tests using alternative tools and assessing the relative performance of the tools against those tests. (Kitchenham, 1996.)

Kitchenham (1996) provides technical criteria regarding the selection of the evaluation approach. The systematic assessment of the technical criteria provides indications of which evaluation approach is the optimal choice for a given situation. The technical criteria are: the evaluation context, the nature of the expected impact of using the tool, the nature of the tool to be evaluated, the scope of impact of the tool, the maturity of the tool, the learning curve associated with the tool, and the evaluation maturity of the organization undertaking the evaluation. The *evaluation context* addresses four scenarios of industrial evaluation of ISD tools: selection of tools for an individual project, initial screening of tool candidates followed by a detailed evaluation, change monitoring as a part of a process improvement program, and evaluation of tools for re-sale as part of a larger product or a product line. The *nature of impact* identifies two major categories of impact: quantitative and qualitative. The quantitative category includes numerically measurable impacts such as improved productivity, better maintainability, and better quality. The qualitative category includes e.g. better visibility of progress, better usability of support tools, improved interoperability of tools, and commonality of tool interfaces. The nature of the tool category divides the evaluation of several tool candidates into two categories: tools which support the same basic ISD method, and tools that support quite different ISD methods. The *scope of the impact* addresses two dimensions: product granularity and extent of impact. The product granularity identifies whether the tool applies to the development of a product as a whole or individual parts of the product such as modules. The extent of impact identifies how the effect of the tool is likely to be felt over the product/project life-cycle. The *maturity of the tool* indicates the extent of how likely there is information available about the tool. The concept can assessed in terms of following categories: not used in commercial projects, used in a few state-of-the-art projects in home organization, and in widespread use in the home organization. The *learning curve* indicates the time it would take an individual to become familiar enough with the tool to access its capabilities or to use it effectively in the evaluation. The *evaluation maturity of the organization* determines the type of evaluation an organization is able to take. The level of adaptation, adherence and monitoring of well-defined ISD standards, are regarded as the key metrics in assessing the criterion. (Kitchenham, 1996.)

The results of the technical criteria assessment are further analyzed by matching them to the favoring conditions presented in TABLE 14. The outcome is the recommendation of an evaluation approach or a combination of the approaches. (Kitchenham, 1996.)

TABLE 14 Selection of Evaluation Approach (Kitchenham, 1996, p. 11)

Evaluation Approach	**Conditions Favoring the Approach**
Quantitative experiment	Benefits clearly quantifiable. Staff available for taking part in experiment. Tool related to a single task/activity. Benefits directly measurable from task output. Relative small learning time. Desire to make context independent tool assessments.
Quantitative case study	Benefits quantifiable on a single project. Benefits quantifiable prior to product retirement. Stable development procedures. Staff with measurement experience. Timescales for evaluation commensurate with the elapsed time of your normal size projects.
Quantitative survey	Benefits not quantifiable on a single project. Existing database of project achievements including productivity, quality, tool data. Projects with experience of using the tool.
Qualitative screening	Large number of tools to assess. Short timescales for evaluation exercise.
Qualitative experiment	Benefits difficult to quantify. Benefits directly observable from task output. Relatively small learning time. Tool user population very varied.
Qualitative case study	Benefits difficult to quantify. Benefits observable on a single project. Stable development procedures. Tool user population limited. Timescales for evaluation commensurate with the elapsed time of your normal size projects.
Qualitative survey	Benefits difficult to quantify. Tool user population very varied. Benefits not observable on a single project. Projects with experience of using the tool, or projects prepared to learn about the tool.
Qualitative effects analysis	Availability of expert opinion assessments of tools. Lack of stable development procedures. Requirement to mix and match methods/tool. Interest in evaluation of generic tools.
Benchmarking	Tool not human-intensive. Outputs of tool able to be ranked in terms of some "goodness" criteria.

4.2.4 More Evaluation Methods

Rational Unified Process (RUP) is a popular ISD method originally developed by Rational Software (Kruchten, 2004). RUP provides extensive guidance for activities of ISD life-cycle processes, one of which is the *Select and Acquire Tools* method, which is outlined here. The method is divided into five main activities, which are presented in TABLE 15. The method provides guidance on tasks, evaluation criteria, and rating mechanisms for the activities.

Lukman and Mernik (2008) present a method for the procurement of metamodeling tools they utilized in a research project. As the initiation of the method, they suggest the decision of the stakeholders to introduce MDE into their organization. It also identifies the analysis of the target DSML as a key input for the process. This study briefly describes the only evaluation method for model-driven ISD tools that we were able to locate in the existing literature. The method is divided into four main activities, presented in TABLE 15.

TABLE 15 Activities of Kruchten (2004) and Lukman and Mernik (2008)

Kruchten (2004)	**Lukman and Mernik (2008)**
Identify Needs and Constraints	Requirements Specification
Collect Information about Tools	Market Analysis
Compare Tools	Tool Evaluation
Select Tools	Tool Selection
Acquire Tools	

Wheeler (2011) describes a general method for evaluating software applications, providing specific guidance on how to evaluate free and open source software (FOSS). The method focuses primarily on context-free comparisons of application characteristics, mainly for the purpose of evaluating potential FOSS candidates against commercial products. The method is divided into four main activities, presented in TABLE 16.

Morera (2002) presents an evaluation method for Commercial-of-the-Shelf (COTS) products, which applies the feature analysis method of DESMET for the evaluation and Analytic Hierarchy Process (AHP) for the selection. Commercial DSM tools can be considered as COTS products. Morera (2002) suggests that the method makes COTS selection less human-dependable and more straightforward by providing strictly quantitative means for decision-making. The method is divided into nine main activities, presented in TABLE 16.

TABLE 16 Activities of Wheeler (2011) and Morera (2002)

Wheeler (2011)	**Morera (2002)**
Identify Candidates	Set Roles and Responsibilities
Read Existing Reviews	Set Time Scale and Effort
Briefly Compare the Leading Programs' Attributes to your Needs	Specify Assumptions and Constraints
Perform an In-Depth Analysis of the Top Candidates	Define Scope and Candidates
	Select an Evaluation Method
	Evaluate and Present Results
	Identify Selection Criteria
	Evaluate Final Candidates
	Agree on Final Decision

4.3 Classification of Activities of Existing Evaluation Methods

The phases of the ISO 14102 standard are utilized as categories for the classification of the main evaluation activities of the previously discussed methods. Some of the activities are originally described at convergent granularity levels, which is taken into account during the analysis. Thus, the categories are Preparation, Structuring, Evaluation, and Selection. The classification is presented in TABLE 17. The *Preparation* category includes activities related to the organizational needs, project management and evaluation approach selection. The *Structuring* category includes activities related to the setting of the evaluation framework by requirements engineering and selection of the tool candidates for evaluation. The *Evaluation* category includes activities related to the evaluation data collection, which is carried out by utilizing the evaluation framework. The *Selection* category includes activities related to the evaluation data analysis and making a recommendation for a tool.

The ISO 14102 standard provides the most comprehensive guidance for an evaluation effort of ISD tools. Kruchten (2004) presents a straightforward approach to the evaluation of ISD tools, providing guidance from Preparation to Selection, closely integrated to the RUP ISD method. Lukman and Mernik (2008) propose a high-level evaluation method for MDE tools, providing activities for Structuring, Evaluation and Selection. Wheeler (2011) suggests an evaluation method specialized for OSS ISD tools, providing activities for Structuring and Evaluation. Morera (2002) incorporates DESMET feature analysis and AHP selection procedure for the evaluation of COTS products, providing activities for all the categories. The 2G method provides guidance for the formulation of the evaluation framework as activities of Structuring. The classification provides a baseline for the structuring of the method elements in Section 5.2.

TABLE 17 Classification of Activities of Existing Tool Evaluation Methods

Classification	**ISO (2008)**	**Kruchten (2004)**	**Lukman and Mernik (2008)**	**Wheeler (2011)**	**Morera (2002)**	**Lundell and Lings (2003)**
Preparation	Goal Setting, Establishing Selection criteria, Project Planning and Control	Identify Needs and constraints			Set Roles and Responsibilities, Set Time Scale and Effort, Specify Assumptions and Constraints, Select an Evaluation Method	
Structuring	Requirements Definition, Tool Information Gathering, Identifying Final Candidate Tools	Collect Information about Tools	Requirements Specification, Market Analysis	Identify Candidates, Read Existing Reviews, Briefly Compare the Leading Programs' Attributes to your Needs	Define Scope and Candidates	Develop Strategic Evaluation Framework, Develop Pragmatic Evaluation Framework
Evaluation	Preparing for Evaluation, Evaluating the Tools, Evaluation Reporting	Compare Tools	Tool Evaluation	Perform an In-Depth Analysis of the Top Candidates	Evaluate and Present Results	Evaluation
Selection	Preparing for Selection, Applying the Selection Algorithm, Recommending a Selection Decision, Validating the Selection Decision	Select Tools, Acquire Tools	Tool Selection		Identify Selection Criteria, Evaluate Final Candidates, Agree on Final Decision	

4.4 Summary

This chapter discussed the socio-technical dimensions of IS and ISD tool evaluations as well as the existing methods for the evaluation of ISD tools. The socio-technical dimensions should be considered, as DSM tools are utilized by human beings in a situational context. It was outlined in Section 2.6 that DSM tools are a specific type of ISD tools that enable the design and generation of customized ISD tools. As the literature provides very limited methodical support for the evaluation of DSM tools, existing evaluation methods for ISD tools were discussed in this chapter, for the purpose of investigating the baseline concepts for the method elements of situational evaluation methods for DSM tools.

The socio-technical dimensions of ISD tool evaluations were discussed first by introducing the related discourse in the field, then by presenting the CPP framework of IS evaluation, and finally by aligning the dimensions of CPP to the elements of ISD tool evaluations. The elements, Stakeholders, Context, and Activity are employed as the structural dimensions of the evaluation efforts, upon which evaluation method concepts are reflected in the following sections. Furthermore, the human factors affecting evaluation efforts, such as the novelty effects and the expectation effects were discussed. Next, a set of existing evaluation methods and a classification of their activity elements were outlined. The ISO 14102 standard was introduced as the most comprehensive approach to an evaluation effort, including the phases of Preparation, Structuring, Evaluation, and Selection. Then, the 2G method was discussed as a means to appropriately address the dimensions of evaluation in the formulation of the evaluation criteria. The DESMET method was presented as a method for the selection of the overall approach of the evaluation, such as case study or experiment. Furthermore, four other evaluation methods were discussed, and finally, their activities were classified according to the phases of ISO 14102. The classification is utilized in the next Chapter, in the effort to produce a synthesized method base for Artifact II.

5 ARTIFACT II – A BASELINE METHOD FOR THE ENGINEERING OF SITUATIONAL EVALUATION METHODS FOR DSM TOOLS

This chapter discusses the components of Artifact II, based on the principles of Situational Method Engineering (SME). First, the SME foundation is presented. Second, a conceptual method base and its three types of method elements: WorkUnit, WorkProduct, and Producer are defined. Third, situational factors are discussed on a general level, and aligned with the previously discussed socio-technical dimensions of the evaluation. Fourth, various approaches to method construction and tailoring, i.e. the construction guidelines are presented. Fifth, the resulting high-level architecture of Artifact II is presented, based on the previous sections. Sixth, Artifact II is conceptually instantiated in an agile usage situation. Finally, a summary of the chapter is presented.

5.1 Situational Method Engineering Foundation

Situational Method Engineering (SME) is a process of constructing or modifying a method for a particular situation, as opposed to adapting a method "as is" by following a specific method guideline "by the book" (Henderson-Sellers et al., 2014, p. 3). The main constructs of SME are illustrated in FIGURE 17. Method engineer is the central role of SME, being responsible for constructing and/or tailoring situational methods according to situational factors and construction guidelines. Instances of situational methods are then enacted by project managers in their respective situational contexts. The series of practical activities taken in the real world, due to the enactment of the situational method, are called as the method in action (Lundell & Lings, 2004a; Ågerfalk & Fitzgerald, 2005). The method base is the construct which contains individual method elements. The method elements are instances of classes in process engineering metamodels, i.e. they conform to specific meta-constructs that govern their structure, relationships, and cardinalities. (Henderson-Sellers et al., 2014, p. 4.)

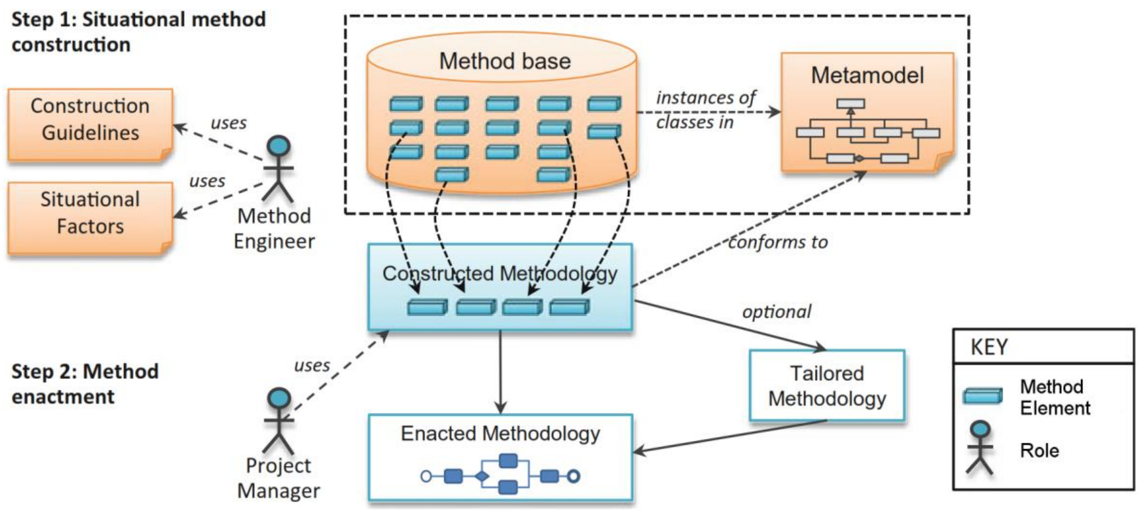


FIGURE 17 Situational Method Engineering Framework, Adapted from Henderson-Sellers et al. (2014, p. 4)

There are various process engineering metamodels available, emphasizing different viewpoints to methods, such as activity, product, decision, context, and strategy viewpoints (Hug, 2009). Artifact II is activity-oriented, as the method elements are derived from activity-oriented evaluation methods. Activity-oriented process engineering metamodels enable the building of models that concentrate on the activities performed in the construction of products, along with their ordering (Hug, 2009). There are a number of activity-oriented process engineering metamodels for ISD, such as SPEM (OMG, 2008b), OPEN Process Framework (OPF) (Firesmith & Henderson-Sellers, 2002), ISO 24744 (ISO, 2007), and Essence (OMG, 2014). In order to be of practical use in a real-world usage situation, SME requires extensive ISD tool support. Such tools can be instantiated by Computer Aided Method Engineering (CAME) tools (Niknafs & Ramsin, 2008). There are also a few production-ready ISD tools available for process engineering in ISD, such as Eclipse Process Framework Composer (Haumer, 2007) and Rational Method Composer (Haumer, 2005) that employ the SPEM metamodel, as well as EssWork Practice Workshop (Jacobson, 2007) that uses the Essence metamodel. To our knowledge, production-ready tool support for the other metamodels is scarce.

Some DSM tools can also be considered as CAME tools, such as MetaEdit+ (Kelly et al., 2013). FIGURE 18 illustrates the relationships between CAME tools and ISD tools that implement process engineering metamodels. The illustration adopts the MOF-driven four-layer architecture of SPEM, analogous to the UML architecture discussed in the context of DSM tools in Section 2.6. In fact, SPEM also provides a UML profile for the specification of the situational methods. In the process engineering context, MOF similarly resides in the layer M3, enabling the construction of process engineering metamodels, such as SPEM in the layer M2. SPEM concepts are then available for the engineering of situational method specifications in the layer M1. The situational method specifications are constructed by instantiating SPEM-compliant method elements from the method base and arranging the elements into a workflow that conforms to the characteristics of the usage situation. Furthermore, the enactment of the situational method specification, i.e. the method in action, resides in the layer M0. M1 specifications can also be transformed into other artifacts, such as executable scripts for business process workflow engines (Cervera et al., 2012; Mallouli & Assar, 2013).

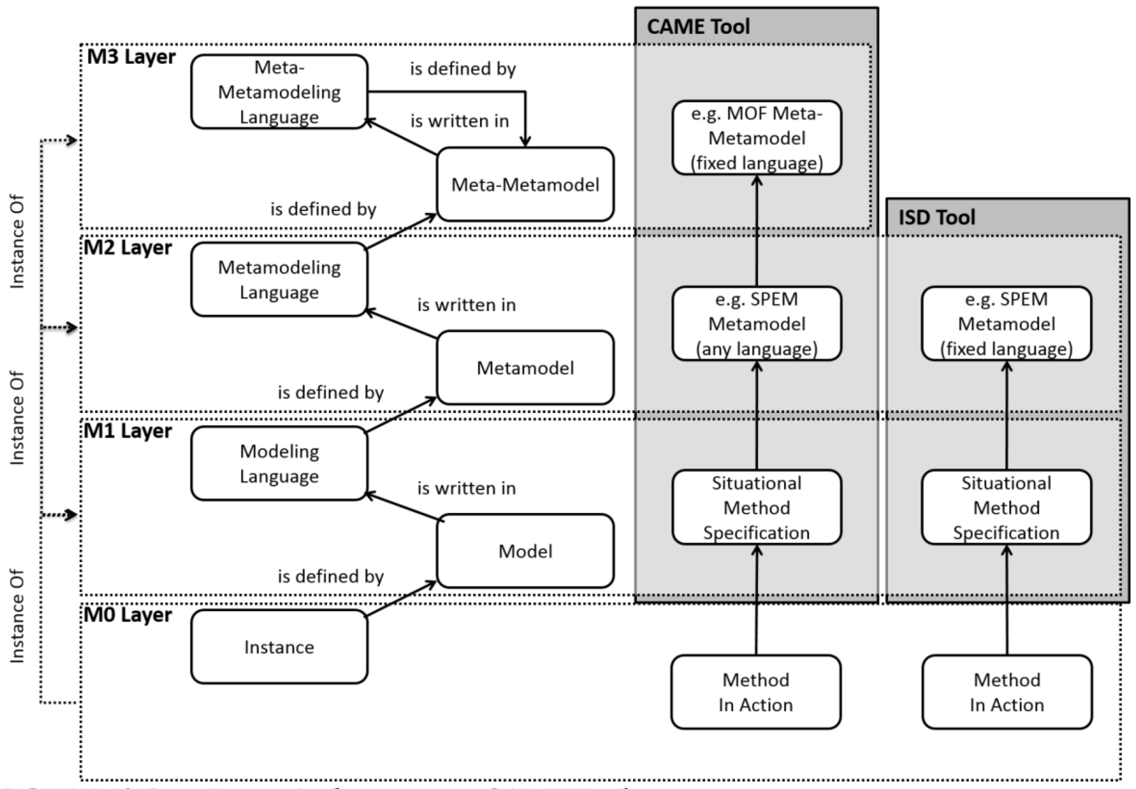


FIGURE 18 Four-Layer Architecture in CAME Tools

Our work adapts the SME practices from the ISD domain into the domain of DSM tool evaluation. A somewhat similar approach has been previously proposed by Kornyshova, Deneckère and Rolland (2011) by introducing the concept of method families in the domain of decision-making methods. Furthermore, Buchner et al. (2008) suggest that the principles, concepts, and techniques of SME are potentially useful also for domains other than ISD, such as IT and business process engineering, organizational change engineering, and enterprise modeling. The process engineering metamodels define various kinds of method elements, often called fragments, chunks, or process components, which differ in granularity. Metamodels also govern the interrelations of the elements (Henderson-Sellers et al., 2014, p. 28). The most common method elements in the current process engineering metamodels are Producer, WorkUnit, and WorkProduct, as illustrated in FIGURE 19. *Producer* represents a role or a stakeholder who performs the *WorkUnit*, i.e. an activity or a task to produce a *WorkProduct*, i.e. the outcome of a process (Henderson-Sellers et al., 2014, p. 28).

In the evaluation domain, the Producers are evaluators and other stakeholders, the WorkUnits are evaluation activities and tasks, and WorkProducts are e evaluation plans, reports, and pieces of software.

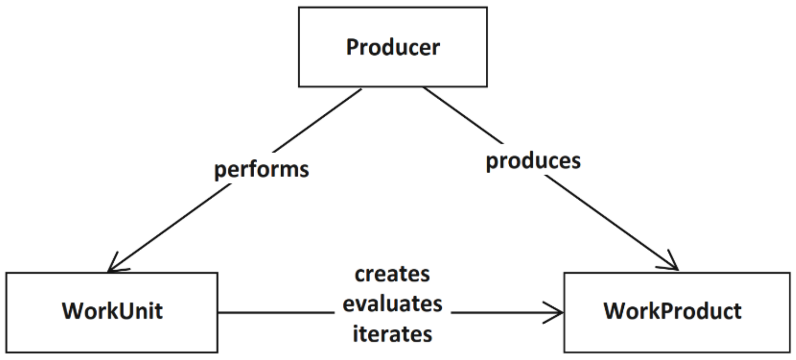


FIGURE 19 Common Method Elements and their Interrelations (Henderson-Sellers & Gonzales-Peres, 2010, p. 224)

The body of method elements structured in the following section is considered as a method base for situational evaluation methods for DSM tools. The method elements are available for instantiation in CAME tools with compatible meta-metamodels, such as MOF, and in ISD tools with compatible metamodels, such as OPF, SPEM and Essence (Elvesæter et al., 2013).

5.2 Method Base

In the following the method elements for the method base of Artifact II are constructed. WorkUnits, WorkProducts, and Producers are structured in UML.

5.2.1 WorkUnit Elements

The classification presented in section 4.3 is here utilized in the synthesis of the activities of the existing evaluation methods. FIGURE 20 represents the main activities of the method base of Artifact II as subconcepts of WorkUnit. Although Activity dimension is the central point of attention, Stakeholder and Context dimensions are to be taken into extensive consideration in the instantiation of the following activities.

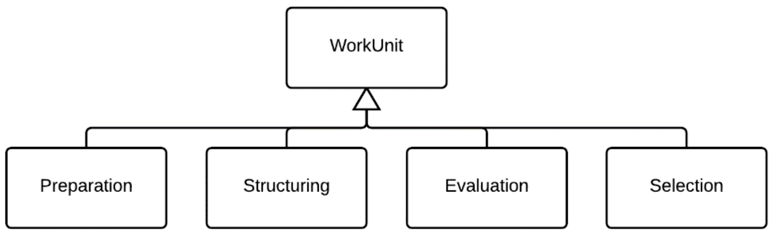


FIGURE 20 Main WorkUnits

The main Preparation WorkUnits, presented in FIGURE 21, are business case building (see Appendix 3), goal & policy setting, selection criteria planning, and evaluation approach selection. *Business case building* seeks justification to the initial investment of conducting a situational evaluation project as well as the acquisition and use of a DSM tool (Maes et al., 2014). The business benefits, risks, and costs are initially based on estimations made collaboratively by the management and expert stakeholders. Managerial and technical issues such as the introduction or upgrade of the DSM tools in the organization and the specifics of the target DSML(s) should be considered (Lukman & Mernik, 2008). During the conduct of an evaluation project, more concrete data is accumulated and the business case is updated accordingly. *Goal & policy setting* produces high-level goals for the evaluation project as well as a policy for the acquisition of the tools, addressing constraints such as budget, schedule and procedure for the implementation (ISO, 2008; Kruchten, 2004; Morera, 2002). *Project planning and control* addresses generic project management issues typical to all projects such as scoping, staffing, scheduling, and effort estimation, ultimately producing a project plan containing all the content produced by the Preparation activities (ISO, 2008; Morera, 2002). *Selection criteria planning* addresses the tentative selection criteria by which "go/no go" decisions can be justified in the Selection WorkUnits (ISO, 2008). *Evaluation approach selection* addresses the analysis of the Context variables according to the DESMET method (Kitchenham, 1996; Morera, 2002), producing the decision on a procedure and extent of the evaluation data collection to be conducted.

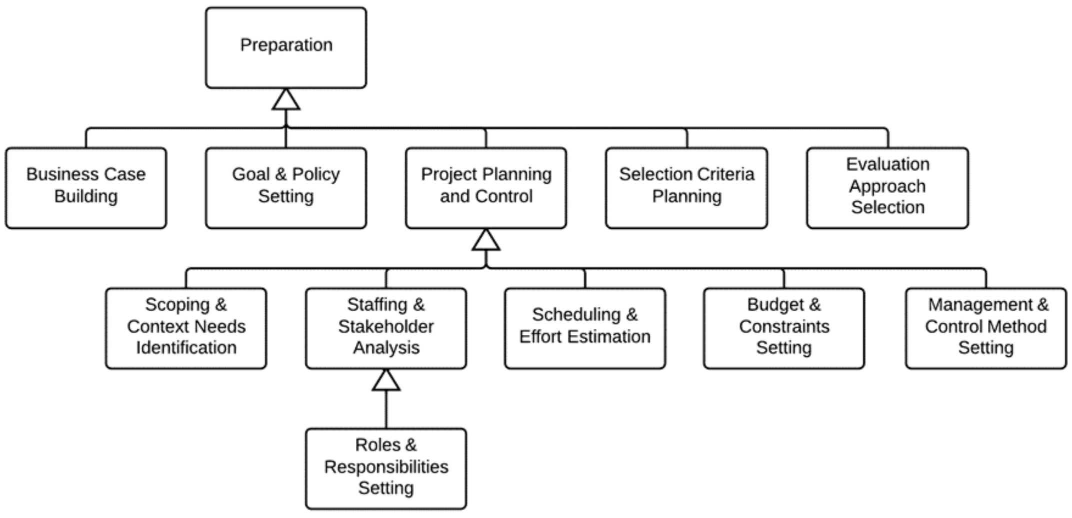
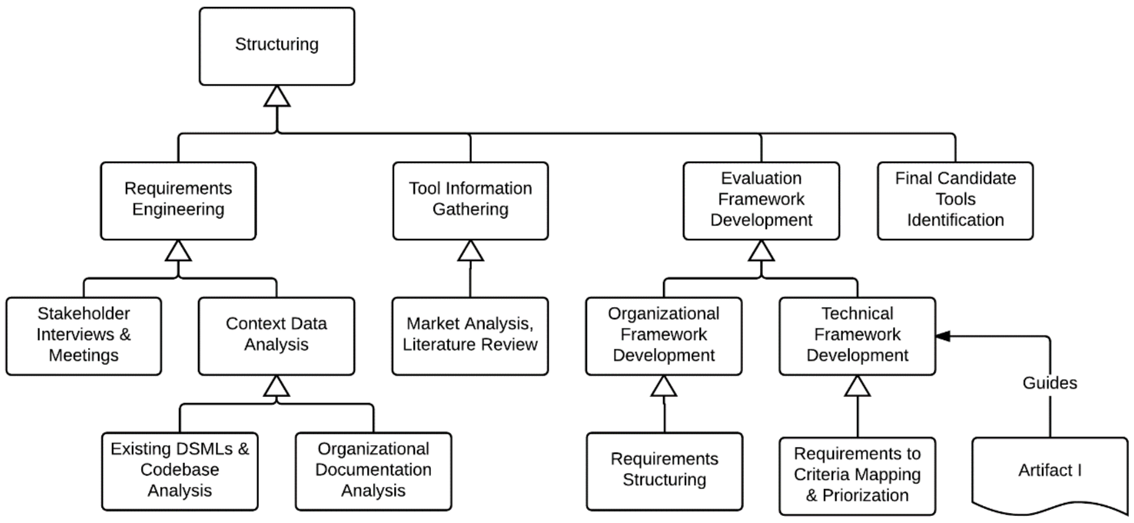

Preparation
Business Case Building
Goal & Policy Setting
Project Planning and Control
Selection Criteria Planning
Evaluation Approach Selection
Scoping & Context Needs Identification
Staffing & Stakeholder Analysis
Scheduling & Effort Estimation
Budget & Constraints Setting
Management & Control Method Setting
Roles & Responsibilities Setting

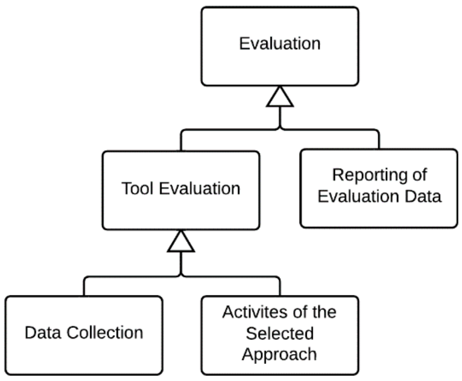
FIGURE 21 Main Preparation WorkUnits

The main Structuring WorkUnits, presented in FIGURE 22, are requirements engineering, tool information gathering, evaluation framework development, and final candidate tools identification. The specific content of the activities is dependent on the evaluation approach selected. *Requirements engineering* addresses the gathering of information from the context and stakeholders (ISO, 2008; Lukman & Mernik, 2008; Lundell & Lings, 2004b). Interviews and meetings with the stakeholders are included as well as analyses of e.g. organizational documents, existing DSMLs and target codebase (Hoisl et al., 2013; Lundell & Lings, 2003). *Tool information gathering* addresses the search of information regarding the identification of DSM tools and their characteristics from various sources, e.g. by conducting market analysis and by investigating existing reviews and available tool documentation (ISO, 2008; Kruchten, 2004; Lukman & Mernik, 2008; Wheeler, 2011; Morera, 2002). *Evaluation framework development* includes the construction of the organizational and technical evaluation frameworks along the guidelines of the 2G method (Lundell & Lings, 2003; Lings & Lundell, 2005). Full GT application can also be replaced by a more lightweight conceptual mapping. The previously identified requirements are prioritized and structured into the organizational framework and then linked to respective tool criteria of the technical framework. The criteria are then decomposed into metrics, representing each atomic tool sub-characteristic of interest. Artifact I, presented in Chapter 3, can be utilized as a reference for the technical framework during the mapping process. *Final candidate tools identification* addresses the exclusion and inclusion of the tool candidates to be evaluated, based on the information gathered (ISO, 2008; Wheeler, 2011; Morera, 2002).


Structuring
Requirements Engineering
Tool Information Gathering
Evaluation Framework Development
Final Candidate Tools Identification
Stakeholder Interviews & Meetings
Context Data Analysis
Market Analysis, Literature Review
Organizational Framework Development
Technical Framework Development
Guides
Existing DSMLs & Codebase Analysis
Organizational Documentation Analysis
Requirements Structuring
Requirements to Criteria Mapping & Priorization
Artifact I
FIGURE 22 Main Structuring WorkUnits with Artifact I as a Guideline

The main WorkUnits of Evaluation are tool evaluation and reporting of evaluation data, presented in FIGURE 23. The *tool evaluation* addresses the conduct of an evaluation according to the selected evaluation approach, which steers the managerial and evaluation procedures (ISO, 2008; Kruchten, 2004; Lukman & Mernik, 2008; Wheeler, 2011; Morera, 2002; Lundell & Lings, 2003). The evaluation is conducted by following the guidelines of the selected evaluation approach, such as benchmarking in which the relative performance of the final candidate tools is measured by using the evaluation criteria defined in the evaluation framework, which is constantly updated during the project as new requirements are accumulated. *Reporting of the evaluation data* includes the documentation of the data collected, in the detail agreed upon in the project plan

(ISO, 2008; Morera, 2002). The evaluation data should also be made available in a computer-readable format for the selection algorithm.


Evaluation
Tool Evaluation
Reporting of Evaluation Data
Data Collection
Activites of the Selected Approach
FIGURE 23 Main Evaluation WorkUnits

The main WorkUnits of Selection, illustrated in FIGURE 24, are selection criteria and algorithm setting, evaluation data analysis, tool selection, validation, and acquisition. *Selection criteria and algorithm setting* addresses issues such as refining the selection criteria based on the knowledge accumulated and selecting a decision-making algorithm (ISO, 2008; Morera, 2002). Such algorithms are provided by e.g. outranking methods, AHP, multi-attribute utility theory, weighting methods, fuzzy methods, and decision tree analysis (Kornyshova, 2011, p. 135; Morera, 2002). *Evaluation data analysis* includes the application of the selection algorithm to produce aggregated evaluation data about the tools (ISO, 2008; Morera, 2002). *Tool selection* addresses the making of the recommendation for the optimal tool, based on the results of the evaluation data analysis (ISO, 2008; Kruchten, 2004; Lukman & Mernik (2008); Morera, 2002). *Validation* is an effort to assure the validity of tool recommendations, carried out by matching the evaluation goals to the recommendation as part of a meeting in which the matter is discussed and analyzed by stakeholders, in effort to eliminate potential subjectivity of the decision-making and to collectively agree upon the selection (ISO, 2008; Lundell & Lings, 2004b; Morera, 2002). *Acquisition* includes the application of the acquisition policy, which in case of commercial software includes the negotiations of terms of tool licensing and exchange of money, and in case of OSS the potential negotiations of commercial support terms (Kruchten, 2004; Wheeler, 2014). All of the Selection WorkProducts are documented in a selection report.

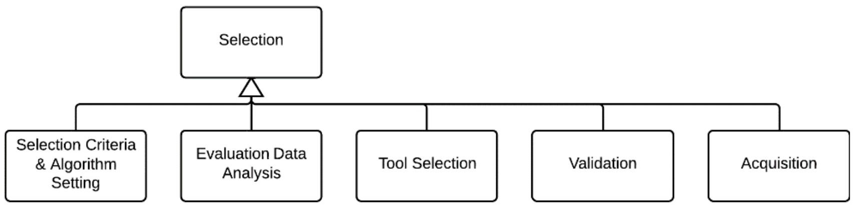
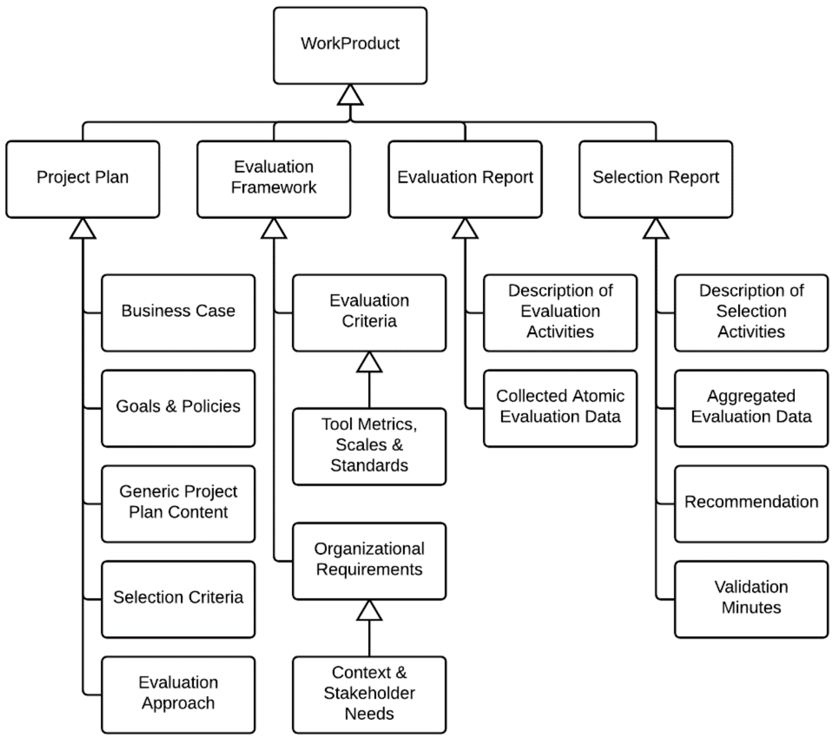


FIGURE 24 Main Selection WorkProducts

5.2.2 WorkProduct Elements

The main content produced by the WorkUnits is presented as subconcepts of WorkProduct in FIGURE 25. The main content produced by Preparation is the *project plan*, which contains the business case, evaluation goals and policies, generic project plan content, selection criteria, and the selected evaluation approach (Maes et al., 2014; ISO, 2008; Morera, 2002; Kitchenham, 1996). The main content produced by Structuring is the *evaluation framework*, which contains the organizational requirements and the technical evaluation criteria. (ISO, 2008; Lundell & Lings, 2003). The main content produced by Evaluation is the *evaluation report*, which includes the description of the conducted evaluation and the atomic evaluation data collected (ISO, 2008, Morera, 2002). The main content produced by Selection is the *selection report*, containing the description of the selection procedure, the aggregated evaluation data, the tool recommendation, and the minutes of the validation session.


WorkProduct
Project Plan
Evaluation Framework
Evaluation Report
Selection Report
Business Case
Goals & Policies
Generic Project Plan Content
Selection Criteria
Evaluation Approach
Evaluation Criteria
Tool Metrics, Scales & Standards
Organizational Requirements
Context & Stakeholder Needs
Description of Evaluation Activities
Collected Atomic Evaluation Data
Description of Selection Activities
Aggregated Evaluation Data
Recommendation
Validation Minutes
FIGURE 25 Main WorkProducts

5.2.3 Producer Elements

The stakeholders producing the content in the activities are presented as subconcepts of Producer in FIGURE 26. *Evaluator* is the main role responsible for the conduct of evaluation (ISO, 2008). Evaluator can be an employee of the context organization or a consultant from another company (Lundell & Lings, 2004b). *Manager* is responsible for project management and communication between the stakeholders (ISO, 2008). *Technical personnel*, such as DSML engineer, modeler, and developer are the stakeholders who are consulted on technical matters of the context (Cho, 2013, p. 30). *Domain expert* is consulted in the matters of the application context (Hoisl et al., 2013).

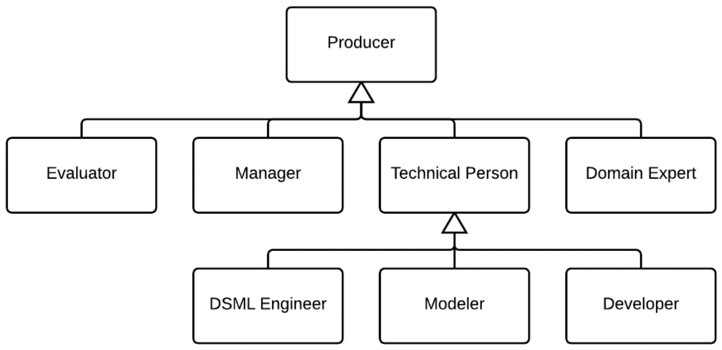


FIGURE 26 Main Producers

5.3 Situational Factors

SME for the evaluation of DSM tools is affected by various situational factors of the context, residing in the Stakeholders and Context dimensions discussed in Section 4.1. Hoppenbrouwers et al. (2011) suggest that the situational factors affect the context on the levels of an organization, a process, and an individual project. Bekkers et al. (2008) identified 27 situational factors in the SME for Software Product Management, and divided them into five categories: Business Units, Customers, Markets, Products and Stakeholders (TABLE 18).

TABLE 18 Situational Factors in Software Product Management (Bekkers et al., 2008)

Category	**Situational Factors**
Business Units	Development Philosophy, Size of Business Unit Team, Size of Development Team
Customers	Customer Loyalty, Customer Satisfaction, Customer Variability, Number of Customers, Number of End-Users, Type of Customers
Markets	Hosting Demands, Localization Demand, Market Growth, Market Size, Release Frequency, Sector, Standard Dominance, Variability of Feature Requests
Products	Application Age, Defects Per Year: Total, Defects Per Year: Serious, Development Platform Maturity, New Requirements Rate, Number of Products, Product Lifetime, Product Size, Product Tolerance, Software Platform
Stakeholders	Company Policy, Customer Involvement, Legislation, Partner Involvement

The high-level conceptual alignment of the situational factors (Bekkers et al., 2008), in respect to the dimensions of evaluation, discussed in Section 4.1 is outlined as follows: Stakeholders dimension contains Stakeholders and Customers, Internal Context contains Business units and Products, whereas External Context contains Markets. Thus, we argue that the dimensions affecting the evaluation activities also affect the SME of evaluation methods. Situational factors should always be considered in the selection of the method elements for a situational method. The method elements should also be attached with information that describes the situation in which they are useful.

Ågerfalk and Fitzgerald (2005) propose a concept of method rationale, which concerns the reasons, opinions, and arguments behind the engineering and adaptation of situational methods. SME is, to a large extent, a social conduct, thus it should consider the values, beliefs and understanding of the stakeholders, for it to be successful. Thus the rationality of the different aspects of a situational method should be regularly evaluated as a collective process, allowing the method to evolve and improve in regards to changing contextual requirements and stakeholder interaction. (Henderson-Sellers et al., 2014, p. 8.)

5.4 Construction Guidelines

The selection of a construction/tailoring approach for an evaluation method depends on the requirements and capabilities of an organization as well as situational factors in the Context and Stakeholders dimensions. Method construction can be conducted in a top-down or bottom-up fashion. In *the top-down approach* the top-level architecture of a method is modeled and refined downwards in effort to identify the required method elements. In *the bottom-up approach* the method elements are first identified and then composed upwards as a unified method architecture. According to Henderson-Sellers et al. (2014), the technical approaches to the construction of situational methods are:

- Assembly-Based
- Paradigm-Based
- Deontic Matrix
- Activity Diagrams
- Configuration-Based
- Extension-Based
- Ad-Hoc.

The Assembly-Based approach promotes the reuse of existing method elements stored in a repository (Ralyté & Rolland, 2001). *The Paradigm-based approach* utilizes the instantiation, abstraction or adaption of existing metamodels as the baseline method (Gupta & Prakash, 2001; Ralyte et al., 2003; Tolvanen, 1998). *The Deontic Matrix approach* uses collaboratively formed deontic matrices in method construction (Graham et al., 1997). *The Activity Diagrams approach* utilizes UML-style activity diagrams in method construction (Seidita et al. 2007). *The Configuration-Based* approach promotes the use of configuration templates and packages to adapt existing methods into a situation (Karlsson & Ågerfalk, 2009a). *The Extension-Based approach* uses patterns to enhance or reduce existing methods (Deneckére, 2001). Finally, *the Ad-Hoc approach* represents the construction of a method from scratch, e.g. for a new domain which is not yet supported by a specific method (Ralyté et al., 2004).

5.5 Artifact II Decomposition

Artifact II is composed of the method base, situational factors, and construction guidelines, as illustrated in FIGURE 27. Artifact II aims to give guidance to the construction and tailoring of situational evaluation methods for DSM tools. The main contribution of Artifact II is the method base for the domain of DSM tool evaluation, which is described in Section 5.2. The method base includes the following method elements: WorkUnits, WorkProducts, and Producers. The WorkUnits are divided into the high-level concepts of Preparation, Structuring, Evaluation, and Selection, representing the activities conducted in an evaluation effort. The WorkProducts represent the content produced in the WorkUnits, whereas the Producers represent the stakeholders that are involved with an evaluation effort. The method elements are based on the synthesis of the existing evaluation methods for ISD tools and DSM literature.

The situational factors are provided on a general level and aligned with the socio-technical dimensions of evaluation presented in Section 4.1. Stakeholders, Internal Context, and External Context are considered as the high-level concepts of situational factors. Stakeholders represents concepts such as Producers, Customers, and Partners, thus covering both internal and external Stakeholders. Internal Context is divided into Usage and Organization. Usage represents the context in which the direct users of the DSM tools reside. Organization represents the context in which the managers operate. External Context represents the factors that are usually beyond the control of the organization in which the evaluation is conducted, such as government policy and legislation, markets and competition, technological advancements, and other environmental pressures. In the terms of the situational factors identified by Bekkers et al. (2008), we outline that the evaluation efforts as well as the method construction and tailoring activities are probably affected by the situational factors that propagate from the interactions of the following technical, social, and economic phenomena: *Products are produced in Business Units as a result of the intertwined efforts by Stakeholders such as Producers and Partners, for the purpose of competitively supplying the demand that is created by the potential Customers in the Markets that are strategically selected by Stakeholders such as Managers, Directors, and Shareholders.*

Construction guidelines are discussed on a general level, introducing various approaches to the technical construction and tailoring of evaluation methods: Assembly-Based, Paradigm-Based, Deontic Matrix, Activity Diagrams, Configuration-Based, Extension-Based, and Ad-Hoc.

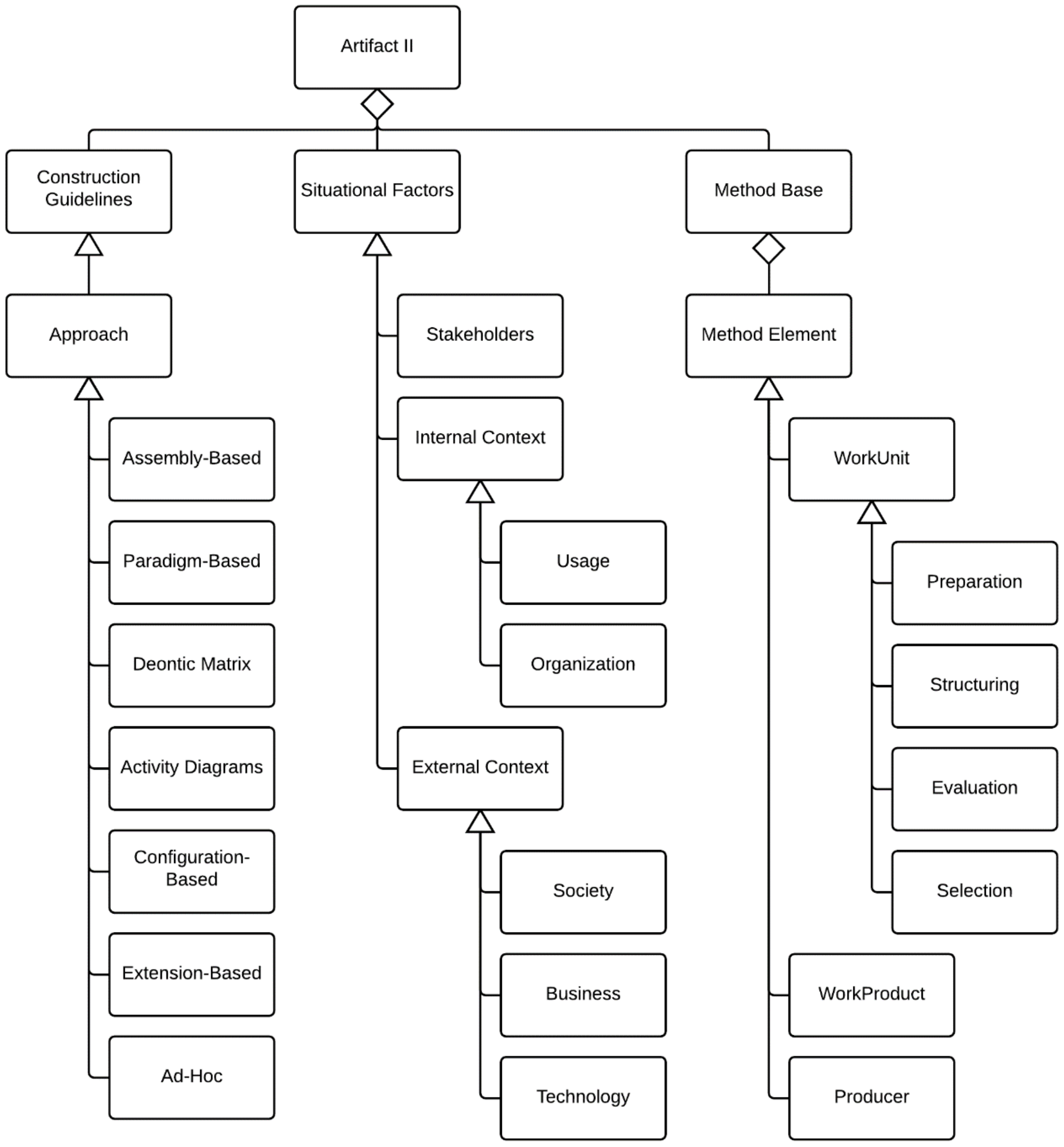


FIGURE 27 Decomposition of Artifact II

In the following section Artifact II is conceptually instantiated in an agile usage situation, based on Scrum method. The instantiation further demonstrates the domain shift from ISD to evaluation of DSM tools.

5.6 Conceptual Instantiation of Artifact II into an Agile Usage Situation

Situational Method Engineering (SME) in agile contexts have been reported by Karlsson (2013) and Karlsson and Ågerfalk (2009b). Agile ISD methods promote stakeholder communication and collaboration by fostering practices that rely heavily on socialization to access and share tacit knowledge within a project organization (Chau, Maurer & Melnik, 2003). Iterative activities and continuous integration of WorkProducts, coupled with a strong sense of collective ownership have a positive effect on stakeholder satisfaction and project success (Ferreira & Cohen, 2008). Such principles quite aptly address the important Stakeholder and Context dimensions of evaluation, promoting the consideration of the situational factors that are essential in the SME activities. Building on the widely adapted agile ISD method Scrum (Schwaber & Sutherland, 2014), presented in the following, the agile principles are applied in a conceptual instantiation of Artifact II into an agile usage situation. The high-level conceptualization is illustrated in FIGURE 28. It represents the dynamic part of Artifact II, whereas the method elements form the static part.

The situational evaluation effort is divided into three logical phases: pre-evaluation, evaluation, and post-evaluation. The evaluation effort is run in iterations called sprints, each including the selected method elements of the method base of Artifact II. Each sprint is initiated with a meeting in which all the stakeholders should be present. The length of the sprint can be any amount of days that is agreed upon, within the limit of a few weeks. Daily meetings should also be implemented among the core stakeholders to address the concrete issues faced in the everyday operations. Depending on whether the evaluation effort is distributed or not, an online tool and/or a physical board for project management should be utilized to monitor the progress and to promote process transparency and ownership as well as communication.

In the *pre-evaluation phase* an evaluation framework is iteratively developed by structuring and prioritizing requirements into the organizational framework. In the case of DSM tool requirements, they are linked into technical framework as criteria, using the Artifact I presented in Chapter 3 as guidance. The technical framework also includes the metrics, scales, standards, and measurement techniques for the criteria.

In the *evaluation phase* a set of requirements, in the order of the set priorities, are taken under consideration and goals are derived for the evaluation sprint. The goals are then transformed into concrete tasks to be taken and added into the sprint backlog list, steering the activities taken in the evaluation sprint. The evaluation phase can include any of the method elements of Artifact II. In the case of Evaluation WorkUnits, the tool candidates are measured by the criteria and guidance provided by the technical framework. As the outcome of the evaluation phase, a data increment is produced. In the case of Evaluation WorkUnits, the increment should always provide a complete piece of evaluation data of the tool candidates, in regards to the sprint goals.

In the *post-evaluation phase*, the data increments produced in the sprints are integrated into corresponding WorkProducts, e.g. an evaluation report or an update to the evaluation framework. As the outcome of the final sprint, tool selection activities produce a recommendation for a tool. At the end of the evaluation effort, a selection workshop is organized, in which the recommendation is validated, and the experiences from the effort are analyzed and documented.

The evaluation framework refined during the evaluation effort will be an asset in future evaluations.

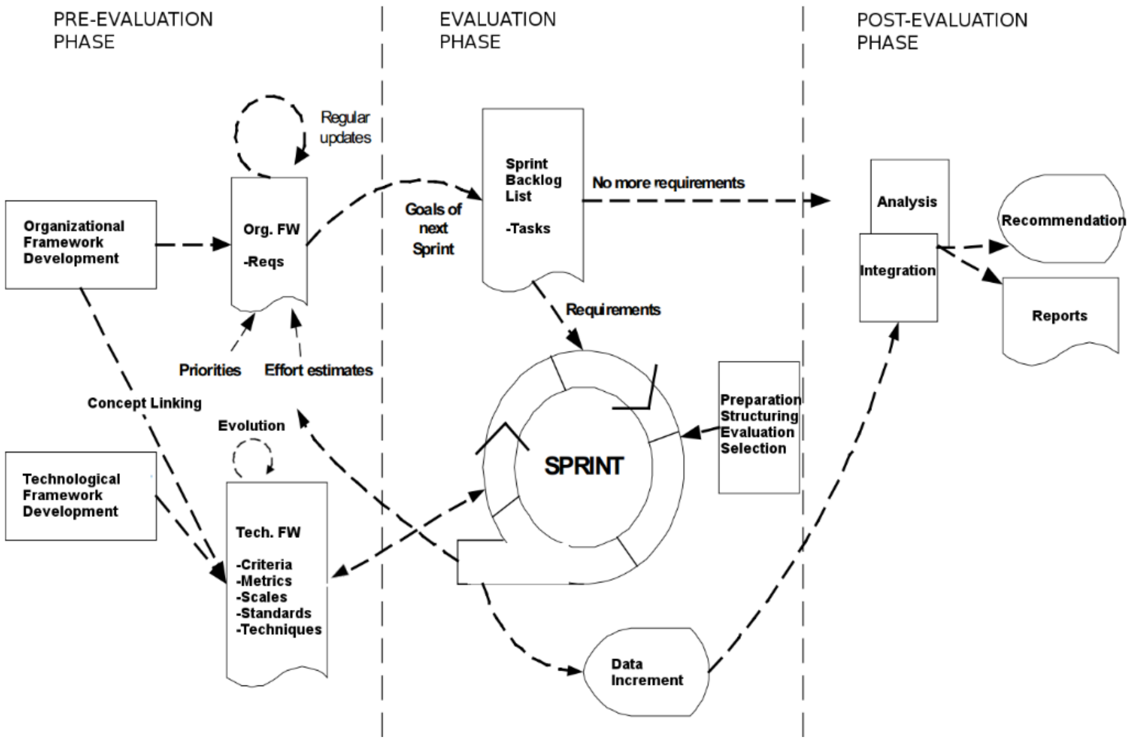


FIGURE 28 Conceptual Instantiation of Artifact II into an Agile Usage Situation, Adapted from Abrahamsson et al (2002, p. 28)

5.7 Summary

The chapter discussed the components of the baseline method for the engineering of situational evaluation methods for DSM tools, called Artifact II. First, the SME foundation, along with the related process engineering metamodels and CAME tools, were discussed. Second, a conceptual method base and its three types of method elements, WorkUnit, WorkProduct, and Producer, were discussed. The WorkUnits were derived from the synthesis of the activities of the existing evaluation methods. The WorkUnits were further divided into the high-level concepts of Preparation, Structuring, Evaluation, and Selection. Furthermore, the WorkProduct and Producer elements were constructed and presented. Third, situational factors were discussed on a general level, and aligned with the previously discussed socio-technical dimensions of evaluation. Fourth, the various approaches to method construction and tailoring, i.e. the construction guidelines, were presented. Fifth, the resulting high-level architecture of Artifact II was presented, based on the previous sections. Finally, Artifact II was conceptually instantiated in an agile usage situation.

6 EVALUATION OF META-METHOD

This chapter presents the empirical and conceptual evaluation of Artifacts I and II, discussed in the previous sections. The main objective is to seek validation for the artifacts by structuring them into a single design theory called Meta-Method and by comparing it against the similarly structured design theory of ISO 14102. In this evaluation process we use the evaluation criteria of progress for IS design theories.

First, the evaluation methodology is presented. The methodology is divided into four sections: Grounding Approach, Empirical Approach, Conceptual Approach, and Evaluation Criteria. Here, the current state of the discourse on the evaluation of DSR artifacts as well as the rationale behind the evaluation approaches chosen are first discussed. Then, the internal, external, and empirical grounding of the artifacts are delineated. Next, the empirical approach to evaluation, i.e. case study, is discussed, followed by the presentation of the conceptual approach, in which the relationships between design theories, instantiations, and humans are discussed and reflected to the phenomena of the case study. Additionally, design theory componentization and the evaluation criteria of progress for IS design theories: utility, internal consistency, external consistency, broad purpose and scope, simplicity, and fruitfulness of new research, are introduced.

Second, the design theories of Meta-Method and ISO 14102 are structured as design theory components, which is conducted to prepare them for the comparative evaluation with the criteria of progress.

Third, a case study in which Meta-Method was initially designed and empirically evaluated is discussed. The empirical evaluation is based on the analysis of the data collected during the case study, which included a company operating in the industry of professional mobile radio networks and devices. The empirical evaluation addresses the utility criterion.

Fourth, the conceptual evaluation is provided on the basis of the other criteria of progress. Finally, a summary of the chapter is presented.

6.1 Evaluation Methodology

Design Science Research (DSR) is still an emerging paradigm, as there is a consensus only on the broadest delineation: "DSR involves, in some way, learning through the act of building" (Kuechler & Vaishnavi, 2008). There are two major design views in DSR, a pragmatic technical artifact orientation and a theory-grounded user and meta-artifact focus (Miah et al., 2012). Nevertheless, both of them emphasize the importance of the evaluation of the artifacts. Furthermore, Iivari (2007) and Hevner (2007) suggest that the mark of a "good" DSR study lies in the rigorous evaluation of the artifacts propagated from the mixture of theoretical grounding and relevant engineering practice. While the debate continues in the DSR community, the artifacts discussed in this study are designed and evaluated according to the most adapted principles of DSR (Hevner et al., 2004; Peffers et al., 2007). The design perspective of the study is based on the rigorous theoretical body of previous work, from which the artifacts are derived from, as well as the practical adaptation of the artifacts, i.e. instantiating them in a context. The design objective is to solve a relevant real-world business problem faced in a company operating in the industry of professional mobile radio networks and devices. The first evaluation objective of the study is to empirically evaluate the utility of the instantiated artifacts in the case study context. Furthermore, a conceptual evaluation of the artifact constructs is conducted, in effort to evaluate the progress made during the study.

The initial design and empirical evaluation of the artifacts were conducted as an iterative case study, similarly to the cyclic approach introduced by Andersson and Runeson (2007). As design is an inherently iterative and incremental activity, the evaluation is considered an essential source of feedback for the design process (Hevner et al., 2004). One could argue that the approach taken was conformant to action research (Robson, 2002: Tolvanen, 1998) or action design research (Sein et al., 2011), as they are closely related (Runeson et al., 2012, p. 13; Iivari & Venable, 2009). In a broader sense, case study and action research are in many ways interrelated and share similarities with various aspects of DSR (Iivari, 2014). Nevertheless, the empirical evaluation of the artifacts is conducted on their instantiations, described in Section 6.3, by following the generally agreed principles of case study, which are also compatible with several other research approaches, such as action research (Runeson et al., 2012, 13). Artifacts are also conceptually evaluated as a single design theory. The objective of the conceptual evaluation is to derive a design theory by structuring the artifacts according to the design theory components (Gregor & Jones, 2007), and evaluate them in terms of criteria of progress for IS design theories (Aier & Fischer, 2011), by effectively comparing the artifacts against ISO 14102 (ISO, 2008). The conceptual evaluation is based on descriptive evaluation, i.e. informed argumentation, which is the "use [of] information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility" (Hevner et al., 2004, p. 86). Furthermore, the artifacts are grounded from three sources of knowledge (Goldkuhl, 1999). The design theory, derived from the intertwined combination of Artifact I and II, called A Meta-Method for the Engineering of Situational Evaluation Methods for Domain-Specific Modeling Tools, is hereinafter referred to as Meta-Method.

6.1.1 Grounding Approach

Goldkuhl (1999) proposes three classes for the grounding of action knowledge: internal, external, and empirical grounding (FIGURE 29). *Action knowledge* refers to the theories, strategies, and methods governing people's actions in social practices, such as evaluation efforts. The grounding of action knowledge means presenting arguments for the justification of such knowledge. *Internal grounding* means the grounding of the action knowledge in its own background knowledge. Internal grounding is often implicit and/or conceptual grounding, potentially consisting of the evaluation of knowledge cohesion, i.e. how the knowledge parts relate to each other and that there is a meaningful and logical consistency. *External grounding* refers to dealing with external warrants for the action knowledge, i.e. the established theories related to the action knowledge, such as ISO 14102 (ISO, 2008) and SME (Henderson-Sellers et al., 2014). *Empirical grounding* means observing and evaluating the application of the action knowledge, i.e. determining whether or not the action knowledge is successful in practice. In our work, the case study is considered as the empirical grounding of Meta-Method, whereas the theories from which Meta-Method is derived are considered as the external grounding. The conceptual evaluation carried out in Section 6.3 provides evidence for the internal grounding for Meta-Method. (Goldkuhl, 1999)

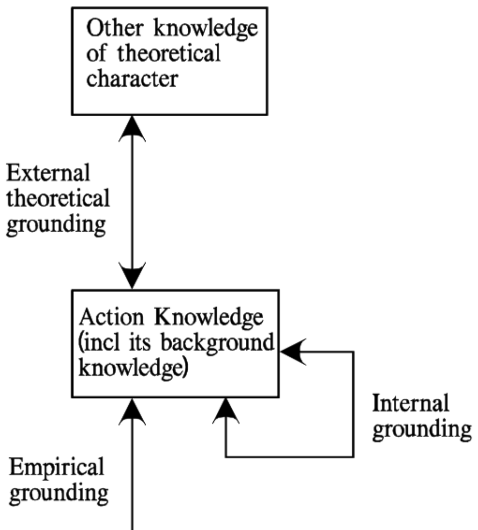

Other knowledge of theoretical character
External theoretical grounding
Action Knowledge (incl its background knowledge)
Internal grounding
Empirical grounding
Empirical observations

FIGURE 29 Three Classes of Grounding Action Knowledge (Goldkuhl, 1999, p. 8)

6.1.2 Empirical Approach

Case study is an established research approach for which distinct contributions are made by Robson (2002), Yin (2003), and Benbasat et al. (1987). All three agree on that case study is an empirical research approach, aimed at investigating contemporary phenomena in their context. Based on their definitions, Runeson et al. (2012, p. 12) derive the following definition for case study, aimed specifically for the field of software engineering:

> "… an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified".

There are also several other relevant research approaches, which would have been useful as alternative approaches in the effort of evaluating the artifacts empirically, such as survey, experiment, ethnographic study, longitudinal study, project monitoring, assertion, and field study (Runeson et al., 2012, p. 13). Furthermore, according to some researchers, action research would be the preferred approach in studies in which the researcher is involved with the change process under study, whereas case study would be purely observational (Runeson et al., 2012, p. 13). In this sense, our approach could be classified as action research, since the artifacts under study are instantiated, analyzed, and modified iteratively by us, both in and out of the context of the case study, strongly affecting the phenomenon studied. In addition, project monitoring shares similarities to our approach, as we collect and store operational data that accumulates during the application (Zelkowitz, 1997). The boundary between the types of research approaches is not, however, always clear, as Robson (2002, p. 185) denotes: "Many flexible design studies, although not explicitly labeled as such, can be usefully viewed as case studies".

The data collection in our case study was primarily conducted through documenting feedback sessions at the end of research iterations. During each iteration the artifacts were incrementally designed and then instantiated in the case study context. Artifact I was utilized as a guideline in the formulation of the technical evaluation framework for DSM tools. Artifact II was designed as the baseline method for the engineering of situational evaluation methods for DSM tools. The situational methods were enacted in the evaluation of DSM tools. Thus, there were three logical levels of activity in effect during the case study, namely research, method engineering, and evaluation, presented in TABLE 19. In the context of this thesis, the primary level of interest is research. TABLE 19 also presents the following high-level classifications: the domains in which the activities were conducted, the producers which conducted the activities, the methods that were utilized in the conduct of the activities, the WorkProducts that were produced in the activities as well as the types of phenomena that were addressed in the activities.

TABLE 19 Levels of Activities

Level	Domain	Producer	Method	WorkProduct	Phenomenon
Research	SME	Researcher	DSR	Meta-Method	Theory
Method Engineering	Evaluation Method	Method Engineer	SME	Situational Method	Theory / Instantiation
Evaluation	DSM Tools	Evaluator	Evaluation Method	Reports	Instantiation

The utility of the iterative instantiations of Meta-Method was evaluated in the feedback sessions by stakeholders, primarily in terms of the WorkProducts that were produced as the outcomes of the enacted situational methods. The participants of the feedback sessions were typically the researcher (we), project leader, chief engineer, developers, and modelers, i.e. the primary stakeholders of the evaluation effort. We composed and shared an agenda for the feedback sessions in advance to guide the discussion, and also to let the participants prepare for the sessions in advance. Furthermore, the feedback was captured in detailed meeting minutes, devised by the researcher and shared afterwards to the participants for further commenting. The data collection approach is similar to the unstructured interview technique. The feedback data is considered as the first degree research data, as it was inquired directly from the participants, and it steered the research activities of the iterations (Runeson et al., 2012, p. 48). The feedback data is analyzed as part of the case study description in Section 6.3.

6.1.3 Conceptual Approach

There is an active school of thought promoting that DSR should produce design theories by combining proven theories with goals of actors in a business context (Venable, 2006; Gregor & Jones, 2007; Walls et al., 2004). Gregor and Jones (2007) propose that design theories can have as a primary design goal either a method or a product. A design theory can also be instantiated, resulting in some form of physical existence in the real world. FIGURE 30 illustrates the three phenomena of interest in DSR, as proposed by Gregor and Jones (2007): instantiations, theories, and human subjective understanding of artifacts. *Instantiations* are artifacts that have physical existence in the real world, such as hardware and software, or the series of physical actions taken that lead to the existence of them, i.e. method in action (Lundell & Lings, 2004a). *Theories* are artifacts that do not have a physical existence, except in the sense that they are communicated in words, images, diagrams or some other means of representation. These types of artifacts are e.g. constructs, models, and methods, such as the artifacts presented in this thesis. *Human subjective understanding of artifacts* represents the human component in relation to instantiations and theories. Humans conceptualize and describe artifacts in abstract terms as theories as well as use the theories as guidance to build the instantiations. Theories are also used to understand the material artifacts utilized in the real world. On the other hand, theories can be extracted by the means of observing and analyzing the instantiated artifacts. (Gregor & Jones, 2007.)

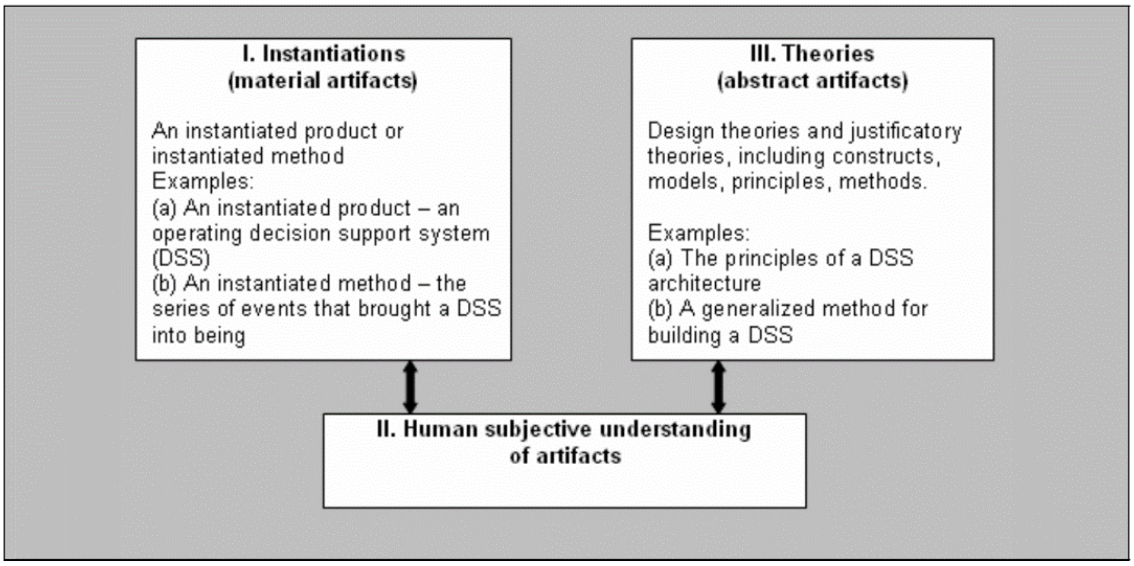

I. Instantiations
(material artifacts)
An instantiated product or instantiated method
Examples:
(a) An instantiated product – an operating decision support system (DSS)
(b) An instantiated method – the series of events that brought a DSS into being
III. Theories
(abstract artifacts)
Design theories and justificatory theories, including constructs, models, principles, methods.
Examples:
(a) The principles of a DSS architecture
(b) A generalized method for building a DSS
II. Human subjective understanding of artifacts

FIGURE 30 Relationships Among IS/IT Artifacts (Gregor & Jones, 2007, p. 321)

In the terms of the levels of activities performed in our work, we deal with phenomena that are classified as theories or instantiations (TABLE 19). In addition, our subjective understanding of the phenomena affects the instantiations of the theories as well as the theorizing of the observed instantiations. The conceptual evaluation focuses mainly on the abstract artifacts whereas the empirical evaluation concentrates solely on the material artifacts. The research activities produce Meta-Method, which is abstract. The method engineering activities instantiate Meta-Method into the case study context, producing situational method specifications, which in itself are classified as abstract artifacts, whereas the method engineering in action that leads to the situational method specifications is the instantiation of Meta-Method and the theories utilized in its conception. The evaluation activities are instantiations of the situational method, resulting in as the evaluation method in action. Furthermore, it can be argued that the reports that are produced by the evaluation activities are abstract artifacts that guide the further instantiations of further activities in action, such as the acquisition and implementation of a selected DSM tool.

For the purpose of preparing the artifacts for the evaluation of progress, Meta-Method is structured as a composition of design theory components (Gregor & Jones, 2007). A design theory is composed of eight components: purpose and scope, constructs, principle of form and function, artifact mutability, testable propositions, justificatory knowledge, principles of implementation, and expository instantiation. The components are defined in TABLE 20. The componentization of the design theories enables the categorization, comparison, and extension of the design theories in respect to other design theories (Gregor & Jones, 2007). In the evaluation, the design theory components of Meta-Method are compared to those of ISO 14102 (ISO, 2008), in the effort of determining the potential progress achieved.

TABLE 20 Design Theory Components (Gregor & Jones, 2007, p. 322)

Component	Description
Purpose and Scope	"What the system is for," the set of meta-requirements or goals that specifies the type of artifact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory.
Constructs	Representations of the entities of interest in the theory.
Principle of Form and Function	The abstract "blueprint" or architecture that describes an IS artifact, either product or method/intervention.
Artifact Mutability	The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory.
Testable Propositions	Truth statements about the design theory.
Justificatory Knowledge	The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories).
Principles of Implementation	A description of processes for implementing the theory (either product or method) in specific contexts.
Expository Instantiation	A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing.

6.1.4 Evaluation Criteria

Aier and Fischer (2011) adapt the five high-level criteria for scientific progress by Kuhn (1977) into the domain of IS design theories. Ultimately, they propose six evaluation criteria for IS design theories, by which the progress of one theory in comparison to another can be defined. The evaluation criteria are: utility, internal consistency, external consistency, broad purpose and scope, simplicity, and fruitfulness of new research. The criteria are defined in TABLE 21. They base the criteria on the "ceteris paribus clause", i.e. "a design theory *A* can only be called "better" than a design theory *B* if *A* fulfils at least one criterion better than *B*, whereby the fulfillment of all other criteria remains equal" (Aier & Fischer, 2011, p. 170). By employing the criteria of progress we evaluate Meta-Method in comparison to ISO 14102, which also was the initial baseline method upon which Meta-Method was designed. Ultimately, we aim to determine the degree of progress achieved by our work, especially in the area of providing method support for the engineering of situational evaluation methods for DSM tools, by contrasting it to ISO 14102, which is a general guideline for the evaluation of ISD tools, intended to be tailored according to organizational needs. In the empirical evaluation we determine the degree of progress by the criterion of utility whereas the other criteria are determined in the conceptual evaluation.

TABLE 21 Scientific Progress Criteria for Design Theories (Aier & Fischer, 2011, p. 158)

Criterion	Description
Utility	• The utility of a design theory is the artifact's ability to fulfill its purpose if the purpose itself is useful. The purpose of an artifact is only useful if it is relevant for business • The purpose of an artifact is concretized by testable propositions. They help to prove that the artifact fulfills its purpose • Three forms of utility can be differentiated: gross utility (absolute output), net utility (difference between output and input), and efficiency (output divided by input)
Internal Consistency	• Each element of a design theory should be consistent with itself • A consistent system of constructs is the common basis for all design theory elements. All constructs unstructured should be defined concisely. In the interests of consistent terminology, it is important that homonyms, including subtle homonyms, and synonyms are avoided • Form and function of the artifact, artifact mutability, principles of artifact implementation, and testable propositions directly depend on scope and purpose • Testable propositions refer to form and function of the artifact, artifact mutability, and its principles of implementation • Justificatory knowledge should justify form and function of the artifact, artifact mutability, and its principles of implementation
External Consistency	• Justificatory knowledge should be consistent with the knowledge base. • Consistency with a selected part of the knowledge base, i.e. with justificatory knowledge (or with kernel theories), is covered by internal consistency. In addition, external consistency refers to a sound justification of the choice of justificatory knowledge (or of kernel theories); moreover, the relationship to parts of the knowledge base that contradict design decisions should be explicated • Constructs should be consistent with constructs commonly used • Sometimes, design theories are innovative simply because they contradict commonly accepted assumptions
Broad Purpose and Scope	• Scope and purpose of an artifact should be broad • If one design theory A covers a purpose and scope that has previously been covered by more than one design theory $B_1,\ldots,B_n$, ≥ n 2, A is ceteris paribus progressive in comparison to $B_1,\ldots,B_n$
Simplicity	• Design theories should be simple in order to be easily understandable and manageable • Simple artifacts will often cost less when used. This aspect is already covered by two notions of utility: net utility and efficiency
Fruitfulness of New Research	• Design theories should disclose new phenomena or previously unnoted relationships among already known phenomena • They should initiate/stimulate further research activities

6.2 Design Theorization

In order to evaluate the progress made in our work in the domain of providing method support for the engineering of situational evaluation methods for DSM tools, commensurate design theories are required (Gregor and Jones, 2007). Thus, ISO 14102 is componentized as a design theory in TABLE 22 and Meta-Method is similarly structured in TABLE 23. As there is no methodical support available for the componentization process, the structuring of the components are primarily based on the subjective interpretation of the standard document by ISO (2008) as well as the analysis and combination of Artifacts I and II. Although we are experienced in the analysis, instantiation and tailoring of ISO 14102 in a practical setting, we don't claim to possess exhaustive knowledge of all the aspects of the method, which is reflected in the componentization.

TABLE 22 The Design Theory Components of ISO 14102

Component	**Description**
Purpose and Scope	The international standard gives guidelines for the evaluation and selection of ISD tools, covering a partial or full portion of the software engineering life cycle. It establishes processes and activities to be applied for the evaluation of ISD tools and selecting the most appropriate ISD tools from several candidates. These processes are generic, and organizations must tailor them to meet organizational needs. The ISD tool evaluation and selection processes should be viewed in the larger context of the organization's technology adoption process.
Constructs	Phases: preparation, structuring, evaluation, selection. ISD tool characteristics: categories, atomic sub-characteristics, description.
Principle of Form and Function	An abstract evaluation method is provided that consists of four phases along with related tasks and expected outcomes. In addition, a list of generic ISD tool characteristics is provided. There's no method support for the instantiation of the method in a situational context, nor for the formulation of the evaluation criteria for a specific type of a tool (Lundell & Lings, 2002). Additional guidance for ISD tool adoption processes, following the tool selection, is provided in ISO TR 471:2007.
Artifact Mutability	The method is generic and intended to be tailored to the context. The static generic method doesn't provide method support for the tailoring.
Testable Propositions	The method can be adapted to various situational contexts in which ISD tools are evaluated by ad-hoc instantiation practices.
Justificatory Knowledge	The method is a result of decades of standardization work and is the current standard for ISD tool evaluation, synthesizing and building on the previous standard IEEE 1209:1992. Various reports of application have been reported on real evaluation cases.
Principles of Implementation	The implementation of the method requires a facilitator that is experienced in ISD tool evaluation.
Expository Instantiation	Instantiation have been reported by Hilera and Martinez (1999) as well as Lundell and Lings (2002).

TABLE 23 The Design Theory Components of Meta-Method

Component	Description
Purpose and Scope	Method support is required for the engineering of situational evaluation methods for DSM tools, as the previous evaluation methods are generic and/or limited in various ways, e.g. they are designed for generic ISD tools, lack the guidance of how to adapt the methods in a situational context, and/or don't provide the guidance on which evaluation criteria should be used for DSM tools in the situational methods.
Constructs	Meta-Method: method base, situational factors, constructional guidelines. Method base: WorkUnits, WorkProducts, Producers. Situational factors: stakeholders, internal context, external context. Constructional guidelines: assembly-based, paradigm-based, deontic matrix, activity diagrams, configuration-based, extension-based, ad-hoc Evaluation criteria checklist: category, criterion, type, range, example.
Principle of Form and Function	A conceptual baseline method is provided, to be instantiated in SME that specifies situational evaluation methods, which are instantiated in the evaluation efforts, producing reports for decision-making. The evaluation criteria checklist is used as guidance in the situational methods for the formulation of the technical evaluation frameworks for DSM tools.
Artifact Mutability	Meta-Method is created on the basis of SME, which promotes the development and evolution of the constructs and adaptability to the situational context. The constructs are mainly generic, which promotes the extension and specialization of them according to characteristics of the situational context.
Testable Propositions	(1) Meta-Method can successfully support the engineering of situational evaluation methods for DSM tools in various contexts in which DSM tools are evaluated, due to the use of SME principles. (2) The application of the evaluation criteria checklist can produce meaningful evaluation results that should be commensurate between different evaluation effort instances. (3) The application of the situational evaluation methods engineered with Meta-Method can produce successful DSM tool recommendations, at least when measured in gross utility
Justificatory Knowledge	Meta-Method is derived from various previous evaluation methods as a synthesis and specialized as a baseline method for the engineering of situational evaluation methods for DSM tools, utilizing proven SME principles. The evaluation criteria checklist is derived from eight previous studies that discussed the evaluation of DSM tools. Furthermore, an extensive literature review and practical experiences in DSM guided the design of Meta-Method.
Principles of Implementation	The implementation of Meta-Method requires a facilitator that is experienced in DSM, DSM tools, evaluation methods, and SME practices. The evaluation criteria checklist provides guidance on which criteria to evaluate in the situational evaluation methods.
Expository Instantiation	Description of the instantiation of Meta-Method in a case study is provided in Section 6.3.

ISO 14102 is a highly generalized method, addressing the evaluation of all types of ISD tools, intended to be tailored according to organizational needs. The main constructs of ISO 14102 are the phases, tasks and outcomes of the generic evaluation process, in addition to the list of ISD tool characteristics, which consists of the categories, sub-characteristics, and descriptions of the characteristics. The constructs are intended to be adapted to the organizational needs, but no method support for the adaptation is provided. As is typical for established international standards, they are developed over a long period of time and have been instantiated numerous times.

Meta-Method is a specialized method, focusing on the engineering of situational evaluation methods for DSM tools. It was developed for this specific need and instantiated in the case study context described in the Section 6.3. The main constructs of the method are the method base, situational factors, and constructional guidelines. Whereas the constructional guidelines are mainly generic SME theories, the method base includes method elements for the evaluation of DSM tools. Furthermore, the situational factors of SME are aligned with the multi-disciplinary dimensions of evaluation. Additionally, the evaluation criteria checklist for DSM tools is provided, to be used as a guideline in the situational methods. Meta-Method is intended to be instantiated in the engineering of situational evaluation methods for DSM tools, using SME practices. The adoption of the proven SME principles enables the construction and tailoring of the situational evaluation methods.

In the following sections the delineated design theories are evaluated from the perspective of potential progress achieved in the efforts of developing the generic method support provided by ISO 14102 towards the specialized method support that is provided by Meta-Method. The premise of comparing a generic method against a specialized method is not optimal, but the lack of existing counterparts of Meta-Method dictate the compromise. Furthermore, ISO 14102 was utilized as a baseline method for the construction of Meta-Method, which promotes the commensurability of the theories as well as provides a meaningful perspective to the evaluation of progress. The evaluation of progress is conducted in the domain of providing method support for the engineering of situational evaluation methods for DSM tools. Thus, the assumptions are not generalizable for other domains.

6.3 Empirical Evaluation

In this section the empirical evaluation of Meta-Method is presented. Meta-Method combines Artifacts I and II. Meta-Method was constructed in an industrial case study, in which the empirical evaluation of its utility was also conducted. The iterative evaluation approach presented in Section 5.6 was applied in the evaluation component of the case study. First, the context of the case study is described. Second, the iterations carried out during the case study are discussed, especially from the viewpoint of the evolution of Meta-Method. Third, the outcomes of the case study are analyzed. Finally, the utility of Meta-Method is evaluated on the basis of the data collected.

6.3.1 Context

The case study was initiated in the summer of 2010, as we were hired to conduct an evaluation of DSM tools for a multinational technology company, operating in the industry of professional mobile radio networks and devices. Our role was to act as an external researcher to the company. The evaluation was conducted in Jyväskylä, Finland. The main business need of the company was to evaluate the state-of-the-art of DSM tools in order to select the optimal tooling for their major re-engineering effort of a software product line for professional mobile radio devices. The devices are conformant to ETSI TETRA standard, intended for the use of public authorities such as the police and the fire and rescue departments. The re-engineering effort was related to the upgrade of the software platform for which the professional applications were developed using DSM. As the re-engineering effort would be a significant and time consuming investment, the most suitable DSM tool for the effort had to be resolved, in order to reduce the risk of failure and to improve the quality and productivity of ISD processes in the long run.

From the perspective of the DSR process, the primary interest of the company was the demonstration of Meta-Method, i.e. the evaluation of DSM tools in the situational context, leading to the production of reports to support the decision-making in the company. A secondary objective was to update the stakeholders' knowledge on the current trends and practices related to DSM tools. The following academic stakeholders were involved in the evaluation effort: project leader and researcher (we). The internal stakeholders of the company were chief engineer, DSML engineer, developer, and modeler. Initially, the chief engineer had built a business case for the re-engineering effort, eliciting the justification for the investment. Going forward with the optimal tooling was critical for the re-engineering effort, but there was no internal resources available at the time for conducting a systematic evaluation of DSM tools. Thus, such an evaluation was initiated in collaboration with the local university. Most of the stakeholders were working part time in the effort. Common project management practices and tools, such as regular feedback meetings and communication and knowledge sharing platforms were employed.

FIGURE 31 illustrates the operational framework for the case study. The three levels of activities (cf. TABLE 19), namely research, method engineering, and evaluation were intertwined during the iterative research effort. The respective roles conducting the activities, namely researcher, method engineer, and evaluator, were mostly assumed by us as a single physical person. First, during a typical iteration, the research level analyzed the feedback received from the previous iteration and then produced the next increment of Meta-Method, based on the feedback data. Next, Meta-Method was instantiated in the method engineering level, which produced a situational method for the iteration. Then, the situational method was enacted in the evaluation level, producing an increment of the reports related to the evaluation efforts. Finally, the activities commenced during the iteration were reported and evaluated in the feedback sessions, in which the feedback data was collected and analyzed for the planning of the next iteration of activities.

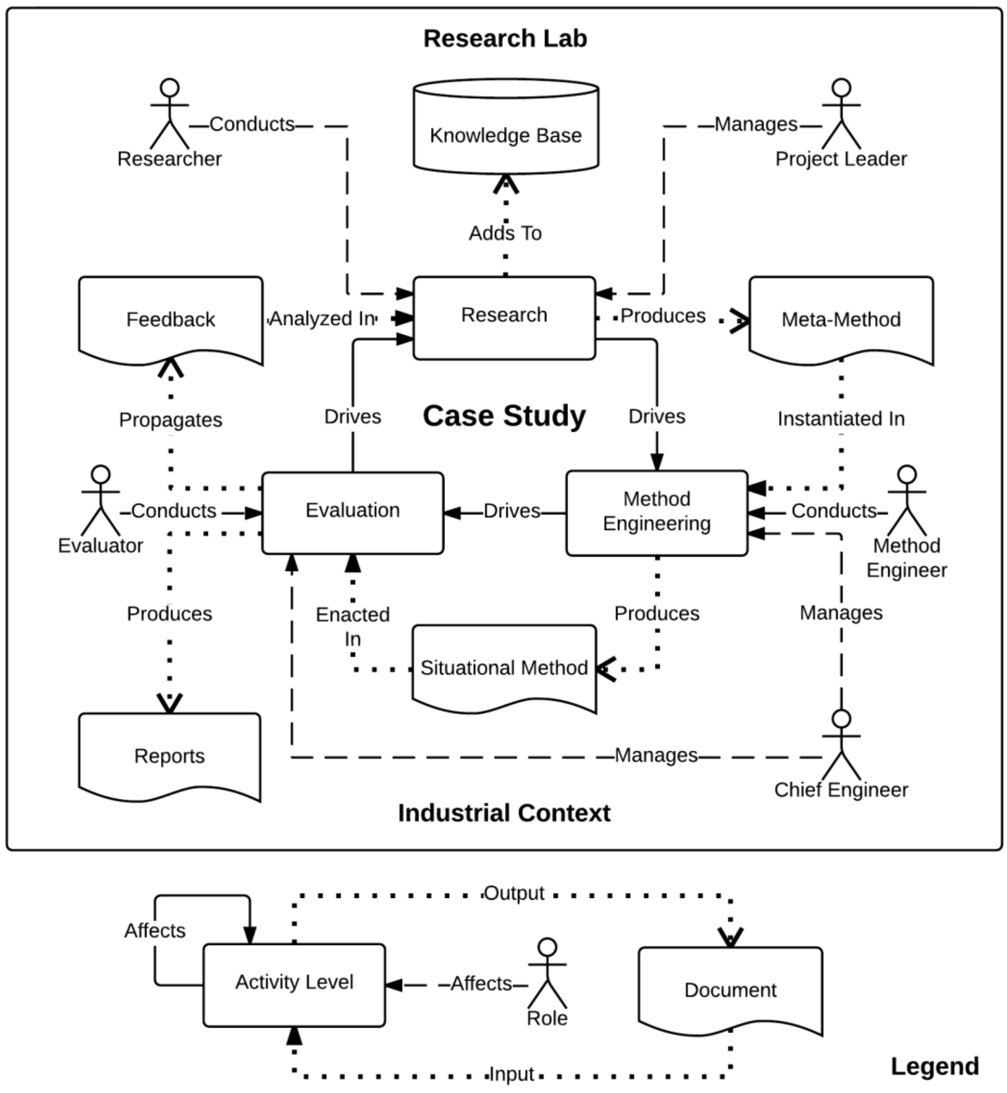


FIGURE 31 Operational Framework of the Case Study

The primary unit of analysis in the case study is Meta-Method and more specifically its evolution throughout the evaluation effort, which is analyzed in the qualitative analysis of the feedback data. This is relevant for research as it provides case-specific evidence from the business perspective as to which method elements are useful at specific points in time of an evaluation effort and which situational factors affect them. The main objective of the case study is to validate Meta-Method in terms of utility, as described in Section 6.1.4. As the initial baseline method for the evaluation effort, ISO 14102 was utilized. The highly generic nature of ISO 14102 quickly evoked us to seek alternative measures in the effort of better conducting an evaluation of DSM tools in the situational context. Thus, the construction of Meta-Method was required.

6.3.2 Iterations

The preparation of the evaluation effort was initiated in collaboration between the chief engineer and the project leader, focusing primarily on the generic project management issues and high-level goal setting. As we were included in the project team, the preparation activities continued by a kick-off meeting in May 25th 2010, in which all available stakeholders were present. An iterative method was decided to be implemented in the effort. The iterations, the respective key objectives and the resulting key activities conducted, along with the dates for the feedback sessions are presented in TABLE 24. In the following, the iterations conducted in the evaluation effort are described in addition to the retrospective analysis of the respective feedback data, in the effort of delineating the utility of Meta-Method.

TABLE 24 Iterations in the Case Study

#	**Objectives for Iteration**	**Research**	**Method Engineering**	**Evaluation**	**Feedback Session**
	Primary Goal	Key Activity	Key Activity	Key Activity	Date
1	Project Initiation	Initial Sketching of Meta-Method	Adaptation of ISO 14102 as the Initial Method	Preparation	09.06.2010
2	Requirements Engineering	Design of Meta-Method	Construction of the Evaluation Framework	Structuring	18.06. & 23.06.
3	Meta-Method Construction	Design of Meta-Method	Construction of the Situational Method	Structuring	11.08.
4	Tool Familiarization	Design of Meta-Method	Tailoring of the Situational Method	Structuring	25.08.
5	Tool Evaluation, 1st Set	Design of Meta-Method	Tailoring of the Situational Method	Evaluation	09.09.
6	Tool Evaluation, 2nd Set	Design of Meta-Method	Tailoring of the Situational Method	Evaluation	11.11.
7	Tool Selection	Design of Meta-Method	Tailoring of the Situational Method	Selection	18.11

In the *first iteration* of the evaluation effort we focused primarily on knowledge search and familiarization of the problem domain issues, such as DSM, DSM tools, and evaluation methods. Also, research methods and reporting standards were studied. Furthermore, existing literature on the evaluation of DSM tools were reviewed. ISO 14102 was adapted as the initial baseline method, as it was assumed that it provides the method support required. High-level goals were discussed in more specific terms. For example, it was established that DSM tools licensed as open source were the ideal candidates, albeit the commercial rivals were to be evaluated in equal terms. The main WorkProduct of the iteration was a refined version of the project plan. The feedback data is distinctly showing that it was the early days of the evaluation effort as the practices were still unestablished and they were being addressed in a more specific manner.

Literature findings were discussed and reflected to the practical experiences and situational factors of the client. The activities were mainly related to the preparation WorkUnits of Meta-Method. Situational factors such as Business Units and Customers were primarily analyzed.

In the *second iteration*, the activities were more focused on the concrete structuring of the conduct of the evaluation effort. More specific goals were determined, such as the desired structure for the evaluation report and the client's long-term objectives of improving productivity and quality of the ISD processes by optimal DSM tool selection. The most significant activities were related to the evaluation framework development, as the first structured set of evaluation criteria, provided mainly by ISO 14102, were discussed. It was established that the general evaluation criteria for ISD tools were not sufficient for the evaluation of DSM tools. To address the issue, the client's requirements specific to DSM tool architecture, DSML specification, and DSML application were elicited. After another literature review, effectively forming the base for Artifact I, the requirements were mapped to the evaluation criteria for DSM tools, which was the inception of the evaluation framework. A subset of the evaluation framework is presented in TABLE 25, in which the Priority and Requirement columns represent the organizational framework, whereas the concept linking to the technical framework is represented by the Criterion column.

TABLE 25 A Subset of the Evaluation Framework

Priority	**Requirement**	**Criterion**
1st	The tool must have multi-platform support, or at least Windows and Linux support.	Platform Support
1st	The tool must have the capability of generating any programming language.	Transformation Output Language
1st	The tool must provide facilities for modeling language specification, thus UML based MDA tools are not sufficient.	Metamodel Syntax Specification
2nd	The tool licensing model is preferably open source, but it is not mandatory.	Licensing Model
2nd	The tool must provide versatile integration mechanisms to the development environment.	Integration
2nd	The tool must provide a good user experience.	Usability

2G method was found helpful in the development of the evaluation framework. During the second iteration, also an initial set of tool candidates were selected for evaluation, as a result of market analysis and literature review. Furthermore, a demonstrative DSM solution that was to be utilized in the comparison of the tools in the practical evaluation was specified. The target code to be generated was a mobile Java application, compliant with the upcoming target platform. In the analysis of the feedback data, it is evident that ISO 14102 didn't provide the required method support for the specific tasks in the case in question, and it was required to look for guidance beyond the standard. The WorkUnits conducted were mainly related to structuring. The main WorkProducts of the iteration were an evaluation framework and a list of candidate tools. Situational factors of Business Units, Markets and Products were extensively investigated.

During the *third iteration* the construction of Meta-Method was effectively started. Literature was extensively reviewed and method elements identified in the effort to find the proper method support for the required tasks at hand. ISO 14102 was assumed as the baseline method upon which additional and more specified method elements were added, in the efforts to optimally conduct the evaluation of DSM tools in the situational context. As the conceptual method base was starting to take its form, SME practices were conducted to instantiate the method elements to the situational context. The DESMET method (Kitchenham, 1996) was utilized to select qualitative case study as the evaluation approach. The evaluation also included characteristics of qualitative screening and benchmarking. The feedback data analysis indicates that the iteration was primarily devoted to the study of evaluation methods, which was crucial for the continuation of the efforts, as the stakeholders were not experienced in formal evaluation methods. The activities of the situational method were related to the structuring and evaluation WorkUnits. Situational factors of Business Units, Products, and Stakeholders were considered.

The *fourth iteration* was devoted to the pragmatic familiarization of the characteristics of the selected tools. The first objective was to learn the basic usage of the tools in order to conduct practical DSM processes in the divergent modeling environments. The second objective was to learn tool-specific approaches to model transformation. The selected DSM tools in question were MetaEdit+, Eclipse Modeling Tools, Borland Together, TOPCASED, and Rational Software Architect. The majority of the tools are based on the Eclipse platform, which made the familiarization quicker, as the basic functionalities were shared. In the feedback data it was denoted that the commercial tools provide better DSM process support, maturity, and usability as well as streamlined integration to their respective platforms, which are desirable characteristics in the production context, and should thus be emphasized in the evaluation. The method in action constituted primarily the evaluation WorkUnits. Situational factors of Business Units and Products were investigated.

During the *fifth and sixth iterations*, the actual evaluation of the DSM tools was conducted. Meta-Method had taken its basic form in terms of providing method support for the engineering of situational evaluation methods for DSM tools. The first subset of the DSM tools were evaluated according to the situational method. The benefits and risks related to the selection between open source and commercial DSM tools were further studied. The benefits of open source DSM tools were determined as better interoperability via open file formats and source code, rapid development of new technologies by the open source community, and proven industrial applications. The risks of open source tools were delineated to be unpredictability, availability of customer support, learning curve, and complexity of the DSM process. The commercial DSM tools primarily address the risks of the open source DSM tools, whereas they introduce the risks of vendor lock-in and limited customization as well as restricted interoperability. The demonstrative DSM solution was developed according to the specification, and the characteristics of the specification process were evaluated between the candidates. The tools introduce divergent metamodeling concepts according to the varying meta-metamodels, which were analyzed for the best fit for the company's approach. The feedback data indicated that although the open source DSM tools show a great promise, the risks may be too critical for the business. The method support was adequate, although fine adjustments were made to Meta-Method, rendering the evaluation WorkUnits generic in the sense that they are mainly dependent on the evaluation approach selected. Situational factors considered were Products, Business Units, and Stakeholders. As the main WorkProduct, an evaluation report was produced.

The selection activities were initiated in the *seventh iteration* by evaluation data analysis, which was conducted by both quantitative and qualitative measures. Weighted averaging was utilized to derive aggregated criteria values from the quantitative metrics. As all the criteria were not quantified, a qualitative analysis was also conducted to support the formation of the recommendation for the tool selection. The candidates were argued for and against from various perspectives in the selection report. Nevertheless, a tool recommendation was made as a pair of conditional suggestions. It was concluded that if the emphasis of the selection was on the overall maturity, DSM process support, user experience, and customer support, the recommendation would be a commercial DSM tool. Alternatively, if the emphasis was on interoperability, compliance to standards, and unlimited customizability, the choice would be an open source DSM tool. As a personal recommendation we suggested a commercial DSM tool, as it provides the most stable product and streamlined DSM process support as well as a lean learning curve for the stakeholders. At the end of the iteration, the initial version of the selection report was delivered to the client for a final decision. As the conclusion of the evaluation activities, the selection of the tool was validated in a workshop in 21st Jan 2011, in which the reasoning of the selection was discussed, practical demonstrations with the selected tool were conducted, and the overall experience was analyzed. Finally, the evaluation effort ended and the client proceeded with the implementation of the selected tool. The outcomes of the validation workshop were updated in the selection report. Meta-Method provided the required method elements for the selection of the DSM tool, albeit its terminology was further unified. Situational factors of Stakeholders, Products, and Business Units were considered.

6.3.3 Remarks

The actual evaluation data collected for the selection of DSM tools during the evaluation effort was produced under a non-disclosure agreement (NDA). Thus, it is not available for discussion in this context. However, the methodical artifacts designed in the previous chapters of the thesis are considered as empirically grounded in the case study. The utility of Meta-Method is considered adequate, as it includes all the method elements and guidelines that were required in this specific evaluation effort and it successfully produced a recommendation for a DSM tool selection, which was validated by the case company. The primary constructional guideline employed in the SME activities was assembly-based. Furthermore, the data collection was stopped at the point when client decided to implement the selected tool, as the evaluation effort was concluded. Thus, no further data is available. An inquiry of the success of the implementation and long-term experiences from the usage of the selected tool is part of the future research. In conclusion, Meta-Method was found useful in the case study and no further method elements were required. An evaluation of the utility of Meta-Method as progress against ISO 14102 is presented in the next section.

6.3.4 Utility

The primary evaluation criterion for DSR artifacts is utility, which represents the measure to which the artifact produces its desired effect, i.e. achieves its goal in the application context (Hevner et al., 2004; Venable et al., 2012). Aier and Fischer (2011) suggest that there are three types of utility: gross utility, net utility, and efficiency, which are defined in TABLE 21. *Gross utility* refers to the absolute utility of the design theory, i.e. the success of the application of the instantiation of the theory without the consideration of the costs of design and implementation. Gross utility is evaluated in this analysis, since the financial data required to evaluate the net utility and efficiency are not available to us.

Meta-Method was developed for a real-word business need to evaluate the state-of-the-art of DSM tools in a situational context and ultimately to select the optimal DSM tool for implementation. The design of the method was based on a case study, in which it was constructed, instantiated, tailored, enacted, and evaluated in an iterative design process, as presented in the previous sections. Meta-Method is a generalized baseline method for the engineering of situational evaluation methods for DSM tools. During the case study it was concluded that ISO 14102 is not sufficient for the evaluation of DSM tools in that specific situational context, which is primarily due to the broad scope and purpose of ISO 14102: to provide a generic method for the evaluation of all types of ISD tools. As the standard document (ISO, 2008) states, ISO 14102 is meant to be tailored according to organizational needs. Meta-Method can also be considered as a specific type of tailoring of ISO 14102, as it includes the majority of the method elements provided by ISO 14102. However, the method elements provided by Meta-Method are based on the synthesis of several previous evaluation methods, as presented in Chapter 5. In addition to the proposal of the method elements, SME is suggested as the approach for the construction and tailoring of evaluation methods for DSM tools, according to the situational factors and construction guidelines, whereas the tailoring of ISO 14102 lacks method support altogether. Furthermore, ISO 14102 provides a list of generic tool characteristics for the evaluation of ISD tools, which was found insufficient for the evaluation of DSM tools. Artifact I addresses this deficiency by providing a specialized checklist for the formulation of the evaluation criteria for DSM tools, along with examples, values, ranges and data types.

The utility of ISO 14102 has been demonstrated by Hilera and Martinez (1999) as well as Lundell and Lings (2002). The utility of Meta-Method is discussed in the previous section. As an established international standard, ISO 14102 is useful as a general guideline in the evaluation of ISD tools. Meta-Method is derived from a rigorous body of work, and its instantiations are found useful in the case study. Naturally, a single case study is not sufficient to demonstrate general utility for Meta-Method, which is why future research is required to validate Meta-Method and its usefulness in the engineering of situational evaluation methods for DSM tools in other contexts. Furthermore, a follow-up inquiry on the success of the implementation and the contingent longitudinal use of the selected tool in the case study context would provide more evidence on the gross utility, net utility, and efficiency of Meta-Method. For the future research, three testable propositions are delineated: (1) Meta-Method can successfully support the engineering of situational evaluation methods for DSM tools in various contexts in which DSM tools are evaluated, due to the use of SME principles. (2) The application of the evaluation criteria checklist can produce meaningful evaluation results that should be commensurate between different evaluation effort instances. (3) The application of the situational evaluation methods engineered with Meta-Method can produce successful DSM tool recommendations, at least when measured in gross utility.

6.4 Conceptual Evaluation

In this section the conceptual evaluation of Meta-Method is presented. The potential progress achieved in the design theory development of Meta-Method, in comparison to ISO 14102, is evaluated against the evaluation criteria of progress for IS design theories (Aier & Fischer, 2011). There is no methodical support available for the definition of the precise measures for each criterion. Hence, an example application of the criteria to Codd's model for relational databases as progress beyond pre-relational tree-structured files and network models of data (Aier & Fischer, 2011, p. 165) is utilized as a guideline for the evaluation. The subjective nature and the qualitative approach of the evaluation is in any case reflected in the precision of the measures. Therefore, they should be considered respectively.

6.4.1 Internal Consistency

Internal consistency refers to the principle that design theory elements should be based on a coherent system of constructs (Aier & Fischer, 2011). Internal consistency is also emphasized by Hevner et al. (2004) and Prat et al. (2014). The ISO 14102 document has been refined in numerous iterations of meetings, ballots, and other activities of standardization over the years. Thus, its internal consistency is up to par. There seems to be no notable inconsistencies in the terminology, figures, or overall presentation of the constructs. In comparison, Meta-Method is a novel design theory, designed by a single author, synthesized from various independent sources of respective sets of terminology, style, and representation. Thus, it most likely contains more inconsistencies than ISO 14102. Furthermore, ISO 14102 is designed for a broader purpose and scope than Meta-Method, which is reflected in the design theorization.

The internal consistency of Meta-Method can be evaluated from various aspects, such as the coherence and granularity of the method elements, and their consistency to the underpinning metamodels of origin. As Meta-Method is synthesized from various different sources, in which divergent or implicit metamodels are originally used, inconsistency issues may emerge from specific combinations during the instantiation of the constructs. The divergent metamodels present varying concepts at different levels of granularity, which has been a concern during the construction of Meta-Method. For example, method elements of varying granularity have been proposed in the literature, such as method parts, components, chunks, and fragments. Partly due to this problem, no specific metamodel is imposed on Meta-Method. Its generic nature allows the conceptual method elements to be instantiated in various types of SME processes and CAME tools. The minimum requirement is that the process engineering metamodel supports the basic method elements of WorkUnit, WorkProduct, and Producer.

Furthermore, the consistency of the terminology, figures, and overall presentation of Meta-Method constructs provided in this thesis can be evaluated. The unification of the terminology is a major issue in this domain, since the research in the many areas of interest, such as DSM and SME, is not very established. Similar problems are reported in evaluation practices (Arviansyah, 2013). The equivocality issues result in a plethora of different terms that refer to the equivalent or similar phenomena. The objective was to select the most utilized terms for Meta-Method. For the figures of the metamodels of the method elements, widely adopted UML diagrams were used. In overall, the presentation should conform to the required form and structure of the reporting standard in use, adapted from the conventions imposed by the Finnish language to those of American English. The presented constructs should also be coherent in relation to each other.

Similarly, the construction of Artifact I faced the problem of equivocality, which was addressed in the semantic unification of the criteria (see APPENDIX 2). Furthermore, one of the main goals of providing a unified checklist for the evaluation of DSM tools is the commensuration of evaluation results, which addresses the same type of problematic. Finally, the potential wider adoption of Meta-Method as a whole promotes a similar objective.

6.4.2 External Consistency

External consistency refers to the principle that a design theory should be based on justificatory knowledge that is consistent with the existing knowledge base (Aier & Fischer, 2011). The external consistency of ISO 14102 is not fully available for investigation, as the standard document does not provide references to the scientific corpus. The nature of the publication of the international standards is different from the conventions of scientific publication, as the efforts of individuals or organizations that contribute to the standardization process are not credited in the final documents. In this respect, including ISO 14102 as the other "design theory" in this comparison, can be considered as not optimal. In order to properly investigate the external consistency of a standard, one should analyze the archived documents of the standardization process, in which the intermediary stages of preparation are documented. This is out of the scope of this work. However, the previous standards on which the standard is built upon are documented. The earlier standardization work of IEEE 1209:1992 was continued in ISO 14102, which currently builds on various standards, such as ISO 9126, ISO 14598, and ISO 25051.

The method elements of Meta-Method are based on the synthesis of various previous evaluation methods (ISO, 2008; Kruchten, 2004; Lukman & Mernik, 2008; Wheeler, 2011; Morera, 2002; Lundell & Lings, 2003; Kitchenham, 1996). Proposed construction guidelines are based on essential SME literature (Ralyté & Rolland, 2001; Gupta & Prakash, 2001; Ralyte et al., 2003; Tolvanen, 1998; Graham et al., 1997; Seidita et al. 2007; Karlsson & Ågerfalk, 2009a; Deneckére, 2001; Ralyté et al., 2004). The situational factors are based on SME and evaluation literature (Bekkers et al., 2008; Song & Letch, 2012; Stockdale et al., 2008; Lundell & Lings, 2004b; Kitchenham, 1996). Artifact I is based on previous studies on the evaluation of DSM tools (Amyot et al., 2006; De Smedt, 2011; El Kouhen et al., 2012; Langlois et al., 2007; Pelechano et al., 2006; Saraiva & da Silva, 2008; Sivonen, 2008; Vasiljević et al., 2013). The structure and form of Artifact I should conform to the conventional format of technical checklists. The external consistency of Meta-Method was considered throughout its iterative design process as Meta-Method was entirely derived from the literature. The motivation for the selection of ISO 14102 as the initial baseline method was the availability of the standard and lack of more comprehensive methods. The tailoring of ISO 14102 was based on a practical business need, which was then reflected in the literature review that propagated the synthesis of the literature findings, eventually forming Meta-Method. In the pragmatic SME practices, the external consistency of situational methods is probably less emphasized, whereas the ad-hoc construction and tailoring of situational methods is more common for productivity reasons.

6.4.3 Broad Purpose and Scope

The *purpose and scope* of a design theory should be broad (Aier & Fischer, 2011). ISO 14102 has a very broad purpose and scope as it is intended to provide guidance for the evaluation of all types of ISD tools. This can be seen as an advantage or a deficiency. From the scientific point of view, generic artifacts are often considered "better", as for the practitioner, specialized artifacts can be more useful. ISO 14102 is intended to be tailored according to organizational needs, albeit it provides no method support for the tailoring. Meta-Method is intended for the engineering of situational evaluation methods for DSM tools in a context, proposing SME as the means of constructing and tailoring situational evaluation methods. Thereby, ISO 14102 has broader purpose and scope, whereas Meta-Method is a specialized method. As Meta-Method includes the main method elements of ISO 14102, effectively implementing the essential parts of ISO 14102, in addition to providing method support for the construction and tailoring of situational evaluation methods, it can be argued that Meta-Method would be useful also in the evaluations with broader purpose and scope, i.e. in the domain of ISD tools. To confirm this, further research is required.

6.4.4 Simplicity

Design theories should be *simple*, as simplicity promotes communicability, understandability and manageability as well as the cost effectiveness of their instantiations (Aier & Fischer, 2011). ISO 14102 and Meta-Method are both rather complex artifacts, as they provide means and measures for conducting an entire evaluation effort. The evaluation of their simplicity can be conducted in various ways, such as comparing the characteristics of their constructs or the simplicity of their instantiation, i.e. the characteristics of internal and external consistency as well as utility. The overall internal consistency of ISO 14102 is probably higher than of Meta-Method, as it is much more mature, which may be interpreted as higher simplicity. The number of constructs provided by ISO 14102 is lower than what is provided by Meta-Method, which may also be considered as a measure of simplicity. Due to the nature of the publication conventions of international standards, the overall external consistency of ISO 14102 in comparison to Meta-Method as well as its implications to simplicity, are challenging to evaluate. At least it can be stipulated that both provide some references to the source materials, thus enabling the retrieval of further guidance and original sources. As Meta-Method provides comprehensive referencing, it may be simpler to adapt, as the citations to further knowledge are available. The overall guidance provided by Meta-Method is more detailed than in ISO 14102, at least for the evaluation of DSM tools in a situational context. This could imply simpler instantiation, as the ad-hoc approach of ISO 14102 could create unnecessary complexity, at least with novice evaluators. On the other hand, the instantiation of Meta-Method requires skills and knowledge in areas such as SME and

DSM. In the beginning of the case study, the instantiation of ISO 14102 was perceived as challenging to a novice evaluator, as the guidance is very abstract in nature. This is naturally a subjective view. The study of the perceived simplicity of the instantiation of Meta-Method is part of the further work. In conclusion, the background knowledge of the evaluator is probably the most significant factor in the perceived simplicity of the instantiation of either of the methods.

6.4.5 Fruitfulness of New Research Findings

Fruitfulness of new research findings means that design theories should disclose new phenomena or previously undisclosed relationships between known phenomena as well as stimulate new research (Aier & Fischer, 2011). Meta-Method provides a novel method for the engineering of situational evaluation methods for DSM tools. This is achieved by denoting new relationships between previously known artifacts, motivated by a real-world business need. SME, conventionally utilized in the construction and tailoring of ISD methods, is in this work considered as the overarching mechanism for instantiating Meta-Method into situational evaluation methods by applying case-specific constructional guidelines and situational factors. ISO 14102 and several other previous evaluation methods are decomposed into conceptual method elements and synthesized into a unified method base of Meta-Method. Furthermore, a synthesis of several previous evaluation criteria for DSM tools are decomposed and synthesized as an evaluation criteria checklist for DSM tools. As a whole, Meta-Method introduces a novel synthesis of the known theories, and specializes in the engineering of situational evaluation methods for DSM tools. The novel intersections of the theories should be further studied and validated by conceptual research. Meta-Method and its instantiations should be further refined from the practical point of view as well as evaluated for utility in other situational contexts. Furthermore, the applicability of the method elements in different types of SME activities and CAME tools based on various divergent metamodels should be studied. Additionally, experiences from the application of the constructional guidelines and elicitation of relevant situational factors would be of interest in the further inquiries. Finally, the evaluation criteria checklist for DSM tools should be further refined and validated for utility.

6.5 Summary

This chapter presented the empirical and conceptual evaluation of the novel design theory, Meta-Method, derived from Artifacts I and II discussed in the previous sections. The main objective was to seek validation for the artifacts by adapting them into a single design theory called Meta-Method, and by comparing the outcome against the similarly derived design theory of ISO 14102, with the evaluation criteria of progress for IS design theories. First, the evaluation methodology was described. Second, the design theorization in which the ISO 14102 and Meta-Method were structured as design theory components, was presented. Third, the empirical and conceptual evaluations of Meta-Method were presented. The empirical evaluation was based on the analysis of the data collected during the case study in a company operating in the industry of professional mobile radio networks and devices. The conceptual evaluation was based on the comparative analysis of the presented design theories. The evaluations were provided on the basis of the six evaluation criteria of progress for IS design theories: utility, internal consistency, external consistency, broad purpose and scope, simplicity, and fruitfulness of new research. Utility was evaluated in the empirical evaluation, whereas the other criteria were addressed in the conceptual evaluation.

The *utility* was evaluated in terms of gross-utility, due to the unavailability of financial data. In comparison to ISO 14102, Meta-Method was found to be more useful for the evaluation of DSM tools in the case study context. The *internal consistency* of ISO 14102 was considered to be most likely "better" than Meta-Method's, due to the maturity of the international standard. The *external consistency* of ISO 14102 is challenging to evaluate, due to the publication policy of standard documents. It was concluded that while both design theories provide consistency with external knowledge, in the light of the data that is available to us, more evidence of Meta-Method's consistency with external knowledge is found. *Broad purpose and scope* is inherently "better" in ISO 14102, as it is a highly generalized method for the evaluation of ISD tools, whereas Meta-Method is a specialized method for the engineering of situational evaluation methods for DSM tools. It was argued that Meta-Method implements the main method elements of ISO 14102. Thus, it could probably be utilized with a broader scope as well, which is to be confirmed in future research. *Simplicity* was considered from several perspectives, establishing that both of the design theories are rather complex. ISO 14102 includes fewer constructs, whereas Meta-Method provides more detailed guidance, either of which could be a measure of simplicity. Meta-Method's *fruitfulness of new research* was analyzed in terms of identifying its potential areas that require or would benefit from future research, such as the conceptual investigation of the novel intersections of the integrated theories as well as the empirical studies that instantiate Meta-Method in novel situational contexts.

7 SUMMARY AND CONCLUSION

This chapter discusses the summary and conclusion of the study. First, the summary of the thesis is presented. Second, the conclusion of the research is discussed. Third, the limitations of the study are discussed. Finally, the directions for future research are outlined.

7.1 Summary

The evaluation of DSM tools in a situational context requires method support. Typically, DSM tools are evaluated in the industry for the purposes of investigating the opportunities to implement DSM in software production, or to justify the upgrade of the current DSM tools in use. DSM tools are also evaluated for research purposes. A method is intrinsic to any evaluation effort, dictating how the evaluation is conducted. There is no single evaluation method that is suitable for every usage situation. Thus, an evaluation method should be constructed/tailored according to the characteristics of the situational context. The current literature provides very limited support for the engineering of situational evaluation methods for DSM tools.

The research gap was identified in Chapter 1 and further stipulated in the form of a research problem: How to methodically support the engineering of situational evaluation methods for DSM tools? In the process of investigating the problem domain, Situational Method Engineering (SME) was identified as a useful approach for the engineering of situational ISD methods. In the literature, SME has also been considered for the engineering of situational methods for other organizational processes. This led us to our premise that SME would probably provide a useful foundation for the engineering of situational evaluation methods for DSM tools, too. In order to address the research problem, we have designed two artifacts, Artifact I and Artifact II. Artifact I is an evaluation criteria checklist for DSM tools, providing practical guidance for the formulation of evaluation criteria for DSM tools. Artifact II is a conceptual baseline method for the engineering of situational evaluation methods for DSM tools. Thus, Artifact I is utilized in the enactment of situational evaluation methods for DSM tools, which are engineered by instantiating Artifact II in a situational context. In the artifact design we have utilized the Design Science Research (DSR) research framework and its research process, which were discussed in Chapter 1.

The research problem was decomposed into five research questions: (1) What are Domain-Specific Modeling and DSM tools? (2) Which evaluation criteria are proposed in the literature for DSM tools and how to classify them? (3) What are the situational factors affecting the engineering and enactment of the evaluation methods for DSM tools? (4) Which method elements are proposed in the literature for the evaluation of ISD tools and how to classify them? (5) How to engineer situational evaluation methods for DSM tools? The design of the artifacts was achieved by addressing these research questions.

The *first research question* was discussed in Chapter 2, by providing an overview of the DSM approach and DSM tools, and by defining the basic DSM concepts utilized throughout the study. The *second research question* was discussed in Chapter 3, by providing the design of Artifact I, in which the evaluation criteria used in previous evaluations of DSM tools were classified, and adapted into a unified checklist of the evaluation criteria for DSM tools, with data types, ranges, and examples of the criteria values. The *third research question* was discussed in Sections 4.1, 5.3, and 5.5, by presenting the socio-technical dimensions of evaluation and by aligning them in respect to the situational factors in the context of SME. The *fourth research question* was first discussed in Sections 4.2 and 4.3, by structuring the activities of existing evaluation methods for ISD tools and by classifying them according to the phases of ISO 14102. Then, in Section 5.3 the method elements were extracted from the synthesis of the existing method elements and DSM literature, and presented as the method base of Artifact II. The *fifth research question* was addressed in Chapter 5, by providing the design of Artifact II, in which the basic components of SME are utilized: a method base, situational factors, and construction guidelines. Thus, the research questions three and four provide the method base as well as the situational factors of Artifact II, whereas the construction guidelines were presented on a general level. Finally, the artifact designs are evaluated in Chapter 6, by analyzing them in terms of empirical and conceptual criteria.

The evaluation of the artifacts was emphasized in this study. The business need of the case study company provided the required relevance for the study, whereas the theoretical grounding and evaluation of the artifacts provided the necessary rigor. In Chapter 6, the evaluation methodology, as well as the empirical and conceptual evaluation were presented. The artifacts were combined as a single design theory, Meta-Method, and evaluated against the similarly derived design theory of ISO 14102. In the empirical evaluation, case study was utilized. In the conceptual evaluation a theoretical analysis of the design theories were conducted. In the evaluation, the criteria of progress for IS design theories were utilized: utility, internal consistency, external consistency, broad purpose and scope, simplicity, and fruitfulness of new research.

The empirical evaluation discussed the case study, in which Artifacts I and II were iteratively designed and instantiated in the engineering of the situational evaluation methods for DSM tools, in addition to the enactment and evaluation of the instantiations. The case study employed an incremental artifact design method that was organized in seven iterations. Furthermore, a kick-off meeting and a final workshop were arranged. The evaluation data was collected in feedback meetings that were organized at the end of the iterations. The collected data was stored as shared meeting minutes. As the initial baseline method for the case study effort, ISO 14102 was selected. However, ISO 14102 was deemed to be too generic for the tasks at hand, providing very limited support for the evaluation of DSM tools on a practical level. This is naturally a subjective view. This however was the initial motivation for this study. Artifacts I and II were then designed, applied, evaluated, and found useful in the case study. The versions of the artifacts created in the case study were very similar to the artifacts presented in this thesis. The artifacts were added additional rigor after the case study by further grounding them externally, without changing the essential characteristics of the artifacts. Due to this initial setting, in the evaluation, the artifacts were compared against ISO 14102 for progress as a combined design theory Meta-Method. The empirical evaluation provided evidence for the utility criterion of Meta-Method.

In the conceptual evaluation, the other criteria of progress for IS design theories were utilized in the theoretical analysis of the design theories. There is no method support for the evaluation of the criteria, which exposes the analysis to subjective bias. Furthermore, the criteria are qualitative and general in nature, due to which the evaluated aspects of the criteria, although remaining within the boundaries of the broad spectrum of the semantics of the criteria, were selected subjectively. In future research, it would be beneficial to design subcriteria, preferably quantitative in nature, to support the arguments made in the evaluation of the criteria of progress for IS design theories. This would also promote the adoption and commensuration of the criteria.

The *utility* criterion was evaluated in terms of gross-utility. In comparison to ISO 14102, Meta-Method was found to be more useful for the evaluation of DSM tools in the case study context. The *internal consistency* of ISO 14102 was evaluated to be most likely "better" than that of Meta-Method, due to the maturity of the international standard. The *external consistency* of ISO 14102 is challenging to evaluate, due to the publication policy of the standard documents. It was concluded that while both design theories provide consistency with external knowledge, in the light of the data available to us, more evidence on the consistency of Meta-Method with external knowledge is found. *Broad purpose and scope* is inherently "better" in ISO 14102, as it is a highly generalized method for the evaluation of ISD tools, whereas Meta-Method is a specialized method for the engineering of situational evaluation methods for DSM tools. However, Meta-Method should be useful in other instances of evaluations of DSM tools, beyond the context of the presented case study. Moreover, it was argued that Meta-Method implements the main method elements of ISO 14102. Thus, it could probably be utilized within a broader scope as well, which should be confirmed in future research. *Simplicity* was considered from several perspectives, establishing that both of the design theories are rather complex. ISO 14102 includes fewer constructs, whereas Meta-Method provides more detailed guidance, either of which could be a measure of simplicity. Meta-Method's *fruitfulness of new research* was analyzed in terms of identifying its potential areas that require or would benefit from future research, such as the conceptual investigation of the novel intersections of the integrated theories as well as the empirical studies that instantiate Meta-Method in novel situational contexts.

7.2 Conclusion

In conclusion, based on the application of the "ceteris paribus clause" on the evaluated criteria, no explicit statement can be made about which of the two derived design theories is "better". The scope and spectrum of the design theories are inherently different, which makes the comparison challenging. Nevertheless, we argue that Meta-Method would probably be more useful in the evaluation of DSM tools in a situational context, at least when conducted by an informed evaluator with SME skills. The main contribution of this study is Meta-Method, which is of interest both for researchers and practitioners. On the theoretical level, Meta-Method outlines the application of the SME principles in the evaluation of DSM tools and presents a conceptual evaluation of the structured design theory. On the practical level, Meta-Method provides method support for the engineering of situational evaluation methods for DSM tools as well as promotes the commensuration of the evaluation results. An empirical evaluation was conducted to provide evidence for the utility of Meta-Method. Further validation and verification is required in the future research in order to derive more generalized assumptions of Meta-Method.

7.3 Limitations

The main limitation of the research was the scarcity of time, due to which the accomplishment of the study was dispersed across multiple periods of "micro studies", varying in length, location, and resources available. This may implicate incoherence in the various dimensions of the study. It is also evident that an exhaustive exploration of the demarcated subject area requires a number of future studies. Furthermore, the rather unestablished nature of the research in the key areas of interest, such as DSM and SME, could compromise the integrity of the theoretical grounding of the study. In addition, the design of the artifacts combines theories from multiple self-contained areas, which may implicate conceptual and empirical incompatibility issues that remain unexplored within the boundaries of this study. As for the empirical evaluation, the main limitation is the lack of triangulation of data sources, which is recommended by the established practices of case study. Furthermore, the validity and reliability of the collected data in the case study can be compromised, as the empirical endeavor taken was not originally designed as a formal case study. Additionally, the lack of method support and the qualitative nature of the conceptual evaluation exposes the evaluation to subjective bias. Furthermore, the derived design theory will need further validation and verification in the conceptual as well as empirical dimensions. Finally, as a technical limitation, the length of the thesis is a factor that delimits the level of detail in which the subject area can be discussed.

7.4 Future Research

This thesis introduced numerous novel intersections of self-contained theories that should be further studied and validated by conceptual research. The testable propositions outlined for Meta-Method should be tested in the future research. Meta-Method and its instantiations should be further refined from the practical point of view as well as evaluated for utility in other situational contexts. The future case studies should also address the triangulation of data sources by establishing rigorous research methods in the collection of diverse qualitative and quantitative data. A controlled experiment in a lab setting could also provide valuable insight into the various dimensions of Meta-Method, such as usability. Furthermore, the applicability of the elicited method elements in different types of SME activities and CAME tools based on various divergent metamodels should be studied. The investigations of the applicability of SME beyond the domain of ISD should also be continued and expanded. Moreover, as one of the key barriers of SME adoption seems to be the scarcity of production-ready tool support, a practical ISD tool for process engineering with Meta-Method should be developed. This could be accomplished by utilizing a CAME tool or building a standalone tool. Additionally, experiences from the application of the constructional guidelines and elicitation of relevant situational factors would be of interest in the further inquiries. Multi-disciplinary research could benefit the elicitation and validation of the socio-technical factors affecting the SME processes. Moreover, the evaluation criteria checklist for DSM tools should be further refined and validated for utility. In the long run, the potential effect of the adoption of the checklist on the commensuration of the evaluations of DSM tools should be investigated. Finally, proper method support for the construction and evaluation of IS design theories should be constructed to promote the DSR research in general as well as the validation and verification of IS design theories.

BIBLIOGRAPHY

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods - Review and analysis* (VTT Publications 478). Espoo, Finland: VTT.

Achilleos, A., Georgalas, N. & Yang, K. (2007). An Open Source Domain-Specific Tools Framework to Support Model Driven Development of OSS. In D. Akehurst, R. Vogel & R. Paige (Eds.), *Model Driven Architecture - Foundations and Applications, LNCS 4530* (pp. 1-16). Berlin, Germany: Springer-Verlag.

Ågerfalk, P. J. & Fitzgerald, B. (2005). Methods as Action Knowledge: Exploring the Concept of Method Rationale in Method Construction, Tailoring and Use. In T. Halpin, K. Siau & J. Krogstie (Eds.), *Proceedings of the 10th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'05) held in conjunction with the 17th Conference on Advanced Information Systems (CAiSE'05)* (pp. 27-40). Porto, Portugal: 13-14 June, 2005.

Aier, S. & Fischer, C. (2011). Criteria of progress for information systems design theories. *Information Systems and E-Business Management, 9(1),* 133-172.

Amyot, D., Farah, H. & Roy, J. (2006). Evaluation of Development Tools for Domain-Specific Modeling Languages. In R. Gotzhein & R. Reed (Eds.), *System Analysis and Modeling: Language Profiles, Germany, LNCS 4320* (pp. 183-197). Berlin, Germany: Springer-Verlag.

Andersson, C. & Runeson, P. (2007). A spiral process model for case studies on software quality monitoring - method and metrics. *Software Process: Improvement and Practice, 12(2),* 125-140.

Antkiewicz, M. (2006). Round-Trip Engineering of Framework-Based Software using Framework-Specific Modeling Languages. In O. Nierstrasz, J. Whittle, D. Harel & G. Reggio (Eds.), *Model Driven Engineering Languages and Systems, LNCS 4199* (pp. 692-706). Berlin, Germany: Springer-Verlag.

Arviansyah, A., Spil, T. A. M. & Hillegersberg, J. v. (2013). Evaluating IS/IT Projects: Revealing the Causes of Equivocality. In *Proceedings of Pacific Asia Conference on Information Systems (PACIS) 2013, [CDROM],* Jeju Island, South Korea, June 18-22, 2013.

Atkinson, C. & Kühne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *IEEE Software, 20(5),* 36-41.

Atkinson, C. & Kühne, T. (2005). Concepts for Comparing Modeling Tool Architectures. In L. C. Briand & C. Williams (Eds.), *Model Driven Engineering Languages and Systems, LNCS 3713* (pp. 398-413). Berlin, Germany: Springer-Verlag.

Bekkers, W., van de Weerd, I., Brinkkemper, S. & Mahieu, A. (2008). The Influence of Situational Factors in Software Product Management: An Empirical Study. In C. Ebert, S. Brinkkemper, S. Jansen & G. Heller (Eds.), In *Proceedings of the Second International Workshop on Software Product Management (IWSPM'08)* (pp. 41-48). Los Alamitos, CA: IEEE Computer Society.

Benbasat, I., Goldstein, D. K. & Mead, M. (1987). The case research strategy in studies of information systems. *MIS Quarterly, 11(3),* 369-386.

Booch, G., Rumbaugh, J. & Jacobson, I. (1998). *The Unified Modeling Language User Guide.* Reading, MA: Addison-Wesley Professional.

Bucher, T., Bajec, M., Furlan, Š, Kornyshova, E., Saidani, O., Vavpoti, D. & Žvanut, B. (2008). On the Application of the ISD Method Engineering Approach in Non-ISD Domains. St. Gallen: Working Paper, Institute of Information Management, University of St. Gallen, Switzerland.

Cervera, M. & Manoli, A., Torres, V. & Pelechano, V. (2012). The MOSKitt4ME Approach: Providing Process Support in a Method Engineering Context. In P. Atzeni, D. Cheung & S. Ram (Eds.), *Conceptual Modeling, LNCS 7532* (pp. 228-241). Berlin, Germany: Springer Berlin Heidelber.

Chau, T., Maurer, F. & Melnik, G. (2003). Knowledge Sharing: Agile Methods vs. Tayloristic Methods. In *Proceedings of the 12 IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2003)* (pp. 302-307). Piscataway, NJ: IEEE.

Cho, H. (2013). *A demonstration-based approach for domain-specific modeling language creation.* Doctoral Dissertation. University of Alabama.

Cho, H., Gray, J. & Syriani, E. (2012). Creating visual Domain-Specific Modeling Languages from end-user demonstration. In *Proceedings of the 4th International Workshop on Modeling in Software Engineering (MISE)* (pp. 22-28). Piscataway, NJ: IEEE.

Cho, H. & Gray, J. (2011). Design Patterns for Metamodels. In *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11* (pp. 25-32). New York, NY: ACM.

Czarnecki, K. & Helsen, S. (2006). Feature-based Survey of Model Transformation Approaches. *IBM Systems Journal, 45(3),* 621-645.

Czarnecki, K. & Helsen, S. (2003). Classification of Model Transformation Approaches. In *Proceedings of the OOPSLA'03 Workshop on Generative Techniques in the Context of MDA, [CDROM],* Anaheim, CA, October 27, 2003.

De Smedt, P. (2011). Comparing three graphical DSL editors: AToM3, MetaEdit+ and Poseidon for DSLs. Preprint, Submitted to Elsevier, University of Antwerp.

Deneckère, R. (2001). *Approche d'extension de méthodes fondée sur l'utilisation de composants génériques.* Doctoral Dissertation. University of Paris I Panthéon-Sorbonne.

El Kouhen, A., Dumoulin, C., Gerard, S. & Boulet, P. (2012). *Evaluation of Modeling Tools Adaptation* (00706701v2). Lyon, France: HAL.

Falkenberg, E. D., Hesse, W., Lindgreen, P., Nilsson, B. E., Oei, J. L. H., Rolland, C., Stamper, R. K., Assche, F. J. M. V., Verrijn-Stuart, A. A. & Voss, K. (1996). *FRISCO: A Framework of Information System Concepts,* The FRISCO Report (Web Edition), IFIP.

Elvesæter, B., Benguria, G. & Ilieva, S. (2013). A Comparison of the Essence 1.0 and SPEM 2.0 Specifications for Software Engineering Methods. In *Proceedings of the 3rd Workshop on Process-Based Approaches for Model-Driven Engineering (PMDE '13), [CDROM]*, Montpellier, France, July 2, 2013.

Favre, J. M. (2004). Towards a Basic Theory to Model Model Driven Engineering. In *Proceedings of the 3rd Workshop in Software Model Engineering (WiSME'04), [CDROM]*, Portugal, Lisbon, October 10-15, 2004.

Favre, J. M. (2005). Foundations of Meta-Pyramids: Languages vs. Metamodels -- Episode II: Story of Thotus the Baboon. In J. Bézivin & R. Heckel (Eds.), *Proceedings of the Dagstuhl Seminar on Language Engineering for Model-Driven Software Development*. Wadern: Schloss Dagstuhl - Leibniz Center for Informatics.

Ferreira, C. & Cohen, J. (2008). Agile Systems Development and Stakeholder Satisfaction: A South African Empirical Study. In R. Botha & C. Cilliers (Eds.), *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology (SAICSIT '08)*. New York, NY: ACM.

Firesmith, D. & Henderson-Sellers, B. (2002). *The OPEN Process Framework - An Introduction.* Harlow: Addison-Wesley.

Goldkuhl, G. (1999). *The grounding of usable knowledge: An inquiry in the epistemology of action knowledge* (CMTO Research Papers 1999:03). Linköping, Sweden: Linköping University.

Graham, I., Henderson-Sellers, B. & Younessi, H. (1997). *The OPEN process specification.* London: Addison-Wesley.

Gregor, S. & Jones, D. (2007). The anatomy of a design theory. *Journal of the Association for Information Systems, 8(5),* 312-335.

Gronback, R. C. (2009). *Eclipse modeling project: a domain-specific language toolkit.* Boston, MA: Addison-Wesley.

Gupta, D. & Prakash, N. (2001). Engineering Methods from Method Requirements Specifications. *Requirements Engineering Journal, 6(3),* 135-160.

Haumer, P. (2005). IBM Rational Method Composer: Part 1: Key concepts. *Rational Edge,* December 2005.

Haumer, P. (2006). Increasing Development Knowledge with Eclipse Process Framework Composer. *Eclipse Review*, BZ Media, Spring Issue, June 2006. 26-33.

Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. J. & Rossi, M. (2014). *Situational Method Engineering.* Berlin Heidelberg: Springer.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems, 19(2),* 87-92.

Hilera, J. R. & Martínez, J. J. (1999). Evaluation and selection of CASE tools: A real case. In *Proceedings of the 7th International Symposium on the Foundations of Software Engineering, [CDROM],* Toulouse, France, September 6-10, 1999.

Hitt, L. M., Wu, D. & Zhou, X. (2002). Investment in enterprise resource planning: Business impact and productivity measures. *Journal of Management Information Systems, 19(1),* 71-98.

Hoisl, B., Sobernig, S. & Strembeck, M. (2013). Higher-order Rewriting of Model-to-Text Templates for Integrating Domain-specific Modeling Languages. In S. Hammoudi, L. F. Pires, J. Filipe & R. C. D. Neves (Eds.), *Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development (MODELSWARD'13)* (pp. 49-61). Lisbon, Portugal: SCITEPRESS.

Hoppenbrouwers, S., Zoet, M., Versendaal, J. & van de Weerd, I. (2011). Agile Service Development: A Rule-Based Method Engineering Approach. In J. Ralyté, I. Mirbel & R. Deneckère (Eds.), *Engineering Methods in the Service-Oriented Context, IFIP Advances in Information and Communication Technology* (pp. 184-189). Berlin, Germany: Springer Berlin Heidelberg.

Hug, C., Front, A., Rieu, D. & Henderson-Sellers, B. (2009). A method to build information systems engineering process metamodels. *Journal of Systems and Software, 82(10),* 1730-1742.

IEEE (1995). *IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools* (IEEE Std 1348-1995). Los Alamitos, CA: IEEE Computer Society.

Iivari, J. (2007). A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems, 19(2),* 39-64.

Iivari, J. (2014). Distinguishing and contrasting two strategies for design science research. *European Journal of Information Systems, 24(1),* 107-115.

Iivari, J. & Venable, J. R. (2009). Action research and design science research - Seemingly similar but decisively dissimilar. In S. Newell, E. A. Whitley, N. Pouloudi, J. Wareham & L. Mathiassen (Eds.), *Proceedings of the 17th European Conference on Information Systems (ECIS'09)* (pp. 1642-1653), Verona, Italy: University of Verona.

Irani, Z. & Love, P. E. D. (2002). Developing a frame of reference for ex-ante IT/IS investment evaluation. *European Journal of Information Systems, 11(1),* 74-82.

Irani, Z. & Love, P. E. D. (2008). *Evaluating Information Systems: Public and Private Sector.* Hungary: Butterworth-Heinemann.

ISO (1998). *Information Technology – Software Product Evaluation - Part 5: Process for Evaluators* (ISO/IEC 14598-5:1998). Geneva, Switzerland: International Organization for Standardization.

ISO (2001). *Software Engineering - Product Quality - Part 1: Quality Model* (ISO/IEC 9126-1:2001). Geneva, Switzerland: International Organization for Standardization.

ISO (2007). *Software Engineering - Metamodel for Development Methodologies* (ISO/IEC 24744:2007). Geneva, Switzerland: International Organization for Standardization.

ISO (2008). *Information Technology - Guideline for the evaluation and selection of CASE tools* (ISO/IEC 14102:2008). Geneva, Switzerland: International Organization for Standardization.

Jacobson, I., Ng, P. W. & Spence, I. (2007). Enough of Processes - Lets do Practices. *Journal of Object Technology, 6(6),* 41-66.

Jones, C. (2009, 17th March). Programming Languages Table [PLT2006b]. Retrieved 2013-10-06 from http://www.spr.com

Jones, S. (2008). Social dimension of IT/IS evaluation: Views from the public sector. In Z. Irani & P. E. D. Love (Eds.), *Evaluating Information Systems: Public and Private Sector* (pp. 236-256). Hungary: Butterworth-Heinemann.

Karagiannis, D. & Kühn, H. (2002). Metamodelling platforms. In K. Bauknecht, A. Tjoa Min & G. Quirchmayr (Eds.), *E-Commerce and Web Technologies, LNCS 2455* (pp. 182). Berlin, Germany: Springer-Verlag.

Karlsson, F. (2013). Longitudinal use of method rationale in method configuration. *European Journal of Information Systems, 22(6),* 690-710.

Karlsson, F. & Ågerfalk, P. J. (2009a). Towards structured flexibility in information systems development: devising a method for method configuration. *Journal of Database Management, 20(3),* 51-75.

Karlsson, F. & Ågerfalk, P. J. (2009b). Exploring agile values in method configuration. *European Journal of Information Systems, 18(4),* 300-316.

Kärnä, J., Kelly, S. & Tolvanen, J.-P. (2009). Evaluating the use of domain-specific modeling in practice. In M. Rossi, J. Sprinkle, J. Gray & J.-P. Tolvanen (Eds.), *Proceedings of the The 9th OOPSLA Workshop on Domain-Specific Modeling* (pp. 14-20). Helsinki: HSE Print.

Kelly, S. (2007, 1st December). Domain-Specific Modeling Languages: Moving from Writing Code to Generating It. Retrieved 2014-06-08 from http://msdn.microsoft.com/en-us/library/cc168592.aspx

Kelly, S. (2013). Empirical Comparison of Language Workbenches. In J. Gray, S. Kelly & J. Sprinkle (Eds.), *Proceedings of the 2013 ACM Workshop on Domain-specific Modeling* (pp. 33-38). New York, NY: ACM.

Kelly, S., Lyytinen, K. & Rossi, M. (2013). MetaEdit+ A fully configurable multi-user and multi-tool CASE and CAME environment. In J. Bubenko, J. Krogstie, O. Pastor, B. Pernici, C. Rolland & A. Sølvberg (Eds.), *Seminal Contributions to Information Systems Engineering* (pp. 109-129). Berlin, Germany: Springer-Verlag.

Kelly, S. & Tolvanen, J.-P. (2008). *Domain-Specific Modeling: Enabling Full Code Generation.* Hoboken, NJ: John Wiley & Sons.

Kent, S. (2002). Model Driven Engineering. In M. Butler, L. Petre & K. Sere (Eds.), *Integrated Formal Methods, LNCS 2335* (pp. 286-298). London, UK: Springer-Verlag.

Kern, H., Hummel, A. & Kühne, S. (2011). Towards a Comparative Analysis of Meta-Metamodels. In *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11* (pp. 7-12). New York, NY: ACM.

Kirchner, L. & Jung, J. (2007). A Framework for the Evaluation of Meta-Modelling Tools. *Electronic Journal of Information Systems Evaluation, 10(1),* 65-72.

Kitchenham, B. (1996). *DESMET: A Method for Evaluating Software Engineering Methods and Tools* (Technical Report TR96-09). Keele, UK: University of Keele.

Kitchenham, B. A. & Pfleeger, S. L. (2002a). Principles of Survey Research Part 2: Designing a Survey. *SIGSOFT Software Engineering Notes, 27(1),* 18-20.

Kitchenham, B. & Pfleeger, S. L. (2002b). Principles of Survey Research Part 4: Questionnaire Evaluation. *SIGSOFT Software Engineering Notes, 27(3),* 20-23.

Kitchenham, B. & Pfleeger, S. L. (2003). Principles of Survey Research Part 6: Data Analysis. *SIGSOFT Software Engineering Notes, 28(2),* 24-27.

Kornyshova, E. (2011). *MADISE: Method Engineering-based Approach for Enhancing Decision-Making in Information Systems Engineering.* Doctoral Dissertation. University of Paris I Panthéon-Sorbonne.

Kornyshova, E., Deneckére, R. & Rolland, C. (2011). Method families concept: application to decision-making methods. In T. Halpin, S. Nurcan, J. Krogstie, P. Soffer, E. Proper, R. Schmidt & I. Bider (Eds.), *Enterprise, Business-Process and Information Systems Modeling*, *Lecture Notes in Business Information Processing* (pp. 413-427). Berlin, Germany: Springer-Verlag Heidelberg.

Kruchten, P. (2004). *The rational unified process: an introduction.* Boston, MA: Addison-Wesley Professional.

Kuechler, B. & Vaishnavi, V. (2008). On theory development in design science research: anatomy of a research project. *European Journal of Information Systems, 17(5),* 489-504.

Kuhn, T. S. (1977). *The essential tension: Selected studies in scientific tradition and change.* Chicago, IL: University of Chicago Press.

Land, F. (2001). IS Evaluation: Recent Trends, Keynote Speech. In *Proceedings of NUKAIS Information Systems Evaluation Seminar, [CDROM],* Priestley Hall, Leeds Metropolitan University, UK, February 27, 2001.

Langlois, B., Jitia, C. & Jouenne, E. (2007). DSL Classification. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07), [CDROM],* Montreal, Canada, October 21-22, 2007.

Leppänen, M. (2005). *An ontological framework and a methodical skeleton for method engineering : a contextual approach.* Doctoral Dissertation. University of Jyväskylä.

Lings, B. & Lundell, B. (2005). On the adaptation of Grounded Theory procedures: insights from the evolution of the 2G. *IT & People, 18(3),* 196-211.

Ludewig, J. (2003). Models in software engineering - an introduction. *Software and Systems Modeling, 2(1),* 5-14.

Lukman, T. & Mernik, M. (2008). Model-Driven Engineering and its introduction with metamodeling tools. In *9th International PhD Workshop on Systems and Control: Young Generation Viewpoint, [CDROM],* Izola, Slovenia, October 1-3, 2008.

Lundell, B. & Lings, B. (2002). Comments on ISO 14102: the standard for CASE-tool evaluation. *Computer Standards & Interfaces, 24(5),* 381-388.

Lundell, B. & Lings, B. (2003). The 2G method for doubly grounding evaluation frameworks. *Information Systems Journal, 13(4),* 375-398.

Lundell, B. & Lings, B. (2004a). Method in action and method in tool: a stakeholder perspective. *Journal of Information Technology, 19(3),* 215-223.

Lundell, B. & Lings, B. (2004b). On Understanding Evaluation of Tool Support for IS Development. *Australasian Journal of Information Systems, 12(1),* 39-53.

Maes, K., Van Grembergen, W. & De Haes, S. (2014). Identifying Multiple Dimensions of a Business Case: A Systematic Literature Review. *Electronic Journal of Information Systems Evaluation, 17(1),* 47-59.

Mallouli, S. D., & Assar, S. (2013). Enacting a Requirement Engineering Process with Meta-Tools: an Exploratory Project. In *Proceedings of the Eighth International Multi-Conference on Computing in the Global Information Technology (ICCGI '13), [CDROM]*, Nice, France, July 21-26, 2013.

Miah, S. J., Gammack, J. G. & Kerr, D. V. (2012). A Socio-technical Approach to Designing and Evaluating Industry Oriented Applications. *Electronic Journal of Information Systems Evaluation, 15(2),* 163-175.

Mohagheghi, P. & Haugen, Ø (2010). Evaluating Domain-Specific Modelling Solutions. In J. Trujillo, G. Dobbie, H. Kangassalo, S. Hartmann, M. Kirchberg, M. Rossi, I. Reinhartz-Berger, E. Zimányi & F. Frasincar (Eds.), *Advances in Conceptual Modeling – Applications and Challenges, LNCS 6413* (pp. 212-221). Berlin, Germany: Springer Berlin Heidelberg.

Morera, D. (2002). COTS Evaluation Using Desmet Methodology & Analytic Hierarchy Process (AHP). In M. Oivo & S. Komi-Sirviö (Eds.), *Product Focused Software Process Improvement, LNCS 2559* (pp. 485-493). Berlin, Germany: Springer Berlin Heidelberg.

Niknafs, A. & Ramsin, R. (2008). Computer-Aided Method Engineering: An Analysis of Existing Environments. In Z. Bellahsène & M. Léonard (Eds.), *Advanced Information Systems Engineering, LNCS 5074* (pp. 525-540). Berlin, Germany: Springer Berlin Heidelberg.

Nytun, J., Prinz, A. & Tveit, M. (2006). Automatic Generation of Modelling Tools. In A. Rensink & J. Warmer (Eds.), *Model Driven Architecture – Foundations and Applications, LNCS 4066* (pp. 268-283). Berlin, Germany: Springer Berlin Heidelberg.

Obeo (2014, 13th April). Obeo Designer: Domain-Specific Modeling for Software Architects. Retrieved 2014-05-10 from http://www.obeodesigner.com/

OMG (2006). *Meta Object Facility Core Specification* (MOF 2.0). Needham, MA: Object Management Group.

OMG (2008a). *MOF Model To Text Transformation Language (*MOFM2T 1.0). Needham, MA: Object Management Group.

OMG (2008b). *Software & Systems Process Engineering MetaModel Specification* (SPEM 2.0). Needham, MA: Object Management Group.

OMG (2014). *Essence - Kernel and Language for Software Engineering Methods* (Essence 1.0 - Beta 2). Needham, MA: Object Management Group.

Paul, R. J. (2007). Challenges to information systems: time to change. *European Journal of Information Systems, 16(3),* 193-195.

Peffers, K., Tuunanen, T., Rothenberger, M. & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems, 24(3),* 45-77.

Pelechano, V., Albert, M., Muñoz, J. & Cetina, C. (2006). Building Tools for Model Driven Development. Comparing Microsoft DSL Tools and Eclipse Modeling Plug-ins. In *Proceedings of 11th Conference on Software Engineering and Databases (JISBD'06), [CDROM]*, Barcelona, Spain, October 3-6, 2006.

Pfleeger, S. L. & Kitchenham, B. A. (2001). Principles of Survey Research: Part 1: Turning Lemons into Lemonade. *SIGSOFT Software Engineering Notes, 26(6),* 16-18.

Prat, N., Comyn-Wattiau, I. & Akoka, J. (2014). Artifact Evaluation in Information Systems Design-Science Research - A Holistic View. In *Proceedings of Pacific Asia Conference on Information Systems (PACIS) 2014, [CDROM],* Chengdu, China, June 24-28, 2014.

Ralyté, J., Deneckére, R. & Rolland, C. (2003). Towards a Generic Model for Situational Method Engineering. In J. Eder & M. Missikoff (Eds.), *Advanced Information Systems Engineering, LNCS 2681* (pp. 95-110). Berlin, Germany: Springer Berlin Heidelberg.

Ralyté, J. & Rolland, C. (2001). An Approach for Method Reengineering. In H. S.Kunii, S. Jajodia & A. Sølvberg (Eds.), *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001), LNCS 2224* (pp. 471-484). Yokohama, Japan: Springer Berlin Heidelberg.

Ralyté, J., Rolland, C. & Deneckère, R. (2004). Towards a Meta-tool for Change-Centric Method Engineering: A Typology of Generic Operators. In A. Persson & J. Stirna (Eds.), *Advanced Information Systems Engineering, LNCS 3084* (pp. 202-218). Berlin, Germany: Springer Berlin Heidelberg.

Rivas, L., Perez, M., Mendoza, L. E. & Griman, A. (2010). Tools Selection Criteria in Software-Developing Small and Medium Sized Companies. *Journal of Computer Science & Technology, 10(1),* 24-30.

Robson, C. (2002). *Real world research, 2nd edition.* Oxford, United Kingdom: Blackwell Publishing.

Rugaber, S. & Stirewalt, K. (2004). Model-driven reverse engineering. *Software, IEEE, 21(4),* 45-53.

Runeson, P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14(2),* 131-164.

Runeson, P., Höst, M., Rainer, A. & Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples* (1st Edition). Hoboken, New Jersey: John Wiley & Sons.

Sánchez-Ruíz, A. J., Saeki, M., Langlois, B. & Paiano, R. (2006). Domain-Specific Software Development Terminology: Do We All Speak the Same Language? In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07), [CDROM],* Montreal, Canada, October 21-22, 2007.

Saraiva, J. d. S. & da Silva, A. R. (2008). Evaluation of MDE Tools from a Metamodeling Perspective. *Journal of Database Management, 19(4),* 21-46.

Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering. *Computer, 39(2),* 25-31.

Schwaber K. & Sutherland K. (2014, 25th November). *The Scrum Guide. Retrieved* 2015-02-20 *from http://www.scrumguides.org/scrum-guide.html.*

Scriven, M. (2003). Evaluation Theory and Metatheory. In T. Kellaghan & D. L. Stufflebeam (Eds.), *International Handbook of Educational Evaluation, Kluwer International Handbooks of Education 9* (pp. 15-30). Houten, Netherlands: Springer Netherlands.

Scriven, M. (2001). An overview of evaluation theories. *Evaluation Journal of Australasia, 1(2),* 27-29.

Seidewitz, E. (2003). What models mean. *Software, IEEE, 20(5),* 26-32.

Seidita, V., Ralyté, J., Henderson-Sellers, B., Cossentino, M. & Arni-Bloch, N. (2007). A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods. In J. Eder, S. L. Tomassen, A. Opdahl & G. Sindre (Eds.), *Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering* (pp. 85-88). Trondheim, Norway: Sun SITE Central Europe, RWTH Aachen University.

Sein, M., Henfridsson, O., Purao, S., Rossi, M. & Lindgren, R. (2011). Action Design Research. *MIS Quarterly, 35(1),* 37-56.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software, 20(5),* 19-25.

Sivonen, S. (2008). *Domain-Specific Modelling Language and Code Generator for Developing Repository-Based Eclipse Plug-ins.* (VTT Publications 680). Espoo, Finland: VTT.

Song, X. & Letch, N. (2012). Research on IT/IS Evaluation: A 25 Year Review. *The Electronic Journal Information Systems Evaluation (EJISE), 15(3),* 276-287.

Stahl, T. & Völter, M. (2006). *Model-Driven Software Development: Technology, Engineering, Management.* Chichester, UK: John Wiley & Sons Ltd.

Steinberg, D., Budinsky, F., Paternostro, M. & Merks, E. (2009). *EMF: Eclipse Modeling Framework* (2nd Edition). Amsterdam, Netherlands: Addison-Wesley Longman, Amsterdam.

Stockdale, R., Standing, C., Love, P. E. D. & Irani, Z. (2008). Revisiting the content, context and process of IS evaluation. In Z. Irani & P. E. D. Love (Eds.), *Evaluating Information Systems: Public and Private Sector* (pp. 35-48). Hungary: Butterworth-Heinemann.

Tolvanen, J.-P. (1998). *Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence.* Doctoral Dissertation. University of Jyväskylä.

Tolvanen, J.-P., Pohjonen, R. & Kelly, S. (2007). Advanced Tooling for Domain-Specific Modeling: MetaEdit+. In *Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07), [CDROM],* Montreal, Canada, October 21-22, 2007.

Urquhart, C., Lehmann, H. & Myers, M. D. (2010). Putting the 'theory' back into grounded theory: guidelines for grounded theory studies in information systems. *Information Systems Journal, 20(4),* 357-381.

van Deursen, A., Klint, P. & Visser, J. (2000). Domain-specific languages: an annotated bibliography. *ACM SIGPLAN Notices, 35(6),* 26-36.

Vasiljević, I., Milosavljević, G., Dejanović, I. & Filipović, M. (2013). Comparison of Graphical DSL Editors. In *Proceedings of the 6th PSU-UNS International Conference on Engineering and Technology (ICET-2013), [CDROM],* Novi Sad, Serbia, May 15-17, 2013.

Venable, J. (2006). The role of theory and theorising in design science research. In S. Chatterjee & A. Hevner (Eds.), *Proceedings of the 1st International Conference on Design Science in Information Systems and Technology (DESRIST 2006)* (pp. 1-18). Claremont, CA: Claremont Graduate University.

Venable, J., Pries-Heje, J. & Baskerville, R. (2012). A Comprehensive Framework for Evaluation in Design Science Research. In K. Peffers, M. Rothenberger & B. Kuechler (Eds.), *Design Science Research in Information Systems. Advances in Theory and Practice, LNCS 7286* (pp. 423-438). Berlin, Germany: Springer Berlin Heidelberg.

Walls, J. G., Widermeyer, G. R. & El Sawy, O. A. (2004). Assessing information system design theory in perspective: how useful was our 1992 initial rendition? *Journal of Information Technology Theory and Application (JITTA), 6(2),* 43-58.

Ward, J., Daniel, E. & Peppard, J. (2008). Building better business cases for IT investments. *MIS Quarterly Executive, 7(1),* 1-15.

Wheeler, D. A. (2011, 5th August). How to evaluate open source software/free software (OSS/FS) programs. Retrieved 2014-07-23 from http://www.dwheeler.com/oss_fs_eval.html

Wieringa, R. J. (2014). What Is Design Science? In *Design Science Methodology for Information Systems and Software Engineering* (pp. 3-11). London, UK: Springer Berlin Heidelberg.

Wijers, G. (1991). *Modelling support in information systems development.* Doctoral Dissertation. Delft University of Technology.

Yin, R. K. (2003). *Case study research: Design and methods.* Thousand Oaks, CA: Sage Publications.

Zelkowitz, M. V. & Wallace, D. R. (1998). Experimental models for validating technology. *Computer, 31(5),* 23-31.

APPENDIX 1 EVALUATION CRITERIA CHECKLIST

TABLE 1.1 DSM Tool Evaluation Criteria Checklist

General	**Type**	**Range (or none)**	**Example**
Documentation	String	Set of Documentation Types	Tutorials
Customer Support	String	Set of Support Mechanisms	E-Mail
Licensing Model	String	Set of Licensing Models	Open Source
Price (in Currency X)	Int	Natural Numbers	1000
Provider	String	Set of Provider Names	Microsoft
Version	String	Set of Version Identifications	Beta 1.2
Stability	Enum	Low, Medium, Good	Low
Usability	Enum	Low, Medium, Good	Medium
Utility	Enum	Low, Medium, Good	Good
Effort (e.g. in Man Days)	Int	Natural Numbers	2
Tool Architecture	**Type**	**Range (or none)**	**Example**
Storage Mechanism	String	Set of Storage Mechanisms	Repository
Platform Support	String	Set of Platform Names	Linux
Deployment Model	Enum	Embedded, Standalone	Standalone
Extensibility	Enum	Low, Medium, Good	Low
Integration	String	Set of Integration Mech.	API
Maturity	Enum	Low, Medium, Good	Low
Meta-Metamodel Architecture	String	Set of Tool Architectures	Four-layer
Language Validation	Bool	Yes, No	Yes
Interoperability	String	Set of Standards	XMI
Maintainability	Enum	Low, Medium, Good	Low
Language Evolution	Enum	Low, Medium, Good	Low
Customizability	Enum	Low, Medium, Good	Low
Reusability	Enum	Low, Medium, Good	Low
Language Specification	**Type**	**Range (or none)**	**Example**
Metamodeling Language	String	Set of Languages	Ecore
Mutable Logical Levels	Int	Natural Numbers	2
Metamodel Syntax Specification	Bool	Yes, No	Yes
Abstract Syntax Representation	Enum	Tree, Graph	Tree
Concrete Syntax Representation	Enum	Text, Graphic, Matrix	Graphic
Concrete Syntax Style	Enum	Declarative, Imperative	Declarative
AS to CS Mapping	String	Set of Mapping Approaches	Model-Based
Semantics Specification	Bool	Yes, No	Yes
Constraint Language	String	Set of Constraint Languages	OCL
Graphical Completeness	Enum	Low, Medium, Good	Low
Context Adaptive Assistance	String	Set of Assistance Techniques	Tooltips
Workflow Guidance	String	Set of Guidance Techniques	Wizard
Relationships	Enum	Binary, N-Ary	Binary
Language Application	**Type**	**Range (or none)**	**Example**
Model Transformation Capability	Bool	Yes, No	Yes
Problem to Solution Mapping	String	Set of Mapping Techniques	Template
Transformation Def. Language	String	Set of Languages	MERL
Transformation Output Language	String	Set of Languages	Java
Generated Editor Quality	Enum	Low, Medium, Good	Low
Artefact Quality	Enum	Low, Medium, Good	Low
Output Update Mechanism	Enum	Destructive, Incremental	Destructive
Viewpoints	String	Set of Viewpoints	DSL Explorer
Analysis Capabilities	Enum	Low, Medium, Good	Low

APPENDIX 2 CHECKLIST CRITERIA DECOMPOSITION

In this appendix, the abstractions conducted in the construction of the evaluation criteria checklist for DSM tools are presented as a list of criteria decompositions. Every existing criterion is assigned a node of a specific color and a short form citation, corresponding to the piece of literature in which it was discussed. The nodes are connected to the abstracted criteria, representing the decomposition of the criteria included in the checklist. FIGURE 2.1 represents the denotation of the criteria decomposition with the assigned color mappings and short form citations. The list of criteria decompositions is presented as a multi-page illustration in FIGURE 2.2. The list was constructed by structuring the checklist concepts and relationships into a JSON document and applying the D3 JavaScript library to generate the visualization. Photocopy safe colors were utilized.

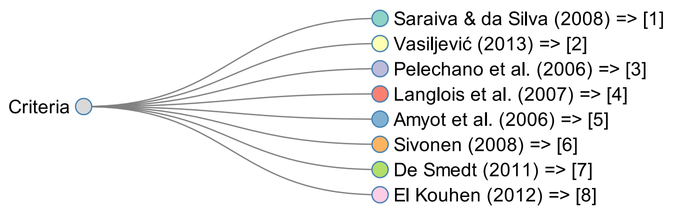
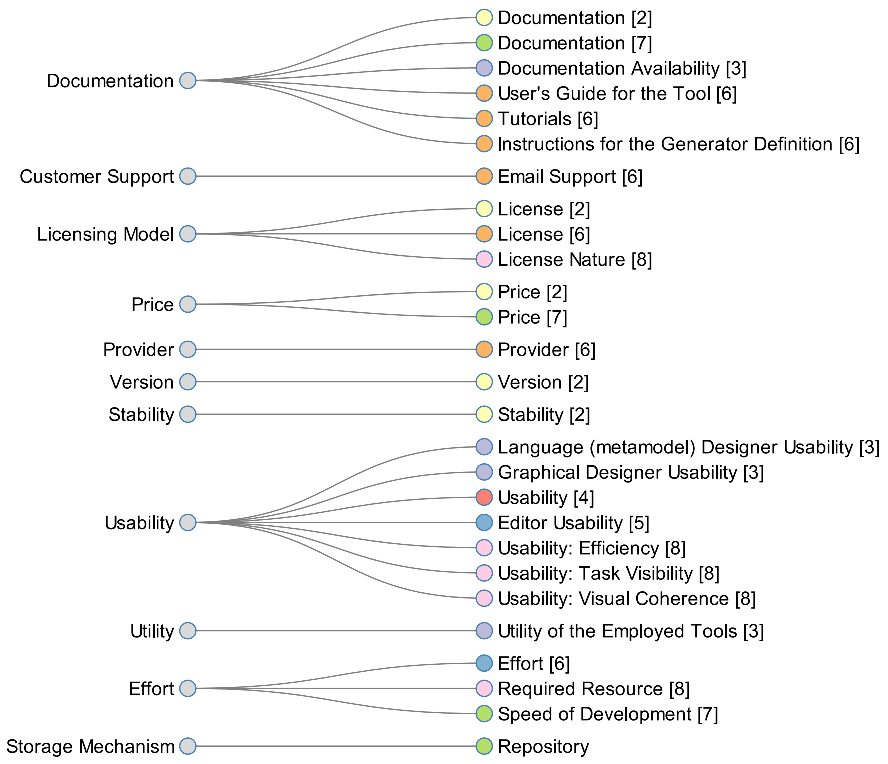
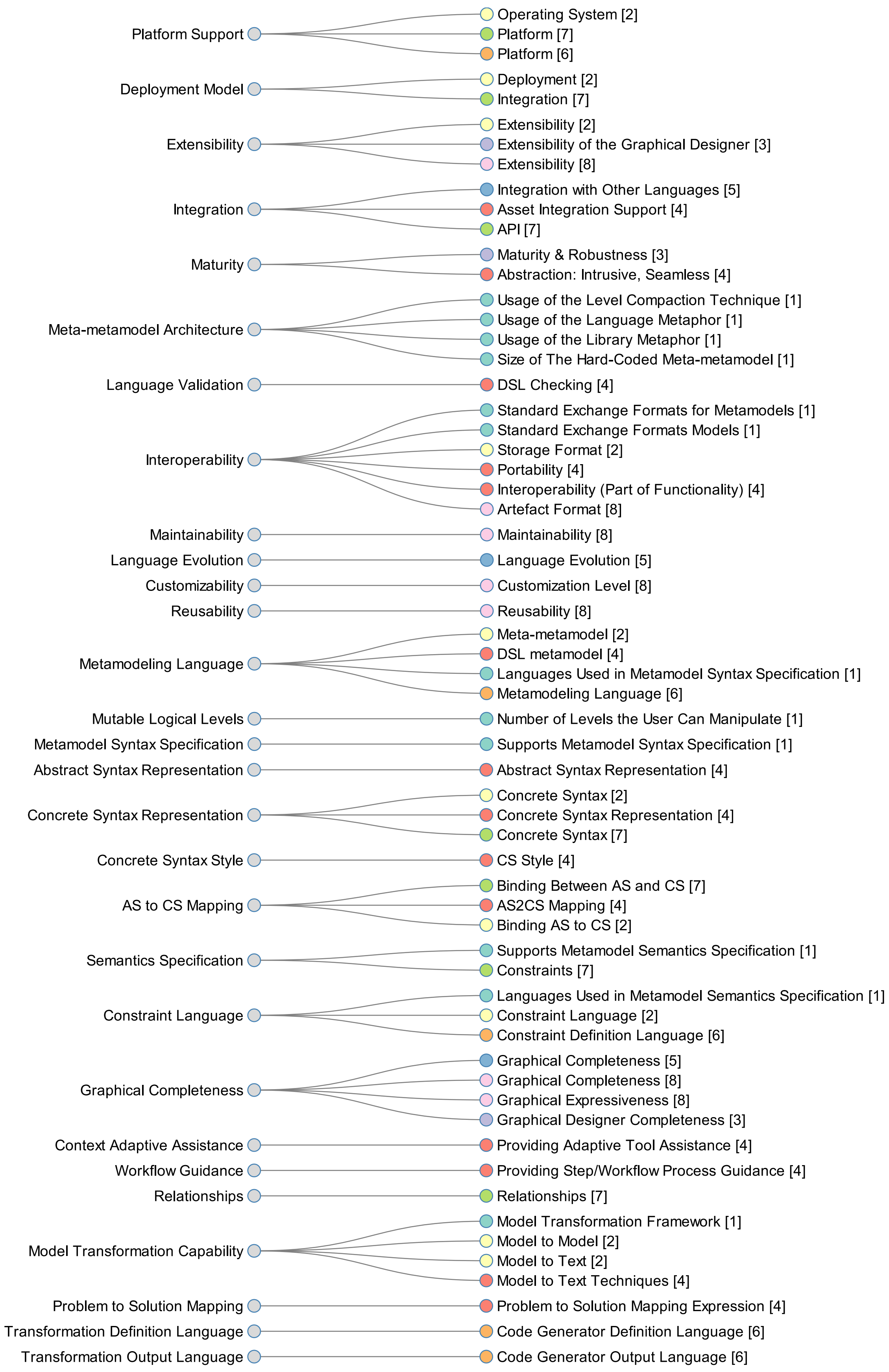

Criteria
Saraiva & da Silva (2008) => [1]
Vasiljević (2013) => [2]
Pelechano et al. (2006) => [3]
Langlois et al. (2007) => [4]
Amyot et al. (2006) => [5]
Sivonen (2008) => [6]
De Smedt (2011) => [7]
El Kouhen (2012) => [8]

FIGURE 2.1 Denotation of Evaluation Criteria Decomposition




Platform Support
Operating System [2]
Platform [7]
Platform [6]
Deployment Model
Deployment [2]
Integration [7]
Extensibility
Extensibility [2]
Extensibility of the Graphical Designer [3]
Extensibility [8]
Integration
Integration with Other Languages [5]
Asset Integration Support [4]
API [7]
Maturity
Maturity & Robustness [3]
Abstraction: Intrusive, Seamless [4]
Meta-metamodel Architecture
Usage of the Level Compaction Technique [1]
Usage of the Language Metaphor [1]
Usage of the Library Metaphor [1]
Size of The Hard-Coded Meta-metamodel [1]
Language Validation
DSL Checking [4]
Interoperability
Standard Exchange Formats for Metamodels [1]
Standard Exchange Formats Models [1]
Storage Format [2]
Portability [4]
Interoperability (Part of Functionality) [4]
Artefact Format [8]
Maintainability
Maintainability [8]
Language Evolution
Language Evolution [5]
Customizability
Customization Level [8]
Reusability
Reusability [8]
Metamodeling Language
Meta-metamodel [2]
DSL metamodel [4]
Languages Used in Metamodel Syntax Specification [1]
Metamodeling Language [6]
Mutable Logical Levels
Number of Levels the User Can Manipulate [1]
Metamodel Syntax Specification
Supports Metamodel Syntax Specification [1]
Abstract Syntax Representation
Abstract Syntax Representation [4]
Concrete Syntax Representation
Concrete Syntax [2]
Concrete Syntax Representation [4]
Concrete Syntax [7]
Concrete Syntax Style
CS Style [4]
AS to CS Mapping
Binding Between AS and CS [7]
AS2CS Mapping [4]
Binding AS to CS [2]
Semantics Specification
Supports Metamodel Semantics Specification [1]
Constraints [7]
Constraint Language
Languages Used in Metamodel Semantics Specification [1]
Constraint Language [2]
Constraint Definition Language [6]
Graphical Completeness
Graphical Completeness [5]
Graphical Completeness [8]
Graphical Expressiveness [8]
Graphical Designer Completeness [3]
Context Adaptive Assistance
Providing Adaptive Tool Assistance [4]
Workflow Guidance
Providing Step/Workflow Process Guidance [4]
Relationships
Relationships [7]
Model Transformation Capability
Model Transformation Framework [1]
Model to Model [2]
Model to Text [2]
Model to Text Techniques [4]
Problem to Solution Mapping
Problem to Solution Mapping Expression [4]
Transformation Definition Language
Code Generator Definition Language [6]
Transformation Output Language
Code Generator Output Language [6]

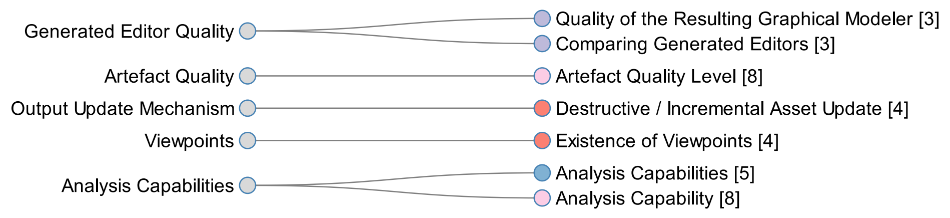

Generated Editor Quality
Artefact Quality
Output Update Mechanism
Viewpoints
Analysis Capabilities
Quality of the Resulting Graphical Modeler [3]
Comparing Generated Editors [3]
Artefact Quality Level [8]
Destructive / Incremental Asset Update [4]
Existence of Viewpoints [4]
Analysis Capabilities [5]
Analysis Capability [8]

FIGURE 2.2 Evaluation Criteria Decomposition

APPENDIX 3 BUILDING A BUSINESS CASE

A business case is perceived as a valuable instrument in many organizations for justification and evaluation of IS investments (Maes, Van Grembergen & De Haes, 2014).

> "A business case is a formal investment document with a structured overview of relevant information that provides a rationale and justification of an investment with the intent to enable well-founded investment decision-making" (Maes et al., 2014).

In the case of evaluating DSM tools for a situational context, building a tentative business case by the management could be the first step towards initiating an evaluation project. The business case may make or break the project. A business case is applicable to the evaluation of investment before, during, and after the implementation (Hitt, Wu & Zhou, 2002). Artifact II, presented in the following sections targets the evaluation of the investment in DSM tools before the implementation. A Business case is a valuable tool for evaluations also beyond the implementation, as an acquisition of DSM tools may turn out unsuccessful during or after the implementation, which requires management intervention.

Ward et al. (2007) propose a process for the development of a business case, consisting of the six following steps:

1. Define Business Drivers and Investment Objectives
2. Identify Benefits, Measures, and Owners
3. Structure the Benefits
4. Identify Organizational Changes enabling Benefits
5. Determine the Explicit Value of each Benefit
6. Identify Costs and Risks

In the first step, business drivers, i.e. the issues and challenges of the organization are identified in its internal context (e.g. a resource, process, or a condition, and) in the external context (e.g. economic conditions or trade relations). Additionally, business objectives are defined, which aim to tackle one or more of the business drivers via an investment. In FIGURE 3.1, the complete structure for a business case is presented. The outcomes of the first step are illustrated in FIGURE 3.1 as elements 1 and 2. In the second step, benefits corresponding to the business objectives are identified, with explicitly defined measures and owners, who are responsible for providing appropriate business value for each benefit, and to ensure the planning and realization of the benefits. The benefits are realized by doing new things, doing things better or stopping doing things, as presented as the element 3 in FIGURE 3.1. In the third step, a framework is developed, in which business changes that give rise to the benefits will be categorized and mapped to each other. In the fourth step, the business changes enabling the anticipated benefits will be identified and added to the framework.

Additionally, owners are assigned to the business changes to ensure commitment for benefit implementation. In the fifth step, each benefit is determined an explicit value, based on valid evidence. In the final step, all costs related to the investment as well as the risks the investment is subject to are identified, as presented by the elements 4 and 5 in FIGURE 3.1 (Ward, 2007)

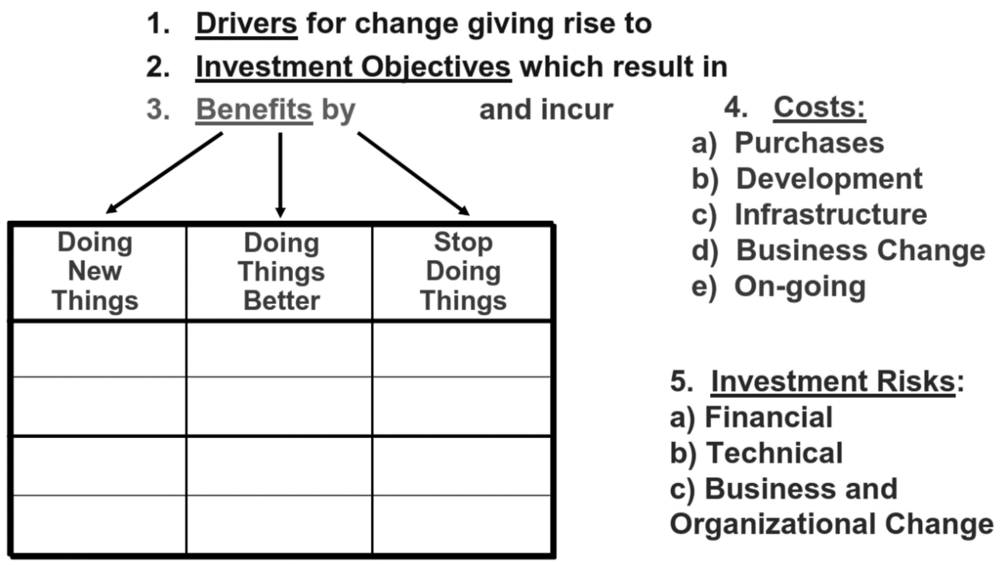


FIGURE 3.1 A Complete Business Case (Ward, 2007, p. 13)