JYVÄSKYLÄ STUDIES IN COMPUTING
130

Sergiy Nikitin

Dynamic Aspects of Industrial Middleware Architectures

JYVÄSKYLÄN YLIOPISTO

JYVÄSKYLÄ STUDIES IN COMPUTING 130

Sergiy Nikitin

Dynamic Aspects of Industrial Middleware Architectures

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Villa Ranan Blomstedt-salissa
maaliskuun 25. päivänä 2011 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Villa Rana, Blomstedt hall, on March 25, 2011 at 12 o'clock noon.

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Dynamic Aspects of Industrial Middleware Architectures

JYVÄSKYLÄ STUDIES IN COMPUTING 130

Sergiy Nikitin

Dynamic Aspects of Industrial Middleware Architectures

UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2011

Editor
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo, Sini Tuikka
Publishing Unit, University Library of Jyväskylä

URN:ISBN:978-951-39-4251-9
ISBN 978-951-39-4251-9 (PDF)

ISBN 978-951-39-4230-4 (nid.)
ISSN 1456-5390

Jyväskylä University Printing House, Jyväskylä 2011

ABSTRACT

Nikitin, Sergiy
Dynamic Aspects of Industrial Middleware Architectures
Jyväskylä: University of Jyväskylä, 2011, 52 p. (+included articles)
(Jyväskylä Studies in Computing
ISSN 1456-5390; 130)
ISBN 978-951-39-4230-4 (nid.), 978-951-39-4251-9 (PDF)
Finnish Summary
Diss.

Design and development of industrial ICT systems is becoming more and more demanding and complex task. Business requirements call for optimization of the IT-expenses seeking at the same time for long-lasting, extensible and robust solutions that would be working during the whole product lifecycle.

The continuous growth of information volumes and interdependency of systems invites the IT-practitioners to look for innovative approaches to IT-systems design and development.

The information technology will undergo drastic changes when trying to resolve the new challenges put by businesses. We have chosen the visions of Global Understanding Environment (GUN) and Autonomic Computing as key paradigms for the change and look for enabling technologies of these.

More specifically, through analysis of several industrial use cases we have identified several aspects that we deem critical for future industrial ICT systems. The key aspects are adaptation (of heterogeneous resources present in the industrial ecosystem), servicing (use of services in an open environment) and domain model sharing between different application areas. It is essential that all these aspects are tackled in a dynamic setting as unpredictable changes in the environment are characteristic for long living industrial systems. To meet the requirements of the industrial cases, we combine three separate technologies – Semantic Web, Agent Technology and Web Services as a unified engine or middleware. The developed innovative middleware platform called UBIWARE offers a language called S-APL that incorporates semantic reasoning, agent messaging and thus, servicing. In particular we introduce semantic componentization of the functionality possessed by agents and planning on top of the semantic components. With several potential industry-driven use case scenarios of the platform we demonstrate both implementability and extendibility of the approach up to augmenting the emerging cloud computing stack with semantic agent-driven middleware.

Keywords: Semantic Web, Agent Technology, Web Service, Ontology,
Middleware, Adaptation, Industrial Applications, Dynamics, Cloud Computing

Author's address

Sergiy Nikitin
Dept. of Mathematical Information Technology
University of Jyväskylä
P.O. Box 35
FIN-40014 Jyväskylä, Finland
sergiy.nikitin@jyu.fi

Supervisors

Prof. Dr. Vagan Terziyan
Dept. of Mathematical Information Technology
University of Jyväskylä, Finland

Prof. Dr. Timo Tiihonen
Dept. of Mathematical Information Technology
University of Jyväskylä, Finland

Prof. Dr. Pekka Neittaanmäki
Dept. of Mathematical Information Technology
University of Jyväskylä, Finland

Reviewers

Prof. Tatiana Gavrilova
Graduate School of Management
St.Petersburg State University
Russia

Dr. Valérie Monfort
Maître de conférences
Université de Paris1 Panthéon Sorbonne, France
ISIG Kairouan, Tunisia

Opponent

Dr. Evgeny Osipov
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Sweden

ACKNOWLEDGEMENTS

I would like to start this section with words of gratefulness to Prof. Vagan Terziyan. Without his supervision and patience this thesis would not have existed. Next I would like to thank to Dr. Olena Kaykova for her outstanding efforts in organizing the exchange program between Kharkiv National University of Radioelectronics and University of Jyväskylä and then being a supervisor and a colleague for all this time. I am grateful to Prof. Timo Tiihonen for his care and support in all means, up to the last line of this thesis.

I am also thankful to Prof. Pekka Neittaanmäki and Dr. Päivi Fadjukoff for building a great working environment and a creative atmosphere in Agora Center. I would like to thank to the department of the Mathematical Information Technology of the University of Jyväskylä as well as to the Intelligent Decision Support Systems department of the Kharkiv National University of Radioelectronics for great inspiration to study and to enjoy science.

Next I would like to thank all the "industry people" who have been our partners in various research projects, in particular Dr. Jouni Pyötsiä (Metso Automation), Dr. Henry Palonen (Inno-W oy), Veli-Jukka Pyötsiä (Fingrid Oyj) and others. Without the industrial input, this research would have never existed. I am also thankful to Dr. Oleksiy Khriyenko, for being a colleague and a friend for all this time. Someone said that "Truth is born in an argument", so with Oleksiy we have produced a lot of "truths" during these years☺.

I would like to thank to all the members of the Industrial Ontologies Group, and in particular, to Dr. Anton Naumenko and Yaroslav Tsaruk, for great scientific discussions and interesting work. I am also thankful to Dr. Artem Katasonov, Michal Nagy, Michael Cochez and Joonas Kesäniemi for their team spirit and great motivation to innovate and create something new.

I would like to express my words of gratefulness to Helen Vershinina, head of the XOBFIZIT Kharkiv charity fund for her invaluable support and care in my hardest times.

Jyväskylä-Kharkiv
February 2011

Sergiy Nikitin

LIST OF FIGURES

FIGURE 1 Alarm message integration tool 18
FIGURE 2 Data integration using Ontonuts 19
FIGURE 3 SOFIA platform architecture 21
FIGURE 4 Ubiware Agent architecture 22
FIGURE 5 UBIWARE platform architecture 23
FIGURE 6 Aspects of the abstract system architecture 26
FIGURE 7 Abstract architecture of Semantic Web Service Platform 28
FIGURE 8 Service delivery process (adopted from (ASG, 2004)) 29
FIGURE 9 Foundation Models for Semantic Web Services 30
FIGURE 10 Mathematical function as a capability 31
FIGURE 11 Dynamics as a component control channel 32

LIST OF TABLES

TABLE 1 Amount of alarm messages processed 18

CONTENTS

ABSTRACT

ACKNOWLEDGEMENTS

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF ORIGINAL ARTICLES

1 INTRODUCTION 11
- 1.1 Semantic Web 13
- 1.2 Agent Technology 13
- 1.3 Service-Oriented Architectures 13
- 1.4 Research objectives and research approach 14
- 1.5 Thesis outline 16

2 INDUSTRIAL PROTOTYPING 17
- 2.1 Metso case study 17
  - 2.1.1 A Tool for Alarm Messages Integration 17
  - 2.1.2 Ontonuts: Dynamic data integration for Metso 19
- 2.2 A case study for Forest Industry 20
- 2.3 A middleware platform 21
- 2.4 A middleware for cloud computing 23
- 2.5 Chapter Summary 24

3 DYNAMIC ASPECTS OF INDUSTRIAL MIDDLEWARE ARCHITECTURES 25
- 3.1 Dynamic Adaptation Aspect 26
- 3.2 Dynamic Servicing Aspect 27
- 3.3 Dynamic Model Sharing Aspect 30
- 3.4 Dynamics in Common 31
- 3.5 Summary 32

4 RELEVANCE TO OTHER RESEARCH 34

5 OVERVIEW OF THE ORIGINAL ARTICLES 37
- 5.1 Article 1: Querying Dynamic and Context-Sensitive Metadata in Semantic Web 37
- 5.2 Article 2: Service Matching in Agent Systems 38
- 5.3 Article 3: Data Integration Solution for Paper Industry - A Semantic Storing, Browsing and Annotation Mechanism for Online Fault Data 39
- 5.4 Article 4: Ontonuts: Reusable Semantic Components for Multi-Agent Systems 39
- 5.5 Article 5: SOFIA: Agent Scenario for Forest Industry 40

5.6 Article 6: Mastering Intelligent Clouds: Engineering Intelligent Data Processing Services in the Cloud 41
6 CONCLUSIONS 42
6.1 Answers to the research questions 43
6.2 Concerns 44
6.3 Further Research 45
YHTEENVETO (FINNISH SUMMARY) 46
REFERENCES 47

LIST OF ORIGINAL ARTICLES

I. Nikitin S., Terziyan V., Tsaruk Y., Zharko A., Querying Dynamic and Context-Sensitive Metadata in Semantic Web, In: V. Gorodetsky, J. Liu, and V.A. Skormin (Eds.): Autonomous Intelligent Systems: Agents and Data Mining, Proceedings of the AIS-ADM-05, June 6-8, 2005, St. Petersburg, Russia, Springer, LNAI 3505, pp. 200-214.

II. Naumenko A., Nikitin S., Terziyan V., Service Matching in Agent Systems, In: International Journal of Applied Intelligence, In: M.S. Kwang (Ed.), Special Issue on Agent-Based Grid Computing, Vol. 25, No. 2, 2006, ISSN: 0924-669X, pp. 223-237.

III. Nikitin S., Terziyan V., Pyotsia J., Data Integration Solution for Paper Industry - A Semantic Storing, Browsing and Annotation Mechanism for Online Fault Data, In: Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO), May 9-12, 2007, Angers, France, INSTICC Press, ISBN: 978-972-8865-87-0, pp. 191-194.

IV. Nikitin S., Katasonov A., Terziyan V., Ontonuts: Reusable Semantic Components for Multi-Agent Systems, In: R. Calinescu et al. (Eds.), Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems (ICAS 2009), April 21-25, 2009, Valencia, Spain, IEEE CS Press, pp. 200-207.

V. Nikitin S., Terziyan V., Lappalainen M., SOFIA: Agent Scenario for Forest Industry, In: Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS-2010), Funchal, Madeira, Portugal, 8-12 June, 2010, pp. 15-22.

VI. Nikitin S., Terziyan V., Nagy M., Mastering Intelligent Clouds: Engineering Intelligent Data Processing Services in the Cloud, In: Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2010), Funchal, Madeira, Portugal, 15-18 June, 2010, pp. 174-181.

1 INTRODUCTION

> It would seem that perfection is attained not when no more can be added, but when no more can be removed.[1]

The informatization of the society has been going at a high pace for several decades. The Moore's law, stated in 60's, still holds true, despite of the technological challenges faced by the research community. We experience an unprecedented growth of the amounts of information held and processed by individuals. The volumes of the information transmitted over the internet as well as persistent storage devices owned by individuals nowadays pose new challenges to the information science. The traditional informational boundaries are vanishing being substituted by Ubiquitous Computing (Weiser, 1993) trends. Efficient and exhaustive information search in the internet is getting more and more complicated every year. Even the domain of intra-organizational information management already calls for novel solutions that improve information search, sharing and reuse.

The challenge of information management is complemented by another challenge: growth of complexity of information systems. The software systems being designed nowadays have become very complex and hence very expensive to maintain (Kephart and Chess, 2003). This economical factor drives the research towards new software design and development paradigms, which would tackle the above mentioned challenges. One of such paradigms is Autonomic Computing (Kephart and Chess, 2003) envisaged by IBM. The paradigm states that the complexity of information systems management can be decreased by introducing autonomy (i.e. a certain degree of freedom and self-awareness) to system components. When a component becomes autonomous, it can observe its own state and act to maintain it or take actions to change it. Such components would take the burden off the software developers by solving the routine tasks themselves. Furthermore, such components, when orchestrated into complex processes, would keep those processes running in a more robust way because of the self-awareness, and, hence self-management capabilities.

---

[1] Antoine de Saint-Exupery, Wind, Sand and Stars, 1939

At the same time, the Information Technology infrastructure of the industrial sector is experiencing regular changes, modifications and updates. The scale of the changes may vary from maintenance and support of the existing applications, up to the revolutionary transfer to a completely new infrastructure. Industrial businesses make significant investments into the IT-tools and solutions, and expect those tools to be long-lasting and reliable. Surprisingly, the solutions they get can fulfill their expectations. However, the technological progress of both the IT-sector and the industrial sector brings new competitive possibilities and advantages. To keep the leading market position, the industrial company simply must offer an up-to-date innovative IT-infrastructure and support. In other words, the company should invest into the subsequent changes in the IT-solutions that have already been developed. Nevertheless, market economy dictates its own rules – in order to be competitive, any company must offer a competitive price. IT-solutions, therefore, should not be a price burden for industrial products, but at the same time, should be modern and reliable. When choosing an IT-solutions provider company, an industrial enterprise will consider the integral price of the required solution with respect to the expected updates in the product lifecycle. The architecture of such IT solution should be robust and scalable, yet easy to configure, update and even extend. The natural question here is – how to architect systems in order to meet the expectations of customers by keeping the reasonable price? In other words, what will be the optimal IT-architecture for industrial solutions in the future?

In order to design an optimal architecture we need to have an insight into the industrial problems and then draw a hypothetic system that would meet these problems. In this work we rely on the vision of Global Understanding Environment (GUN) (Terziyan, 2003) that contemplates the future information medium in three key aspects: Adaptation, Proactivity and Networking. The vision postulates that future information systems, networks and other resources in order to achieve a highest degree of interoperability will utilize a unified approach to information and knowledge exchange. At the same time, the resources being heterogeneous by nature, will be equipped with the software adapters to bridge the resource-specific conceptualization with the shared conceptualization of the environment. Next to the adapter, each resource is supplied with an intelligent agent - a software representative that acts on behalf of the resource in the environment. GUN is an abstraction beyond the Autonomic Computing vision that already puts requirements, restrictions and design principles for the medium as well as its components. GUN environment is an Autonomic Computing environment that follows the design pattern of GUN.

However, any vision requires the implementation ground to become true, as well as any theory becomes a good theory, when it has passed the experimental validation. When a theory needs to be practically tested, the first question we need to answer is: “Do we have a technology to support this theory?” We believe that the ambitious challenges stated by the Autonomic Computing and GUN visions could be fulfilled with the wise combination of already existing technologies. We list those technologies below.

1.1 Semantic Web

By the Semantic Web (Berners-Lee et al., 2001) we understand development towards semantic machine-processable information on the web. The Semantic Web comprises a set of formal specifications, such as RDF (Resource Description Framework) and its notations – RDF/XML, N3, N-triples, etc. for unified information representation. At the same time, Semantic Web uses RDF-Schema and OWL to formalize the knowledge domain with concepts, terms and relationships. The expressiveness of Semantic Web specifications when combined with shared domain ontologies (conceptualizations) constitutes the ground for automated machine-to-machine communication and information exchange. Within the vision of GUN, the role of Semantic Web is to guarantee the disambiguation and expressivity (explicitness) of knowledge and information.

1.2 Agent Technology

We understand Agent technology (Jennings & Wooldridge, 1998, Odell, 2000) as a set of tools, methods and techniques used for design and development of Software Agents (Jennings, 1996) and Agent Systems (Genesereth, 1994, Nwana, 1996). The Agent Technology is a subset of Software Technology, where software design and development utilizes the notion of Software Agent as a key building block of the Software Architecture. The Software Agent possesses a subset of following properties: Autonomous, Interactive, Adaptive, Proactive, Intelligent, Rational, Coordinative, Cooperative, Competitive, etc. Agent as a software component differs from the traditional software object by the additional abstraction level – called self-awareness, i.e. Software Agent is able to observe itself and act to a certain degree autonomously. Being still a software entity, agent can be understood as a software pattern with autonomy as a required characteristic. We consider Agent Technology as an engine for the Autonomic Computing vision. At the same time, we believe that true potential of Agent Technology lies in intelligence (Wooldridge, 1995).

1.3 Service-Oriented Architectures

Service-Oriented architecture (SOA) is a set of software design principles, patterns and tools used to support the development of reusable distributed software components that are loosely coupled and web-accessible via well-defined interfaces. We understand well-defined interface as a specification, sufficient enough to provide the required input and receive the expected output from the component. The specification may include the list of functions with their inputs and outputs as well as service choreography description. The terminology used

in the specification may refer to standards and schemas which makes it more precise. SOA brings the interoperability to the Autonomic Computing, i.e. makes the components universally accessible.

1.4 Research objectives and research approach

In short, the aim of this research is to demonstrate the applicability of the GUN vision (Terziyan, 2003) to the industrial problems via construction of industrial tools and methods that follow this vision. Within the construction we combine existing technologies (Semantic Web, Agent Technology and Service-Oriented Architectures) to prove the viability of those within the industrial settings as well as to see their limitations. Next we analyze the outcomes of the design and development and try to generalize the common principles that should be addressed in the construction of industrial applications.

The following research questions have emerged during the studies performed within various industry-driven research projects:

Q1: Does GUN vision apply for future industrial ICT-architectures?

Q2: Do the candidate enabling technologies meet the needs of industrial applications in the nearest future?

Q3: What are the key architectural features of tools for construction of industrial applications?

As we aimed to tackle questions addressing both long time visions and technical feasibility in the near future, we used a combination of the exploratory and the constructive research approaches that correspond to the natural and design sciences respectively. Within the research framework described in (March and Smith, 1995) the authors distinguish between the research outputs and research activities and claim that both the natural and design research activities should be applied to the IT research. The research framework presented in (Hevner et al., 2004) has elaborated a more rigor approach towards the understanding, description and evaluation of the research in IT. We align our outcome with the framework, however, in addition we refer to the notion of a vision (we may also consider it as a *hypothesis*), which was not included into the original framework guidelines, yet affected this research drastically.

The initial work on this thesis has already started within the scope of the GUN vision. At that time it was introduced as a concept, however, the abstraction level of it corresponds to the "vision" term. The vision was based on, and derived from the industrial problems by generalizing and specifying an "ideal" environment, where components of "different nature" can easily interact to achieve their goals. Within the research framework (Hevner et al., 2004) the vision was a *design artifact* with the *problem relevance* in the area of web services

and industrial applications. However, the evaluation of the vision was difficult as, the *instantiation* (application) of the vision was not yet addressed. Nevertheless, the vision had a *research contribution* in a set of design artifacts that were based on such emerging, yet already viability-proven instruments as web services and software agents, and, therefore could have been considered as rigorous. The research communication of the vision was rather clear, as it did not target a narrow expert group, but covered a domain of service- and agent-enabled industrial systems.

The next phase (iteration) of the research has aimed at the development of models and methods as *design artifacts* that would further conceptualize the GUN vision through the *instantiation* of *models*, *methods* and subsequent software prototypes. The SmartResource (Terziyan, 2007), Adaptive Services Grid (ASG, 2004) and SCOMA (SCOMA, 2005) projects were a testbed for software implementation and testing of different aspects of the GUN. Whereas the SmartResoruce was a mainstream project of GUN-related development, the ASG has allowed us to investigate deeper the semantic servicing and SCOMA helped to explore the semantic domain modeling problems. As a result, the GUN vision has populated into a set of frameworks, one of which is GAF (General Adaptation Framework) that constitutes a crucial basis for this work. Another instantiation of the vision was a SmartResource platform that within its development cycle has undergone three iterations and has provided an experimental basis for further models' and methods' development.

Within the SmartResource project the results of the Articles I, II and III of this thesis were achieved. As design artifacts, all of them have provided both methods and their practical validation via prototype implementations.

The next qualitatively new iteration of the research has started with the UBIWARE project (Katasonov, 2008) – the project has set an ambitious goal – to develop a middleware platform for industrial applications. The work presented in this thesis derives from, and contributes to the UBIWARE platform development. The artifacts presented in this thesis (Articles IV, V, and VI) are based on the UBIWARE platform implementations and extensions. The validity of the methods presented in Article IV has been tested and implemented in the industrial prototypes based on the industrial data- and system samples. The UBIWARE platform itself as an artifact has undergone 3 iterative cycles of design and development. The viability of the platform has been tested in a set of industrial implementations from different problem domains.

When considering the contribution of the Articles V and VI, the design artifacts provided are based on a well-grounded problem domain exploration. The models presented in the articles were not implemented in prototypes, yet the prototyping of those would become a natural continuation of this research.

When considering this thesis as a whole within the guidelines addressed in the framework (Hevner et al., 2004), the *problem relevance* of this work is supported by the amount of relevant topics in industrial IT as well as project-driven case studies and industry-driven implementation tasks. The *research rigor* can be justified by aligning the work with the theoretical foundations of Semantic Web,

Agent Systems as well as standards referred to and used. The *design as a search process* has covered a wide area of semantic web services, business processes, agent systems and the semantic approaches to integration of different resources. In this work we address the *design as an artifact* not only by tools implemented, but also by a key set of interweaving aspects, that were decoupled and identified from the implementation and then generalized into guidelines for design of industrial architectures. The *design evaluation* seems to be hard to perform, at least in quantitative sense, since no equivalent platform tools that would combine all the characteristics, exist at the moment. On the other hand the ability to produce unique outcomes is a strong qualitative indicator of the design. The lack of practical application within the long industrial lifecycle is another issue that hinders the evaluation. As we have utilized and extended an innovative prototype middleware platform, the evaluation has been performed from the point of view of the usability. We have also identified limitations of our platform and tools but due to the limitations we have better realized the scope and the position of these tools in the industrial applications design.

The *research contribution* of this thesis is addressed by the combination of the innovative design principles and the application of those in the prototype implementation of tools (e.g. Ontonuts engine). The aspects derived from the applications design may be considered as a hypothesis for future work on development of industrial architectures. The *research communication* is supported by technical articles published in refereed journals and conference proceedings that target the technical audience. The business and managerial audience is more addressed by the introductory part of this manuscript, as it describes the broad problem area and presents the essence, or features that future industrial middleware architecture would need to have.

1.5 Thesis outline

This work is organized as follows: in the next Chapter we present the industrial prototype tools and case studies we have performed. We highlight the key points of the design and implementation that are challenging, yet common for different problem domains. In Chapter 3 we derive the common aspects that should be addressed by the software architectures to meet the industrial problems. Chapter 4 presents the related work on the relevant research topics. A brief overview of the articles included in this thesis is provided in Chapter 5. We conclude and discuss future work in Chapter 6.

2 INDUSTRIAL PROTOTYPING

In this chapter we present practical outcomes of the research activities conducted within the projects inspired by the GUN vision. We show how different architectural solutions have incrementally led us to the identification of key architectural aspects within the industrial problem domain.

2.1 Metso case study

Metso Automation is a provider of IT-solutions for paper production lines and factories. The cooperation with Metso Automation has been mostly related to data integration solutions utilizing different tools and technologies.

2.1.1 A Tool for Alarm Messages Integration

The first industrial application we refer to in this thesis, was a web service-based alarm message integration tool (see Figure 1). The alarms, coming from the paper machine are processed by the adapter component with the web service interface that transforms the alarm SOAP-messages into the RDF-format in accordance with the ontology elaborated for the paper industry alarms domain. It also provides a web-based interface for the dynamic alarm data management where a user defines an RDF-query via simple web interface that does not require any special semantics-related knowledge from the user except the paper industry domain.

The tool has been launched as a test pilot in the year 2006 and it has been successfully running for full four years already. At the moment of writing this thesis the real data flow from one of the partner factories of Metso Automation is still being forwarded to the university server and the tool successfully performs the tasks that were set. The amount of data we have collected is given in Table 1. Although the amount of data is quite modest, still the stable utilization

of Semantic Web and Web Service technologies and tools has brought a confidence in the potential of the technologies and their combinations.

TABLE 1 Amount of alarm messages processed

Year	Amount of messages
2006	2083
2007	1770
2008	1977
2009	2646
2010 (up to 01.10)	320
Total:	8796

Each incoming SOAP-message is transformed into a set of RDF-statements that are put into the RDFS-reasoning enabled Sesame storage (www.openrdf.org). The storage is then accessed by the MessageBrowser component via adapters.

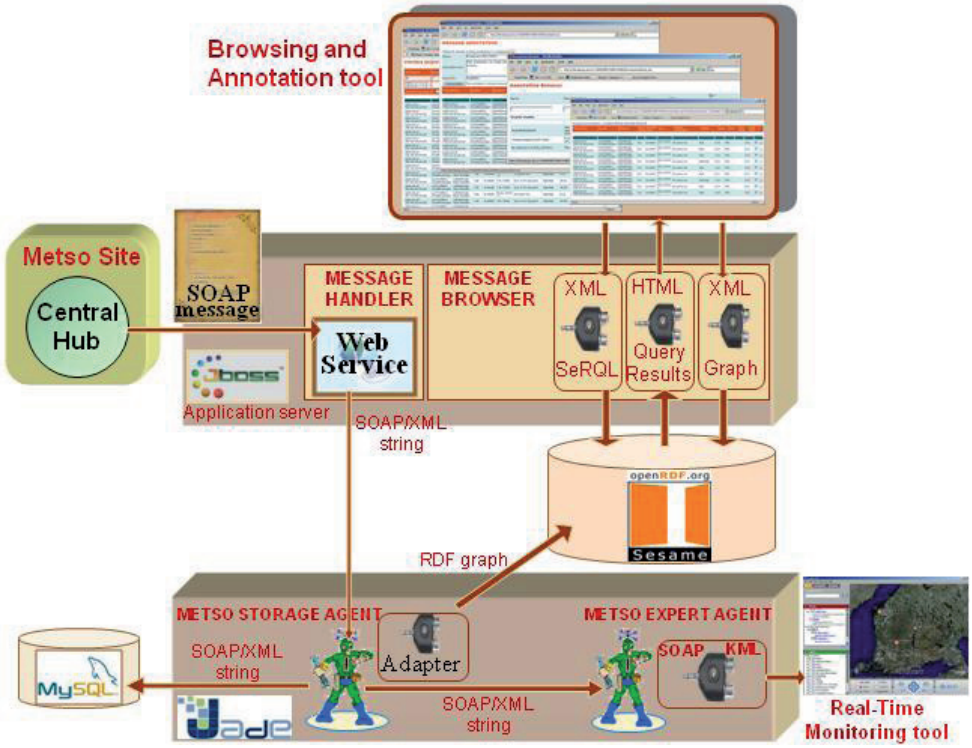

FIGURE 1 Alarm message integration tool

The performance of the whole system has proven to be stable and responsive. We have experienced the delays in query answering only when a database-backend was used for the Sesame. After shifting to the in-memory repository mode, the problem was resolved. More details about the implementation can be found in the Article III of this thesis.

2.1.2 Ontonuts: Dynamic data integration for Metso

The reality of industrial IT-infrastructure has inspired us to develop a new integration solution that would be capable to retrieve data from different data sources at the same time being dynamic. We have generalized the problem and have designed a tool and an engine that allows us to develop and reconfigure adapters to different resources through the web interface (at the moment the relational databases support is implemented). The engine incorporates a dynamic planning mechanism that resolves arbitrary goal-based requests that arise in the runtime. We named such type of adapters and the underlying technology as Ontonuts – reusable semantic components for multi-agent systems. The technology utilizes several principles that simplify the understanding and the implementation of Ontonuts. First of all any operation with the resource can be performed via capability that is defined for it. Even a database is represented by a capability (or a set of capabilities) that act as data services (see Figure 2), i.e. a user of the capability may discover it by specifying a goal request. If the goal is matched against the capability (or a set of capabilities) then the appropriate engine-supported invocation takes place.

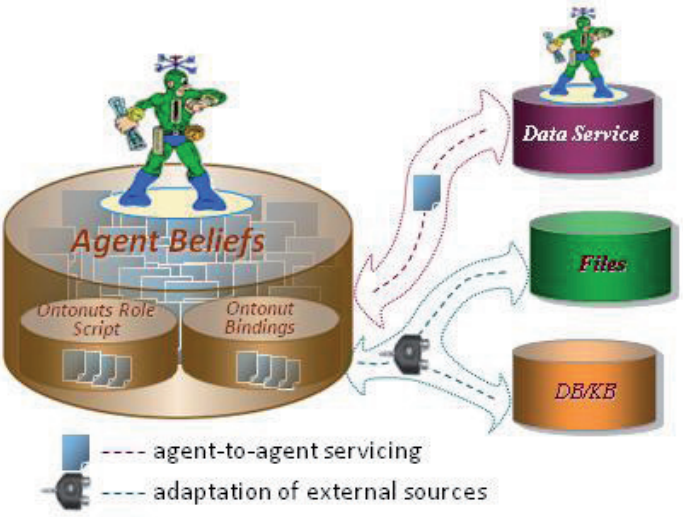

FIGURE 2 Data integration using Ontonuts

The capability specification is fairly simple and is defined in terms of inputs and outputs, e.g. database engine may perform queries and provide query results based on the request received, a service may book a flight ticket and provide a confirmation with the booking reference as an output, or software component may calculate certain mathematical function value and return it based on the input data provided. We approach all types of sources from the same perspective: what is the input required to address the source and what is the output produced by the source. We define inputs and outputs semantically, using domain ontology as a reference data model and specifying the patterns of the input/output data. The unified approach to component annotation allows

us to abstract from source types and concentrate on agent-driven component matchmaking and composition. The resource-specific extension to the capability allows declarative adapter specification that can be reconfigured later on the fly. When such a capability is published, it becomes a service.

With respect to the architectural requirements discussed in the Introduction of this thesis, we have developed a gluing technology that lays the ground for dynamic planning and component matching. At the same time, Ontonuts address the adaptation to external sources – which may be superseded by agent-driven adaptation services in the future. Nevertheless the General Adaptation Framework (GAF) (Kaykova et al., 2005) principles and techniques are adopted in the technology and design of the agent middleware-supported adaptation.

2.2 A case study for Forest Industry

This case study has been inspired by the economic situation in Finnish forest industry that desperately calls for higher degree of efficiency in all stages of the production chain. Recent research conducted in 2005-2008 has shown an extremely high degree of inefficiency in logistic operations amongst logging and transportation companies. Some of them have already realized the need for cooperative optimization, which calls for cross-company integration of existing information and control systems; but at the same time privacy and trust issues prohibit those companies from taking the open environment solutions. In (Vesterinen, 2005, Väätäinen et al., 2008, Lappalainen, 2009) new mediator-based business models were suggested that leverage the utilization and preserve current state of affairs at the same time.

We have performed a feasibility study and have designed an architecture of the IT-platform (called SOFIA) for logging and transportation subcontractors that would serve as an integrator of information systems provided from different order makers (wood buyers and forest owner associations), the orders coming from different systems would be gathered into one integrated view allowing the contractors to apply logistics optimization tools and decrease useless overheads in operation (see Figure 3).

The platform has not been implemented as a prototype; however the requirements to the architecture, that were detected in this case study, are similar to the paper industry ones: adaptation, servicing and domain modeling in dynamics. For details of this study we refer to Article V of this thesis.

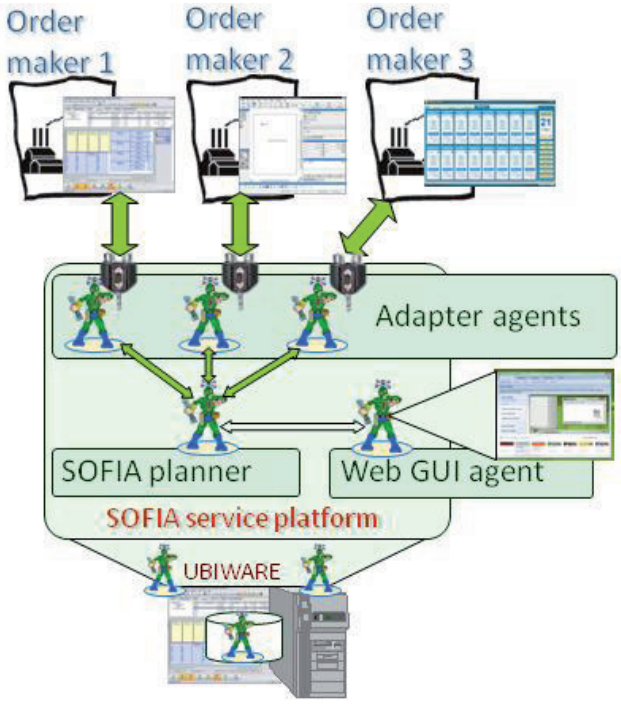

FIGURE 3 SOFIA platform architecture

2.3 A middleware platform

The GUN vision, supported by the industrial studies, has resulted in understanding of a need for such a middleware solution that would facilitate the implementation of industrial distributed systems. The research towards a middleware platform has started in 2003 and resulted in a SmartResource agent platform that was mostly utilizing a combination of existing tools and libraries. The lessons learned from the SmartResource platform have led to the complete rethinking of the platform architecture and specification of new requirements. In 2007 started the UBIWARE project (Katasonov, 2008) that aimed at the new generation middleware platform that would possess the required characteristics regardless of the limitations in the existing tools and applications. The platform architecture consists of two main parts: the architecture of a UbiwareAgent (see Figure 4), and the architecture of the platform itself.

The UbiwareAgent is a main component and a building block of the platform whereas the platform is a self-sufficient middleware, ready to run user-defined applications.

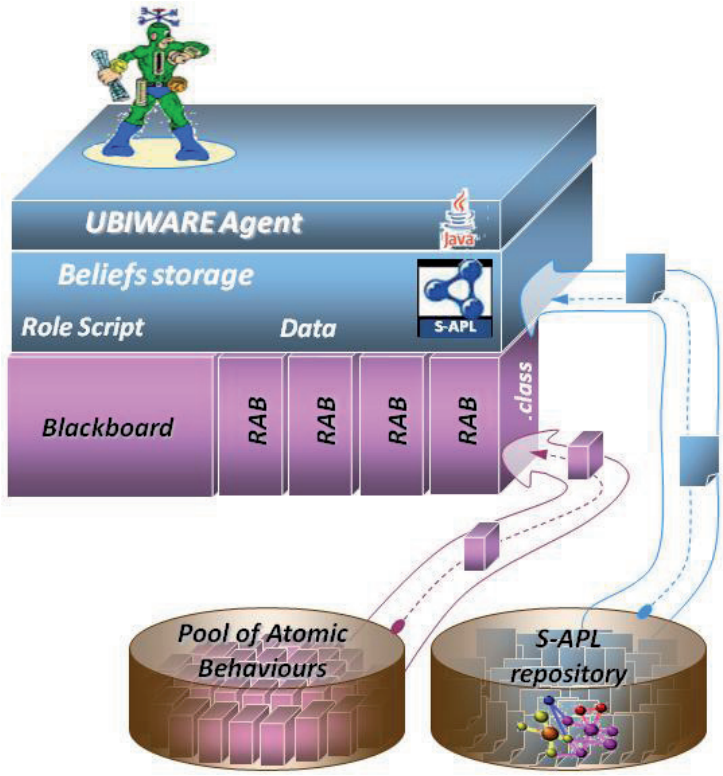

FIGURE 4 Ubiware Agent architecture

The uniqueness of the platform is grounded on the three-tier architecture of the UbiwareAgent. The topmost layer is a Live – behavior of the agent – it implements an endless cycle of agent's live activities. The lowest layer of the agent provides a set of reusable hardcoded Java-components, so called Reusable Atomic Behaviors (RABs) that an agent can use to sense and affect the environment. The middle layer is a scripting layer – where all "brain activities" of an agent take place. The script is defined using S-APL language (Katasonov, 2007, Katasonov and Terziyan, 2008), which is a unique finding of the UBIWARE project. The language is semantic and rule-based. Thus combining the features of descriptive languages like RDF (we use N3 notation as well) and programming capabilities of rule-based languages where if-then constructs can be specified. The script layer can call the layer of hardcoded components and thus interact with the "real world", e.g. another agent, web service or a device driver. However, the most interesting feature of the S-APL language with respect to the industrial challenges discussed above is the ability to produce script out of script and modify the behavior of an agent on the fly.

From the micro-world of an agent we will go to the macro-world of the platform itself. The platform architecture (see Figure 5) introduces a self-sufficient runtime environment which is supported by a set of so-called "Infra-

structure Agents" – platform agents that have a high priority and act to keep the platform running.

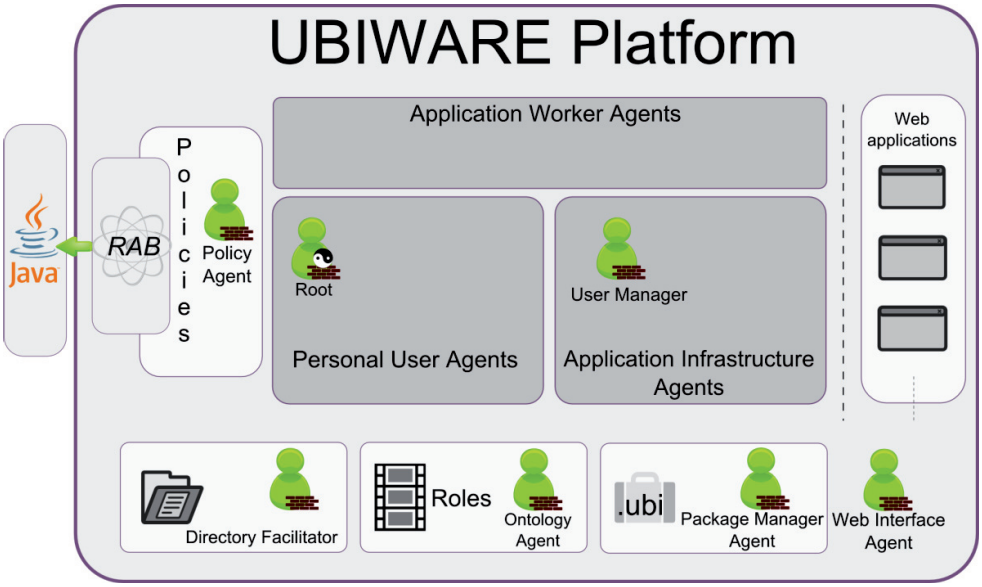

FIGURE 5 UBIWARE platform architecture

The infrastructure provides facilities for applications which are in turn driven by application agents (more details at (Terziyan, Nikitin et al, 2010)).

The key platform elements that refer to the issues of domain model sharing and servicing are the Ontology agent and the DirectoryFacilitator agent. Whereas the former takes care of the common platform-wide shared ontology of concepts and roles, the latter keeps the registry of agent-to-role bindings.

We include this section into the description of case studies because UBIWARE platform itself is an example of a complex dynamic ecosystem of simple applications that is managed by software entities (infrastructure agents) that are designed using the same principles and approaches as industrial applications being run on it.

2.4 A middleware for cloud computing

IT-world is experiencing high demand for so-called cloud computing platforms and cloud-based solutions. The cloud infrastructure is becoming more and more advanced and tailored to different user groups. The management of such infrastructure at some point will require an automated solution that would interoperate with the client applications and optimize or automatically manage the configuration of the cloud stack. In Article VI of this thesis we present an architecture that extends the cloud services stack with intelligent platform ser-

vices, at the same time providing middleware-based cloud management infrastructure. The key principles of the architecture however stay the same – the adaptation is provided as a platform service. The domain model sharing is used to guarantee successful interoperability of the in-cloud components and services, whereas the dynamics is handled by agent-enabled middleware, which provides means for the cloud stack management taking into account the user applications within it.
This case study has been important for positioning the middleware platform within the cloud computing trend and understanding the needs of cloud-based middleware solutions.

2.5 Chapter Summary

In this chapter we have presented a set of industrial prototype implementations as well as conceptual architectural solutions that aim to resolve present-day problems of industry and give an insight to potential industrial architectures of the future respectively. We utilize an innovative UBIWARE platform in implementation of test prototypes and at the same time enrich the platform itself with the generic tools that we derive and generalize from the case studies. It is important to highlight that the architecture of the platform, and especially the language used, have given us a possibility to achieve the highest degree of reusability in implementation. We develop generic hardcoded components that become a part of the platform and, therefore, incrementally enrich and speed-up the applications development. The applications, that were already developed, can address future changes caused by industrial needs by reconfiguring the flexible S-APL script layer. Moreover, the current version of UBIWARE platform lives as it preaches – an essential part of its inner functionality is provided by agents via semantically described services.

Although the UBIWARE platform has undergone three iterative development cycles and has been tested and practically used in industrial settings for 2 years already we still realize that other solutions and platforms from influential software vendors will populate the industrial software market. We can already see this tendency in the Cloud Computing area, where competitive cloud facility providers offer a rich PaaS (Platform-as-a-Service) layer to their customers. We believe that new emerging industry-oriented solutions and platforms will possess common characteristics. In the following chapter we generalize the design issues of the UBIWARE-driven development and derive key common features that will pervasively cross-cut future middleware architectures.

3 DYNAMIC ASPECTS OF INDUSTRIAL MIDDLEWARE ARCHITECTURES

When we consider the industrial IT-infrastructure, several architectural requirements take place. First of all, the architecture should offer easy connectivity to the existing systems through different API's and provide flexible mechanisms for data model mappings. Next, it should support sharing of component functionalities through well-defined, easily accessible and standardized interfaces. And at last it should provide a common ground for all system components by specifying the shared vocabulary of terms and definitions of the problem domain. All the features mentioned above, should also be considered as dynamic or, in other words, the architecture should take into account the evolutionary aspect of the system, when changes are inevitably expected but can hardly be predicted at the moment of the system startup.

We identify following aspects of the architecture in question, that need to be fulfilled in order to meet the requirements stated above:

- *Dynamic adaptation*
- *Dynamic servicing*
- *Dynamic domain model sharing*

All three aspects cross-cut the architecture and have one feature in common – they all are dynamic (see Figure 6). By "dynamic" we mean the capability to change the component or system characteristics not by reprogramming, but rather by reconfiguring them through a well-defined interface. At the same time, the component itself may initiate changes, e.g. triggered by changes in other components or the system environment.

By harnessing the above mentioned architectural principles, we can build system components that would seamlessly integrate with already existing tools and solutions, at the same time being ready to change their behavior in the future. The interoperability amongst those components is guaranteed by a well-defined shared domain model that may also grow upon the need of the environment, where the system operates.

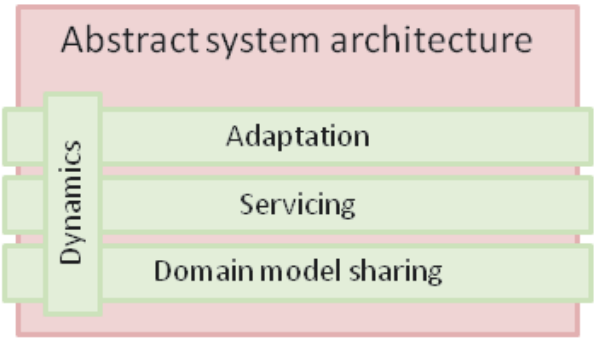

FIGURE 6 Aspects of the abstract system architecture

3.1 Dynamic Adaptation Aspect

The problem of adaptation arises from a variety of models and/or implementations constructed for the same problem domain. The models may differ in structure or in syntax. Domain models may introduce standards, or just internal system-specific data models, but nevertheless, they are described using certain metamodel. In (Naumenko, Nikitin, 2005) we discuss the advantages of the semantic metamodels compared to XML-oriented ones applied to the paper industry standardization effort.

Already existing applications and components may adapt their internal models to a shared domain model. Common binding to a shared model allows easy inclusion of the components into new interoperability scenarios.

General Adaptation Framework (GAF) (Kaykova et al., 2005) introduces several principles and concepts that are common for semantic adaptation process – the notion of the semantic adapter, the differentiation between syntactic and semantic transformation, canonical data forms, pattern-based mapping, etc. The authors differentiate three major classes of resources for adaptation, those are: humans, devices and services.

In Article I of this work we introduce a pattern-based approach to data querying, which is a particular case of data source adaptation. We build an adapter that uses query patterns to extract data from context-reach RDF graph. The idea is similar to the XSLT sheets for extracting the data from XML documents and producing the arbitrary document from the XML input given.

To simplify the adaptation, GAF introduces a two-stage transformation approach, where a notion of canonical native form is used. A canonical form can be defined, for example as XML schema. The semantic adapter development in this case can be as simple as XML-to-XML transformation, whereas the semantic transformation from the canonical XML to RDF or other semantic format may be predefined beforehand and done by domain experts. Thus the adaptation from other XML-formats would require only transforming an arbitrary XML to an XML canonical form.

Next, GAF introduces adaptation ontology – a model for semantic specification of the transformation logic. The ontology may refer to transformation patterns, their variables and thus allow dynamic configuration of the adapter.

From the industrial perspective the importance to have *dynamic adaptation* aspect in the architecture also arises from the need to tweak the systems and components during their lifecycle. Such tweaking of the components may, however, affect the business process chains, therefore building a proper dynamically adjustable adapter to the component may decrease the effort towards the process chain rearrangement, or even preserve the existing process chain by hiding the internal component changes behind the adapter.

3.2 Dynamic Servicing Aspect

Nowadays industry is actively using web services technology in distributed scenarios. Web services have brought several important advantages to the industrial world – they have decoupled the implementation of the components from their usage. The standardized method to access advertised component functionality through a message-based interface has become extremely popular in recent years as it provides common "glue" for different programming language worlds and communities. Service-oriented architecture targets the problem of interoperability amongst distributed applications and introduces standard languages for information exchange and interfaces' specification. Ease of the interoperability amongst services reduces the efforts needed for service composition and integration. However, the de-facto standards for service specification (WSDL, SOAP, etc.) are still far away from automated service discovery, composition and enactment. They rather solve the problem on the syntactic level, thus allowing everyone to speak to each other, but not to understand.

The simplicity of service creation does not imply the simplicity and ease of service consumption. On the API level "poor usability" means non-understandable interface, which may arise from poor interface specification (implicit or no description) or language incompatibility (different standard is used). The web services world have stumbled in the automated service matching challenge. We need to construct adapters in order to make external service API compliant with the shared domain model of an industrial environment, where an automated service matching and discovery would become possible. In terms of Semantic Web Services we need to provide an explicit semantic service annotation and, if needed, perform certain transformations.

As far as the majority of web service interfaces is defined using XML messaging, all the principles and techniques of GAF can be applied to construct adapters to web services.

Semantic Web Services have appeared as a technology which is expected to provide a new level of automation to the web services world. The automated composition of services has been discussed for several years already and a number of research projects were completed aiming at the development of a

sufficient infrastructure and algorithms for automated service discovery, composition and even creation (ASG, 2004, DIP, 2004, Abhijit, 2004). The idea of dynamic linking of services in order to achieve complex functionality is inspired by fast growth of the web service infrastructure, which tends to provide new flexible solutions for customers.

Below we present an abstract architecture of a software platform for semantic web services provisioning, which is aimed at fulfilling a user-defined goal. The architecture contains user interface elements and APIs for user agents (Terziyan, 2005, Veijalainen, Nikitin, 2006). The abstract platform incorporates the functionality sufficient for: discovery of services, their composition and invocation (see Figure 7). This architecture is important as it addresses the generic problem of dynamic composition starting from the goal specification, up to the process re-planning in the runtime.

Here, *Goal specification assistant* – an ontology-driven GUI-tool that helps the user to specify his/her goal explicitly. The goal is then passed to the Semantic Web service Platform.

Semantic Web Service Platform –provides dynamic automated goal-driven search, composition and invocation of web services. It embodies a Dynamic Composition component.

Dynamic composition – Performs semantic service composition; requires reasoning functionality.

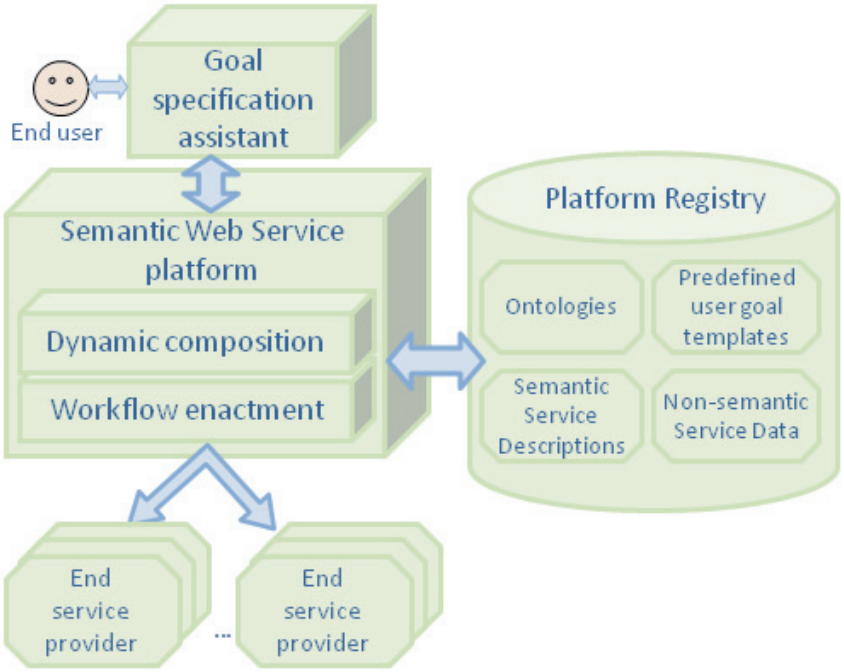

FIGURE 7 Abstract architecture of Semantic Web Service Platform

Workflow enactment –a "service player" component. This component executes composed services and achieves actual service output.

Platform Registry - A persistent storage of semantic and other service-related data

According to (ASG, 2004) the process of the service delivery consists of three main steps (see Figure 8). After user has defined his/her goal in a form of a semantic service request, the platform launches the first sub-cycle (Planning). The aim of planning is to discover suitable service(s) that reach user's goal, or to compose available services into the complex process which can fulfill the goal (the platform deals only with abstract semantic service descriptions at this point). On the second sub-cycle (Binding), the platform starts contracting and negotiation process with the service provider(s) in order to fit user preferences stated in the goal. The negotiation is done concerning QoS parameters and results in a Service-Level Agreements (SLA). The third sub-cycle (Enactment) handles the invocation of services which have signed the SLAs. It also monitors the execution process and resolves errors and failures which may occur in execution time.

In order not to run all the cycles every time a service is requested, the platform stores service compositions. Changes in service's annotation, will force the platform to reconsider all the compositions once again which might be time consuming due to renegotiation of the contracts already signed.

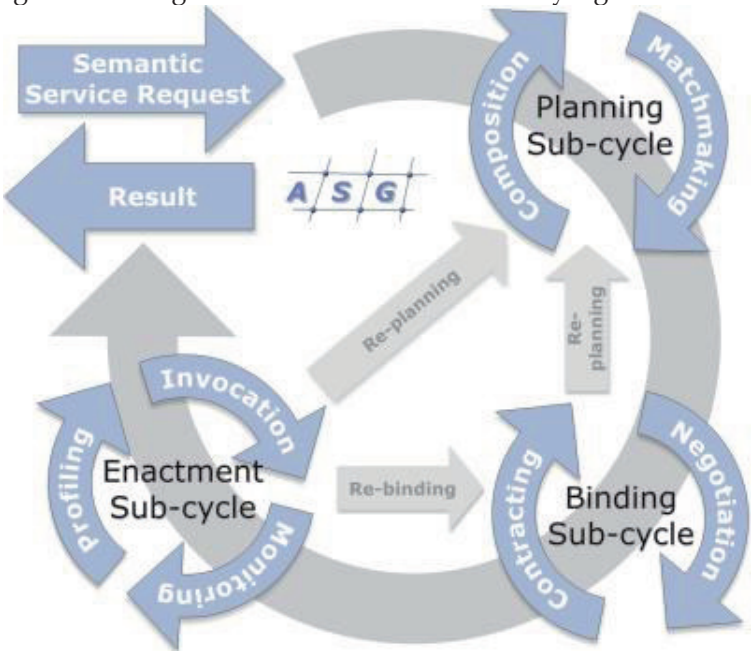

FIGURE 8 Service delivery process (adopted from (ASG, 2004))

To make a service delivery process work, we need to have:

- A semantic web service platform
- A domain ontology
- A set of services
- A set of user-defined goals

In terms of the model-driven approach, the Semantic Web Service construction requires a Service Metamodel (Service Ontology) and an Application Domain Ontology that models the domain of the discourse. The Application Domain Ontology, in turn, is a formalization of the Domain of Discourse given with the Formal Ontology Language (Jones, 1998, Rector, 2003). The Service Metamodel formalizes a Service Concept by means of Formal Ontology Language (Figure 9).

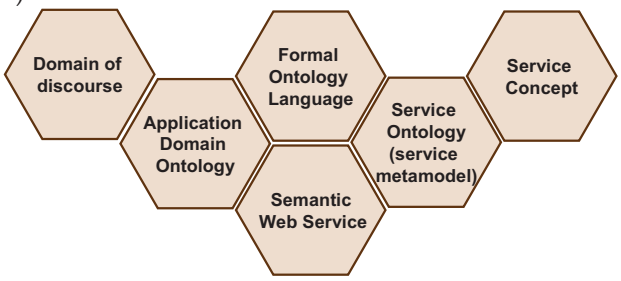

FIGURE 9 Foundation Models for Semantic Web Services

Article II of this work presents an approach for semantic service matching in agent systems and provides several hands-on methods for practical usage in matchmaking of service descriptions. Agent-driven dynamic service matching and subsequent planning are key enabling agent autonomy mechanisms. We believe that planning is a most important capability of an autonomous goal-driven component, as it allows the component to dynamically search and resolve means to achieve its goal. Following the semantic web services approach, we have developed a Ontonuts technology (Article IV of this thesis) that extends the UBIWARE platform with semantic components which are more specific compared to the semantic web services. Ontonuts combine both the internal (possessed by the agent) as well as the external (provided by other sources) capabilities and allow the agent to dynamically define goals and build execution plans to achieve these goals. In the idealistic case the reprogramming of an agent should be narrowed to changing the agent goal.

3.3 Dynamic Model Sharing Aspect

The aspect of a shared model within the abstract architecture plays a crucial role for the performance of the industrial system as a whole. The model should possess following characteristics as expressivity (ability to express domain knowledge without losses), explicitness (ability to define knowledge unambiguously) and granularity (ability to reuse knowledge definitions and exclude redundant or repeated knowledge).

A shared model should respond to the needs of the application domain. For example, to specify a semantic capability description, we may need to introduce concepts and domain-specific constructs that affect the model as a

whole. Let us consider a trivial example of a simplified capability model which implements simple mathematical function (see Figure 10).

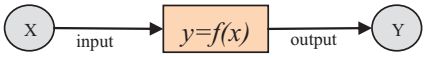

FIGURE 10 Mathematical function as a capability

Where $X \in \Re$ and $Y \in \Re$. The Y as such doesn't say anything about its provenance. We only know that it is a real number. Now, if we look from the process modeler perspective, we definitely take into account the function that has produced this number. We treat Y as a resulting value of the service function. So, in order to annotate a capability or a web service, which calculates some mathematical function in terms of Inputs and Outputs, we have to define an ontological class to specify that Y was produced by some function. Otherwise we won't be able to discover such service automatically on the planning stage. This trivial example gives a hint of what the explicit domain modeling is. The industrial domain model should be precise enough, to interpret the states and operations uniquely. The model should guarantee the possibility to formally reason and match capability inputs and outputs. In the long run, the model should allow extensions and introduction of new concepts and definitions yet keeping the whole model consistent.

3.4 Dynamics in Common

Although the industrial IT-systems in such domains as e.g. machinery, may be designed to work for several decades, still the industrial environment is becoming more and more agile and the appearance of new business processes within it can hardly be predicted for a long term. The required changes in the components are, therefore hard to predict as well. The easiest way to keep the component change-tolerant is to preprogram a dynamic behavior. *Dynamics* of a component can be introduced not as a characteristic of the component itself, but rather as a control channel over it (see Figure 11). The dynamic configuration of a component in a runtime may be performed by a controlling entity that is capable of using component interfaces, restarting it, or reversing to the previous state. This approach is opposed to the configuration through the component's own interface. Using the controlling entity to configure a component is more robust. If the configuration change affects heavily further operation of the component (for example the component becomes inaccessible), then the reverse changes may not be possible through the component interface. However, when defining an abstract control entity on top of the component, we may always observe the state of the component even if it is not functioning properly. A controlling entity may have a right to restart the component if needed.

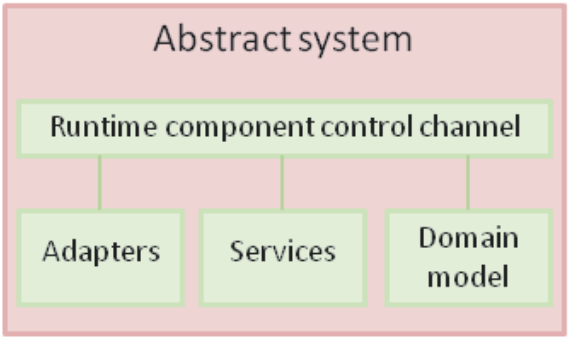

FIGURE 11 Dynamics as a component control channel

We consider software agents as a most suitable technology to implement the dynamic control channel over three main types of components – adapters, services and domain models. We derive the component types from the aspects we have defined above; however, the components may include all three aspects at the same time. As an example, there can be a semantic adapter to the social network that acts as a service within the system. The adapter uses web service interface to provide the functionality to other system components, at the same time, it uses the domain model to annotate the interface, and it also works as configurable adapter to the social network – i.e. it may modify the ontology-driven adaptation logic on the fly if the social network API changes.

3.5 Summary

IT-world is experiencing constant changes through the appearance of new technologies, approaches and visions. The amount of different programming platforms and languages has grown drastically in recent decade. If to seek for a reason of the appearance of new languages, mostly these languages and platforms are designed to simplify the construction of domain-specific applications.

In this chapter we briefly described three key aspects of abstract software architecture for industry – *adaptation*, *servicing* and *domain model sharing*. When these aspects are considered within a long timeframe, the fourth cross-cutting common aspect of *dynamics* is discussed. The aspects were derived from the prototyping and implementation work and repeated the theoretical foundations declared in the early age of agent technology establishment. The aspects, however, have undergone substantial reconsideration from the industrial applicability perspective – we enriched the understanding of these through the prism of new technological cycle, when a web service technology is widely accepted in the industrial architectures and real Semantic Web tools have become mature and their pros' and contras' are well studied. The adaptation has become a natural concept of enterprise-level architectures (e.g. Java Connector Architecture).

We believe that a next technological loop will tie these aspects together into a unified architectural model that will address industrial problems in a uniform way, offering complex solutions for development of enterprise industrial systems. In this work we offer one possible form of such integrated approach – a middleware-driven architecture that combines the above mentioned aspects in one technological platform. We have come to the conclusion that such a middleware platform will become an enabling technology, when qualitatively new tools and even supporting languages will be developed. We believe that S-APL language, despite of its immaturity is one of the hands-on examples of future programming paradigm shift towards semantics-enabled dynamic goal-driven programming.

4 RELEVANCE TO OTHER RESEARCH

This work intersects with a variety of techniques, methods and frameworks that are being actively discussed in the IT-community. The attempts to introduce new flexible approaches to the software design and development vary from using ontologies as a supporting instrument for the development (Akerman and Tyree, 2006), up to the composition of the adaptive software (McKinley et al, 2004a). In this chapter we will dedicate our attention to the mainstream approaches that address similar issues and challenges. The key research dimensions we will discuss below correspond to the key aspects we have presented, therefore we will address work on *adaptation*, *servicing* and *shared domain modeling* within the *dynamic* frame.

The *adaptation* aspect may refer to a large field of research work from topics of software product customization and configuration (Clements and Northrop, 2002), up to the runtime configuration of running software, e.g. (Keeney at al., 2003, Oreizy et al., 1999), generative programming (Czarnecki and Eisenecker, 1999) and compositional adaptation (McKinley et al., 2004a, McKinley et al., 2004b). In (McKinley et al., 2004a) the authors identify the main technologies that would enable the compositional adaptation, those are: separation of concerns, computational reflection and component-based design. The authors state the need for a middleware support (and we fully agree with this!) of the compositional adaptation. The research on the software adaptation is mainly built around the notion of Meta-Object Protocols (Kiczales et al., 1991) that enable reflection, Aspect-Oriented Programming (Kiczales et al., 1997, Walker et al., 1999), that allows the separation of concerns and Component-based software engineering (Heineman and Councill, 2001, Aksit, 2001). Our approach reuses similar principles, but rather addresses semantic component introspection through the well-defined component descriptions, i.e. we bring same problems to the unified layer of a script-like language (S-APL) but operate with semantic language constructs.

In this work, however, we mainly address the topics of semantic adaptation of external resources, which are indirectly discussed e.g. in (Canal et al., 2006). We consider resource adaptation from the perspective of the interface to

the environment, i.e. we adapt only what is needed from the resource, we do not dare (or have access) to analyze the internal resource structure, as we deal with the functional interface of the resource, and, hence the automatic interpretation of the resource logic may never be reliable.

Within the *servicing* aspect we address the related work in Semantic Web Services domain. In August, 2007 the working group of W3C consortium published as a recommendation the SAWSDL specification (Semantic Annotations for WSDL and XML Schema) (SAWSDL, 2007). The working group had been considering four candidate specification submissions – (WSMO, 2005), (OWL-S, 2004), (WSDL-S, 2005) and (SWSF, 2005). All the proposed approaches aimed at the semantic annotation which would simplify discovery and composition of services. SAWSDL specification is based on the WSDL-S approach (Verma, 2007, Abhijit, 2004) and has become an incremental step on top of the existing web services standard (WSDL, 2007) by providing an extension mechanism on top of it. SAWSDL enriches the web service component descriptions with references to semantic annotations. Those annotations can be specified using a suitable formal language, i.e. SAWSDL itself does not determine the languages for semantic specification, but rather bridges service descriptions with the formal model. For example, (Martin et al, 2007) align OWL-S with the SAWSDL specification. The candidate models that were submitted rather deeply address the semantic service specification and mainly consider Semantic Web Services within the business process context, which is absolutely reasonable assumption with respect to the aim of Semantic Web Services technology as such – to enable automated services discovery, enactment and composition. Our work does not compete in any manner with the above mentioned approaches; it rather complements the web service technology with the extensions, e.g. smart service managers (agents) that operate the service. We also consider services under the assumption of the common middleware, i.e. we exclude P2P service mappings from our scope because any external resource needs to be adapted only once within the environment. A lot of theoretical research has been conducted about the composition of the services, yet the models, tools and prototypes did not receive much of industry support, most probably due to complexity of modeling. We address similar problem of composition in Ontonuts approach (Article IV of this work), where we use backward chaining algorithm for internal agent plan composition. We keep the component annotation as simple as possible. The plan refers to components that represent external sources – databases, services, etc.

In (Clements, 1994) the author discusses how domain model affects system architecture, thus showing the tight dependency between the domain and the system. Within the Semantic Web wave, the *domain model sharing* aspect is deeply questioned in (Shadbolt et al, 2006) from the applied perspective of the Semantic Web. The authors state that a new ways for semantic data querying and sharing are needed, that corresponds to our vision of understanding the data source as a service, thus unifying the approach to composition in general

and applying the unified planning scheme to different problems (and distributed querying in particular).

An approach to configurable domain-specific service development and composition is presented in (Marin & Lalanda, 2007). The authors take into account the importance of the domain modeling and propose a model-driven development environment for service compositions. The approach is somewhat similar to our middleware-based solution, as we address the low-level development by platform tools and decouple the application logic into the S-APL level, which gives higher flexibility.

Regarding the *dynamics* aspect, the area of discourse is really broad and has been partly addressed in the adaptation-related works mentioned above in this section. The dynamic web services adaptability using AOP-based approach is discussed in (Baligand and Monfort, 2004, Ben Hmida et al., 2006). A combination of the Web Services, AOP and Agent Programming is dicussed in (Balbo and Monfort, 2009). These approaches are based on the existing SOA-tools and standards and provide practical hands-on hints on the implementation of the dynamic service behavior. In the approaches mostly non-functional aspects such as security are addressed. Our work rather complements the work mentioned by targeting the functional part of the *servicing* components and addressing the *dynamics* in goal-oriented fashion. The Gaia methodology (Wooldridge, 2000) for agent-oriented systems design, although giving a rigorous foundation for development of agent systems, still provides certain assumptions, that in our opinion, limit the applicability of agent up to such extent, that role of an agent is diminished and agent as a software design pattern becomes unnecessary. In our opinion, the true autonomy of an agent can be reached only by introducing intelligence to the agent behavior. The agent as an entity becomes valuable, when it has a unique instance-specific configuration that may be e.g. a result of learning through the experiences collected, or a goal-driven dynamic plan construction and validation (dynamic composition), in other words, an agent should obtain unique characteristics that qualitatively raises it above the understanding of being just a software component.

5 OVERVIEW OF THE ORIGINAL ARTICLES

This chapter provides a short overview of the articles included in this thesis. One of the articles was published in a journal, other five where published and presented on the international conferences.

5.1 Article 1: Querying Dynamic and Context-Sensitive Metadata in Semantic Web

Nikitin S., Terziyan V., Tsaruk Y., Zharko A., Querying Dynamic and Context-Sensitive Metadata in Semantic Web, In: V. Gorodetsky, J. Liu, and V.A. Skormin (Eds.): Autonomous Intelligent Systems: Agents and Data Mining, Proceedings of the AIS-ADM-05, June 6-8, 2005, St. Petersburg, Russia, Springer, LNAI 3505, pp. 200-214.

This article describes an approach to construction of complex semantic context-rich structures. The approach can be beneficial in fast development of semantic adapters to various resources. The patterns allow rich, yet simple transformation from the original data format, to the extensive semantic description, which is further utilized in various agent-driven scenarios. The reference implementation utilizes the RDF language as an output semantic format. The RDF being generated utilizes a state-condition extension to the standard model, i.e. the structure of the output document is not readable for a human, therefore mapping the structures of input and output format would become a challenge. The application of patterns has allowed us to concentrate on the functional part of the adapter and has appeared to be a most viable solution in the long run, with relatively simple support and easiness of processing.

From the industrial perspective, the approach offers a simple solution to the developers of semantic adapters and follows an adapt-on-demand philosophy in application design and development. The article mainly targets the *adaptation* and *model sharing* aspects, whereas the *dynamics* aspect can be addressed

by managing a configuration of the patterns and transformation logic of the adapters.

This article was written by the SmartResource project team (Industrial Ontologies Group). Yaroslav Tsaruk has contributed to the Sections 2.1, 2.2 and 3.1 (Joseki RDF storage). Andriy Zharko and Vagan Terziyan have contributed to the introduction and the editing of the final draft. The Sections 2.3 (RDQL language), 3.2 (Applying RDQL to RscDF querying), 3.3 (Querying patterns) where written by the author of this thesis.

5.2 Article 2: Service Matching in Agent Systems

Naumenko A., Nikitin S., Terziyan V., Service Matching in Agent Systems, In: International Journal of Applied Intelligence, In: M.S. Kwang (Ed.), Special Issue on Agent-Based Grid Computing, Vol. 25, No. 2, 2006, ISSN: 0924-669X, pp. 223-237.

This work addresses several issues of the service matching problem. We analyze the service matching algorithm of a JADE agent system and prove that it does not work adequately. Next we target the problem of uncertain service matching as users of agent system may not always specify precisely the goal they would like to achieve. We demonstrate several approaches to the matching of semantic definitions that employ distance measure in finding a closest service to the goal specified. The approaches address hierarchy-based and facet-based distance measure methods and show hands-on examples of distance calculation.

This work refers to the aspects of *servicing* and *domain model sharing*. We also address *dynamics* by means of agent-driven service search and invocation. Within the scope of this thesis, this article explores practical aspects of search, advertisement and matching of capabilities possessed by an autonomous entity.

The article was written within the SmartResource project. Anton Naumenko has contributed to the Sections 2 and 3 – the analysis of the FIPA matchmaking algorithm and the taxonomy-based distance measure. Vagan Terziyan has contributed to the classification of the semantic distance measure functions. Section 4 (distance measure for ontology with facets) was written solely by the author of this thesis. A minor contribution to the ontology design for Section 3 as well as the final editing of the article as a whole was also done.

5.3 Article 3: Data Integration Solution for Paper Industry - A Semantic Storing, Browsing and Annotation Mechanism for Online Fault Data

Nikitin S., Terziyan V., Pyotsia J., Data Integration Solution for Paper Industry - A Semantic Storing, Browsing and Annotation Mechanism for Online Fault Data, In: Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics (ICINCO), May 9-12, 2007, Angers, France, INSTICC Press, ISBN: 978-972-8865-87-0, pp. 191-194.

The article presents architecture and a pilot solution that utilizes semantic web, web services and agent technology to build a web-based application for paper machine experts that deal with online alarm and fault data. The system we have built, decouples the data collection and management mechanism apart of the expert GUI-based tool. The importance of this contribution is in practical utilization of Semantic Web tools in the construction of close-to-production prototypes. The utilization of new technology and decoupling of the semantic information storing from the user-oriented application has significantly changed our understanding of the semantics and the utilization of semantic content in general. The possibility to incrementally enrich and extend the semantic content provides a huge potential to the modification and improvement of the applications in the long run.

This work addresses all three aspects (*adaptation*, *servicing* and *domain model sharing*) as well as *dynamics* in the full scale.

The article was written within the SmartResource project. Prof. Vagan Terziyan and Dr. Jouni Pyötsiä have supervised the writing from the scientific and industrial perspectives respectively. The author of this thesis is a principal contributor to this article.

5.4 Article 4: Ontonuts: Reusable Semantic Components for Multi-Agent Systems

Nikitin S., Katasonov A., Terziyan V., Ontonuts: Reusable Semantic Components for Multi-Agent Systems, In: R. Calinescu et al. (Eds.), Proceedings of the Fifth International Conference on Autonomic and Autonomous Systems (ICAS 2009), April 21-25, 2009, Valencia, Spain, IEEE CS Press, pp. 200-207.

In this paper we introduce an engine that extends the UBIWARE platform with the possibility to define semantic components in a declarative way. Due to the specifics of the platform the focus and the engine support is put on the components that connect to data sources. The declarative definition of a component allows us to define new components on the fly as well as reconfigure the exist-

ing ones. In the example given we show how a distributed query can be dynamically planned using the semantic capability definitions of data sources. The engine developed for the platform uses backward chaining reasoning algorithm and builds execution plans out of available components taking into account the specifics of the platform language – S-APL. The language extensively uses the notion of containers that makes matchmaking process more complicated. This work also uses pattern-based definition of the component inputs and outputs, thus allowing free-form triple-based semantic constructs.

The paper contributes to all the aspects discussed in this thesis. In particular, the *adaptation* aspect is addressed by engine-supported declarative component definition; the *servicing* aspect is addressed by the possibility to externalize a component, i.e. to make an agent service, the *domain model sharing* is intrinsic to this work as all component definitions are semantic, as well as the ontology of the engine itself. The *dynamics* is the most prominent aspect as the components can be easily created and/or modified on the fly either through web-based GUI, or even generated by the agent itself.

The approach presented in this paper offers a key functionality for the development of middleware-supported autonomic components – a possibility to dynamically plan goal-driven agent activities.

The article was written within the UBIWARE project. Prof. Vagan Terziyan has provided a scientific supervision. Dr. Artem Katasonov has contributed to the Section 3 (UBIWARE platform). The author of this thesis is a principal contributor to this article.

5.5 Article 5: SOFIA: Agent Scenario for Forest Industry

Nikitin S., Terziyan V., Lappalainen M., SOFIA: Agent Scenario for For-est Industry, In: Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS-2010), Funchal, Madeira, Portugal, 8-12 June, 2010, pp. 15-22.

This work presents a case study and an agent scenario for the Finnish Forrest Industry. The motivation of this research has originated from the in-depth business analysis and a simulation that has proven high inefficiency of logistics operations in Finnish forestry sector. We have performed a technical analysis of the current ICT-infrastructure of harvesting and transportation subcontractors and have suggested architecture of the IT-platform called SOFIA that would address the needs of the forestry SMEs in their planning and order management. The main targets identified by this study are – integration of the existing IT-systems (*adaptation* and *domain model sharing* aspects), provision of a centralized web-based platform (*servicing* aspect) and a dynamic inclusion of new stakeholders and new software into the platform support (aspect of *dynamics* within the *adaptation*).

The article was written within the UBIWARE project. Prof. Vagan Terziyan has provided a scientific supervision and Dr. Minna Lappalainen has contributed to the business model analysis. The author of this thesis is a principal contributor to this article.

5.6 Article 6: Mastering Intelligent Clouds: Engineering Intelligent Data Processing Services in the Cloud

Nikitin S., Terziyan V., Nagy M., Mastering Intelligent Clouds: Engineering Intelligent Data Processing Services in the Cloud, In: Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics (ICINCO-2010), Funchal, Madeira, Portugal, 15-18 June, 2010, pp. 174-181.

This paper offers an innovative architecture of the cloud platforms. The agent middleware-supported adaptation and autonomy of components can be offered as platform-level services of the cloud infrastructure. The paper also demonstrates how new type of mathematical computational services may support a declarative service definition and even configuration.

The work addresses the aspect of *servicing* within the cloud infrastructure, at the same time the aspect of *adaptation* and *domain model sharing* is addressed by offering agent-driven adapters as platform services. The *domain model sharing* aspect is also used in the declarative mathematical service model specification that unambiguously defines the logic of the service, not only its inputs and outputs. The *dynamics* aspect is used within the whole architecture, as all the extensions suggested are agent-enabled and therefore proactive.

This paper ties together the ideas and efforts presented in the previous articles and offers middleware-enabled extensible cloud architecture.

The article was written within the UBIWARE project. Prof. Vagan Terziyan has provided a scientific supervision. Michal Nagy has contributed to the Section 2 (State of the Art). The author of this thesis is a principal contributor to this article.

6 CONCLUSIONS

The main contribution of this work is alignment of practical industry-driven problems with theoretical foundations declared in visions of Autonomic Computing, Global Understanding Environment and Agent Technology. We start this work with implementation of pilot industrial applications and finalize it with the derivation of abstract architectural aspects that conform to the visions mentioned. This work provides an architectural and technological prism for an idea–to-practice transformation. We address industrial applications development using Semantic Web and Agent technologies within the scope of the Web Services and Cloud Computing trends.

As generalization of the design outcomes we derive three key aspects: *adaptation*, *servicing* and *domain model sharing*. We crosscut these three aspects with the fourth common aspect of *dynamics*. In our opinion, future industrial architectures will incorporate various combinations of those. The most convenient form to support aspect-oriented software design and development will be middleware platform solutions. The crucial role in the middleware implementation will play a programming language as it will determine the foundations of the platform as a whole. We believe that S-APL language used in the implementation part of this thesis is a good sample to consider for instrumentation of enterprise-level middleware platforms and languages of the future.

As a summary, we offer an innovative architecture that combines the technological solution (a middleware platform) with the cloud infrastructure to enable a qualitatively new class of cloud software applications. These applications are tied by the common environment, where the guaranteed level of interoperability can be reached, at the same time the tools and means for such applications development are offered by the cloud provider infrastructure on different layers. In such cloud eco-system we also address the importance of component autonomy through the goal-driven behavior which can be enabled when the environment provides a consistent playground for safe (error-prone) implementation of semantic planning and composition.

6.1 Answers to the research questions

Below we provide the answers to the research questions stated in Section 1.4.

Q1: Does GUN vision apply for future industrial ICT-architectures?

The future industrial applications will address the challenges of interoperability, complex systems management and *servicing* in a dynamic setting. The GUN vision suggests a set of high-level architectural design patterns to meet these challenges. In order to prove the viability of the vision, we have constructed a set of proof-of-concept prototypes that utilize various technologies in the implementation of the GUN-inspired architectural blocks – adapters, resource agents and agent services. When we compare GUN with other visionary approaches, it is hard to judge which particular vision suits better for industry, as most of the visions discussed nowadays cover complementary parts of the problem domain (e.g. a new Smarter Planet[2] initiative from IBM stating that intelligence is being infused into the various systems and processes that make the world work). In the nutshell these visions lead to the same aspects to be taken into account when designing the applications. These aspects we discuss in answer to the Q3. We can conclude that GUN vision as such applies well to the industrial problems, yet it does not give a prescriptive methodology of how to resolve them. It rather tells from which perspective to approach the problem domain in a future-proof way.

Q2: Do the candidate enabling technologies meet the needs of industrial applications in the nearest future?

Amongst technologies and tools available on the market at present, SOA, Agent Technology and Semantic Web independently from each other have proven to be applicable to industrial problems. . Although only SOA is truly in the current mainstream of the industrial ICT development, still both the Agent Technology and Semantic Web have taken their niche on the ICT market. The amount of well-supported and stable tools (both commercial and open source) for all of the above mentioned technologies is available and is growing.
In this thesis we state that a wise combination of these technologies will bring a synergetic add-value to the industrial ICT. We also think that the technologies chosen are the most viable ones as they provide rigorously explored and analyzed contributions to the topics of *dynamics*, self-awareness, intelligence, *servicing* and *domain modeling*. Yet we believe that these technologies will become more beneficial when they are put to a common ground, e.g. a middleware and a language that combines these technologies, thus enabling their simultaneous use. In our case studies we use a sample of such language called S-APL. We also extend the platform and utilize S-APL language to combine domain modeling,

[2] ibm.com/smarterplanet

servicing and dynamic adaptation in development of agent-driven semantic components (Ontonuts) that provide a ground for planning functionality and, hence, a basis for development of true dynamic goal-driven agent behavior.

Q3: What are the key architectural features of tools for construction of industrial applications?

The key architectural features of future industrial applications are discussed as an outcome of case studies conducted. Based on the pilot implementations and industrial applications design studies, we derive common aspects that should be addressed by both the enterprise level architectural principles as well as within the internal component design. Those are: *adaptation*, *servicing* and *domain model sharing*. All three are cross-cut by the fourth aspect of *dynamics*. We claim that these aspects can be addressed in a best way by a middleware platform tool and a respective platform language that supports the implementation of platform-driven components and applications. The aspects are also considered in the context of cloud computing trend that poses additional architectural enhancements for the middleware platform.

6.2 Concerns

We propose to apply several technologies beyond their current domain of application and moreover, to combine them in new types of scenarios. Hence several concerns about the feasibility of the approach arise. So we have to address both the limits of performance of the individual technologies as well as the performance of the complex scenarios.

In the agent world the criticism mostly raises the problems of agent intelligence and then arguably small benefit of software agents compared to other software development paradigms and principles. We think that a pragmatic utilization of a goal-driven behavior is possible within the industrial scope where the "universe" is limited to the union of a finite set of domain models.

The Semantic Web has been criticized in the direction of utopia of having a common vocabulary for everybody. The concept has evolved into the new notion of Linked Data, at the same time trying to address the problems of model-to-model adaptation, thus putting the focus on the environment-specific domain models that are realistic to apply even nowadays (see Article III of this thesis). Semantic Web usage raises the problem of in-depth alignment of all system tools to a one common model, which can be considered as a programming technique or pattern that is similar to the usage of domain-specific standards.

The practical implementation of all types of scenarios may not be efficient and sometimes not even feasible using the Agent Technology and Semantic Web only, especially when high computational performance is needed. Therefore we put the application scope of the technology to be rather a cost efficient gluing solution to manage the great diversity of interoperability challenges, but

not a panacea from all IT-problems. We address the problems of service-level integration, adaptation and business process management that correspond to the aspects derived.

6.3 Further Research

Within the research framework (Hevner et al, 2004) this work is mainly based on the instantiation of design artifacts and practical testing of the technologies driven by the industrial problems. Yet, the generalization of the common architectural aspects highlights the key issues that need to be addressed more deeply in the future development of the middleware architecture, in particular those are:

- Inter-middleware adaptation and management. This issue addresses the incorporation of several domain-specific middleware solutions into the industrial architectures, e.g. RFID-middleware, VoIP-middleware, etc. We will direct our future research efforts towards an inter-middleware management, especially in the context of cloud computing.
- Addressing the ontology evolution within the organization. This issue needs to be researched to ensure that an Ontology as an instrument can safely evolve and incorporate changes during the organization's life cycle, at the same time keeping the consistency and enabling the automated goal-driven behavior of software components
- Intelligence-as-a-Service: This issue will address the automated learning and proactive goal-driven planning in the context of middleware-supported semantic agent environment of an organization, or industry-specific cross-organizational eco-system

The purpose of this work and future research that will derive from it is to direct the industrial-IT development towards the practical and feasible aspects of the innovative IT-visions that bring industrial IT-systems to the qualitatively new level.

YHTEENVETO (FINNISH SUMMARY)

Dynaamiset piirteet teollisuuden väliohjelmistoarkkitehtuureissa

Teolliseen käyttöön tarkoitettujen tietojärjestelmien suunnittelu ja kehittäminen on yhä haastavampaa ja monimutkaisempaa. Liiketoiminnallisesti kestävin kustannuksin tulisi rakentaa pitkäkestoisia, varmatoimisia ja laajennettavia järjestelmiä, jotka säilyttävät toimintakykynsä tuotteen koko elinkaaren ajan.

Käsiteltävien tietojen määrän kasvu ja osajärjestelmien yhä tiiviimpi vuorovaikutus haastavat järjestelmäkehittäjiä innovatiivisiin suunnittelu- ja toteutusmenetelmiin.

Voidakseen vastata liiketoiminnasta nouseviin haasteisiin tietojenkäsittelyteknologian on uudistuttava laadullisesti. Tässä työssä tutkitaan uusia globaalisti ymmärtävän ympäristön ja autonomisen laskennan paradigmoja ja niiden teknologista kypsyyttä mahdollisina ratkaisuina teollisuuden tulevaisuuden haasteisiin.

Useita konkreettisia käyttötapauksia analysoimalla olemme tunnistaneet useita piirteitä, jotka ovat mielestämme kriittisiä tulevissa teollisuuden tietojärjestelmissä. Näitä ovat teollisessa toimintaympäristössä olevien heterogeenisten resurssien adaptaatio, palveluorientoituneet ohjelmistot avoimessa toimintaympäristössä ja paikallisten tietomallien jakaminen ja yhteensovittaminen eri sovellusalueiden kesken. Kaikkia näitä piirteitä on käsiteltävä dynaamisina, koska teollisilla sovelluksilla on pitkä elinkaari, jonka aikana ympäristö väistämättä muuttuu ennakoimattomalla tavalla.

Vastataksemme teollisuuden vaateisiin olemme yhdistäneet kolme erillistä teknologiaa, semanttisen verkon, agenttiteknologian ja verkkopalvelut yhtenäiseksi väliohjelmistoksi. Kehitetty innovatiivinen alusta, UBIWARE, tukee S-APL kieltä, joka yhdistää semanttisen päättelyn, agenttien kommunikaation ja palveluarkkitehtuurin (SOA). Erityisesti agenttien toiminnalliset kyvykkyydet on kapseloitu semanttisiksi komponenteiksi, joiden avulla agenttiyhteisö voi suunnitella ja koordinoida toimintaansa.

Valitun lähestymistavan toteutettavuutta ja laajennettavuutta on testattu kehitetyn alustan avulla usealle teollisuuden motivoimalle käyttötapausskenaariolle aina semanttisesti tuettuun pilvilaskenta-arkkitehtuuriin saakka.

REFERENCES

Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, Kunal Verma, Meteor-s web service annotation framework, Proceedings of the 13th international conference on World Wide Web, May 17-20, 2004, New York, NY, USA doi:10.1145/988672.988747

Aksit, M. (ed.), Software Architectures and Component Technology: The State of the Art in Research and Practice, Kluwer Academic Publishers, 2001.

Akerman, A., Tyree, J., Using ontology to support development of software architectures. IBM Systems Journal 45(4), 813–825 (2006)

ASG - Adaptive Services Grid, 6th Framework Programme project funded by the European Commission, 2004-2007, http://asg-platform.org/

Balbo, F., Monfort, V., Improving Web Services Adaptability Thanks to a Synergy between Aspect Programming and a Multi-agent Middleware, In: IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 422-425, 2009.

Baligand, F., Monfort, V., A Concrete Solution for Web Services Adaptability using Policies and Aspects, In Proc. of the International Conference on Service-oriented Computing, 2004.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001) The Semantic Web, Scientific American, Vol. 284, No. 5, pp. 34-43.

Ben Hmida, M., Tomaz Feraz, R. and Monfort, V., Applying AOP concepts to increase Web Service Flexibility, in JDIM journal, ISSN 0972-7272, Vol.4 Iss.1 (2006).

Canal, C., Murillo, J. M. and Poizat, P., Software Adaptation, L'Objet., 12(1):9–31, 2006. Special Issue on Software Adaptation.

Clements, P., From Domain Models to Architectures, USC Center for Engineering, Focused Workshop on Software Architectures, June 1994.

Clements, P., Northrop, L., Software Product Lines - Practices and Patterns, Addison-Wesley, 2002.

Czarnecki, K. and Eisenecker, U., Generative Programming: Methods, Techniques and Applications, Addison-Wesley, 1999.

DIP-Data, Information, and Process Integration with Semantic Web Services, Integrated FP6 Project, EU's IST programme, 2004-2006, http://dip.semanticweb.org/

Genesereth, M.R., Ketchpel, S.P., Software Agents, Communications of the ACM 37(7), pp. 48-53., 1994.

Heineman, G., and Councill, W., Component-based Software Engineering, Putting the Pieces Together. Addison Wesley, 2001

Jennings, N., Wooldridge, M., Software agents, IEE Review , vol.42, no.1, pp.17-20, 18 Jan 1996, doi: 10.1049/ir:19960101

Jennings, N.R., Wooldridge, M., (Eds.), Agent Technology: Foundations, Applications and Markets, Springer, Berlin, 1998.

Jones, D. M., Bench-Capon, T. J. M., Visser, P. R. S., Methodologies for Ontology Development., Proceedings IT&KNOWs, Budapest, 1998.

Kaykova, O., Khriyenko, O., Kovtun, D., Naumenko, A., Terziyan, V., Zharko, A., General Adaption Framework: Enabling Interoperability for Industrial Web Resources, In: International Journal on Semantic Web and Information Systems, Idea Group, ISSN: 1552-6283, Vol. 1, No. 3, July-September 2005, pp.31-63.

Katasonov, A., Terziyan, V., SmartResource Platform and Semantic Agent Programming Language (S-APL), In: P. Petta et al. (Eds.), Proceedings of the 5-th German Conference on Multi-Agent System Technologies (MATES'07), 24-26 September, 2007, Leipzig, Germany, Springer, LNAI 4687 pp. 25-36.

Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., Terziyan, V., Smart Semantic Middleware for the Internet of Things, In: Proceedings of the 5-th International Conference on Informatics in Control, Automation and Robotics, 11-15 May, 2008, Funchal, Madeira, Portugal, ISBN: 978-989-8111-30-2, Volume ICSO, pp. 169-178.

Katasonov, A., Terziyan, V., Semantic Agent Programming Language (S-APL): A Middleware Platform for the Semantic Web, In: Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC-2008) / International Workshop on Middleware for the Semantic Web, August 4-7, 2008, Santa Clara, CA, USA, IEEE CS Press, pp. 504-511. doi:10.1109/ICSC.2008.82

Keeney, J. and Cahill, V., Chisel: A policy-driven, context-aware, dynamic adaptation framework, in Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks, (Lake Como, Italy), p. 3, June 2003.

Kephart, J. O. and Chess, D. M. (2003) The vision of autonomic computing, IEEE Computer, Vol. 36, No. 1, pp. 41-50

Kiczales, G., des Rivi`eres, J. and Bobrow, D.G., The Art of Metaobject Protocols, MIT Press, 1991.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira, C., Lopes, J., Loingtier, M. and Irwin, J., Aspect-oriented programming, in Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag LNCS 1241, June 1997.

Lappalainen, M. (2009) Kotimaisen puunhankinnan tulevaisuuden liiketoimintamallit –tutkimushanke. Loppuraportti., University of Jyväskylä, School of Business and Economics., Working paper No. 355/2009.

March, S. and Smith, G., Design and Natural Science Research on Information Technology, Decision Support Systems, 15 (1995), 251–266.

Marin, C., Lalanda, P., Docosoc - domain configurable service-oriented computing, In: Proceedings of 5th IEEE International Conference on Services (SCC'07), July 2007, pp. 52–59

Martin, D., Paolucci, M., and Wagner, M., Towards Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective. In OWL-S Experiences and Future Developments Workshop at ESWC 2007, June 2007, Innsbruck, Austria

McKinley, P. K., Sadjadi, S.M., Kasten, E. P., and Cheng, B. H. C., Composing adaptive software, IEEE Computer, vol. 37, no. 7, pp. 56-64, 2004.

McKinley, P., Sadjadi, S., Kasten, E., Cheng, B., A Taxonomy of Compositional Adaptation, Technical Report, MSU-CSE-04-17, Department of Computer Science and Engineering, Michigan State University, East Lansing, Michigan, 2004.

Naumenko, A., Nikitin, S., Terziyan, V., Zharko, A., Strategic Industrial Alliances in Paper Industry: XML- vs. Ontology-Based Integration Platforms, In: The Learning Organization, Special Issue on: Semantic and

Social Aspects of Learning in Organizations, Emerald Publishers, ISSN: 0969-6474, 2005, Vol. 12, No. 5, pp. 492-514.

Nwana, H. S., Software Agents: An Overview, Knowledge Engineering Review, 11(3), 1996.

Odell, J. ed., Agent Technology, OMG, green paper produced by the OMG Agent Working Group, 2000

Oreizy, P., Gorlick, M., Taylor, R.N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D.S., Wolf, A.L., An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, v.14 n.3, p.54-62, May 1999, doi:10.1109/5254.769885

OWL-S: Semantic Markup for Web Services, W3C Member Submission, 22 November 2004, http://www.w3.org/Submission/OWL-S/

Rector, A., Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL, Proc. K-CAP (Knowledge Capture), 2003

SAWSDL – Semantic Annotations for WSDL and XML Schema, W3C Recommendation, 28 August 2007, http://www.w3.org/TR/sawsdl/

SCOMA – Scientific Computing and Optimization in Multidisciplinary Applications , Tekes project, 2005-2009, http://www.mit.jyu.fi/scoma/

Shadbolt, N., Hall, W. and Berners-Lee, T., The Semantic Web revisited, IEEE Intelligent Systems, pp. 96–101, May-June 2006.

SWSF: Semantic Web Services Framework Overview, W3C Member Submission 9 September 2005, http://www.w3.org/Submission/SWSF/

Terziyan, V., Semantic Web Services for Smart Devices in a Global Understanding Environment, In: R. Meersman and Z. Tari (eds.), On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, Lecture Notes in Computer Science, Vol. 2889, Springer-Verlag, 2003, pp.279-291.

Terziyan, V., Semantic Web Services for Smart Devices Based on Mobile Agents, In: International Journal of Intelligent Information Technologies, Vol. 1, No. 2, Idea Group, pp. 43-55, 2005.

Terziyan, V. (Ed.), SmartResource Project Final Report, Technical Report (final report), SmartResource Tekes Project, Agora Center, University of Jyvaskyla, 2007.

Terziyan, V., Nikitin, S., Nagy, M., Khriyenko, O., Kesämiemi, J., Cochez, M., Pulkkis, A., UBIWARE Platform Prototype v. 3.0, Technical Report (Deliverable D3.3), UBIWARE Tekes Project, Agora Center, University of Jyvaskyla, August 2010, 45 pp.

Veijalainen, J., Nikitin, S., Tormala, V., Ontology-based Semantic Web Service platform in Mobile Environments, pp. 83, 7th International Conference on Mobile Data Management (MDM'06), 2006
DOI: http://doi.ieeecomputersociety.org/10.1109/MDM.2006.119

Verma, K. and Sheth, A., Semantically Annotating a Web Service, IEEE Internet Computing 11, 2 (Mar. 2007), 83-85.
DOI=http://dx.doi.org/10.1109/MIC.2007.48

Vesterinen, M., Kotimaisen puunhankinnan tulevaisuuden liiketoimintamallit. In edition Niemelä, T. et al. Puheenvuoroja yrittäjyydestä maaseudulla., University of Jyväskylä, School of Business and Economics, Publications No: 152/2005, pp. 84-100, 2005.

Väätäinen, K., Lappalainen, M., Asikainen, A. and Anttila, P., 2008, Kohti kustannustehokkaampaa puunkorjuuta – puunkorjuuyrittäjän uusien toimintamallien simulointi., Finnish Forest Research Institute. Working Papers No 73.

Walker, R. J., Baniassad, E. L. A. and Murphy, G. C., An initial assessment of aspect-oriented programming, in International Conference on Software Engineering, pp. 120–130, 1999.

Weiser, M., Ubiquitous computing, IEEE Computer, vol. 26, pp. 71–72, October 1993.

Wooldridge, M., Jennings, N. R., 1995, Intelligent agents: theory and practice. The Knowledge Engineering Review, 10, pp 115-152
doi:10.1017/S0269888900008122

Wooldridge, M., Jennings, N. R. and Kinny, D., 2000, The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3, 3 (Sep. 2000), 285-312.
DOI=http://dx.doi.org/10.1023/A:1010071910869

WSDL - Web Services Description Language, Version 2.0 Part 0: Primer, W3C Recommendation, 26 June 2007, http://www.w3.org/TR/wsdl20-primer

WSDL-S - Web Service Semantics, W3C Member Submission, 7 November 2005, http://www.w3.org/Submission/WSDL-S/

WSMO - Web Service Modeling Ontology Primer, W3C Member Submission 3 June 2005, http://www.w3.org/Submission/WSMO-primer/

IV

ONTONUTS: REUSABLE SEMANTIC COMPONENTS FOR MULTI-AGENT SYSTEMS

By

Sergiy Nikitin, Artem Katasonov and Vagan Terziyan 2009
R. Calinescu et al. (Eds.),
Proceedings of the Fifth International Conference
on Autonomic and Autonomous Systems (ICAS 2009), April 21-25, 2009,
Valencia, Spain, IEEE CS Press, pp. 200-207

2009 Fifth International Conference on Autonomic and Autonomous Systems

Ontonuts: Reusable semantic components for multi-agent systems

Sergiy Nikitin, Artem Katasonov and Vagan Terziyan
Industrial Ontologies Group, University of Jyväskylä
{sergiy.nikitin , artem.katasonov , vagan.terziyan}@jyu.fi

Abstract

The volumes of data in information systems are growing drastically. The systems become increasingly complex in trying to handle heterogeneity of ubiquitous components, standards, data formats, etc. According to the vision of Autonomic Computing, the complexity can be handled by introducing self-manageable components able to "run themselves." Agent Technology fits this vision, whereas interoperability among autonomic components can be tackled by Semantic Technologies. The problem of efficient heterogeneous data sharing, exchange and reuse within such systems plays a key role. We present an approach of constructing semantic capabilities (self-descriptive functional components) for software agents and a mechanism for distributed data management that applies these capabilities to build various industrial business intelligence systems.

1. Introduction

The volumes of data in information systems are growing drastically. The systems become increasingly complex in trying to handle heterogeneity of ubiquitous components, standards, data formats, etc. According to the vision of Autonomic Computing [1], the complexity of information systems can be handled by introducing self-manageable components, able to "run themselves." In our opinion, Agent Technology fits this vision very well. Whereas the interoperability among autonomic components (agents) can be tackled by Semantic Technologies, efficient data sharing, exchange and reuse within such systems still play key roles. Semantic agent-driven systems cannot fully substitute e.g. high-performance industrial data storage, nor can they avoid physical distribution of data and services. In attempt to resolve the challenges stated, we are developing an agent platform of a new generation – called UBIWARE [2], [3]. Efficient data sharing, exchange and reuse determine the usability of the UBIWARE platform and its technological success in industry. In this paper we introduce a mechanism for distributed data management within the UBIWARE platform that allows platform users to build distributed industrial business solutions.

The paper is organized in the following way. In the next section, we present an industrial scenario for distributed querying and discuss problems that call for new ICT solutions. In Section 3, we briefly describe the UBIWARE platform. The fourth section presents a concept of semantic components called Ontonuts and shows how they are applied to the scenario of distributed querying. The discussion on related work is presented in Section 5. We conclude and propose future work in Section 6.

2. Distributed Querying Scenario in Paper Industry

The scenario we have selected is based on the real software infrastructure from process industry. There is a complex production line (e.g. paper producing machine) which is served by a number of control and diagnostic systems. The measurements taken from sensors are stored in the Alarm History Database. The Diary Database contains records about critical alarms and comments from maintenance workers. There are also comments on actions taken. The Scheduled Performance Monitoring database stores results of the analysis that is performed daily. The analysis includes all nodes with performance indices that indicate the condition of the node (Figure 1).

As an example, suppose a serious fault happened in the paper machine that has led to the subsequent alarm and maintenance actions. All events are recorded in the respective databases. However, in order to analyze the preceding events, e.g. during the week before the fault, an expert may need to query portions of data from all the databases, then change filtering parameters and query databases again and again. An expert may have interfaces to all the databases separately; however, the expert's decision will be stored (if it would be at all) to a separate file or database.

978-0-7695-3584-5/09 $25.00 © 2009 IEEE
DOI 10.1109/ICAS.2009.34

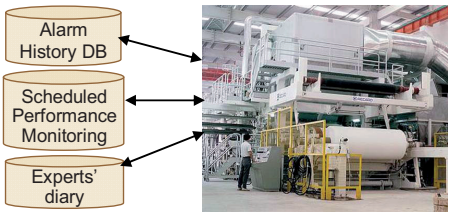

Figure 1. The IT-infrastructure of the paper machine

Within the current IT infrastructure, it is hard to find previously made expert decisions on an immediate fault situation. Thus there is no integrated view on all the contents of the databases, nor on other sources of information about the paper machine operation. In the Semantic Web domain, it is called a *proof* when any knowledge is connected with the rules and facts that were used to infer it. The problem of an integrated view on the information is important in the industrial automation particularly because we perceive that many experienced experts in a majority of companies are going to retire during next 5-10 years without a proper knowledge transfer to new experts. On the other hand, semantic linking of the information will be in a great demand when the standardization efforts taken in companies will go beyond the companies' boundaries and will call for a unified mechanism of distributed querying for knowledge and expertise exchange (see Figure 2).

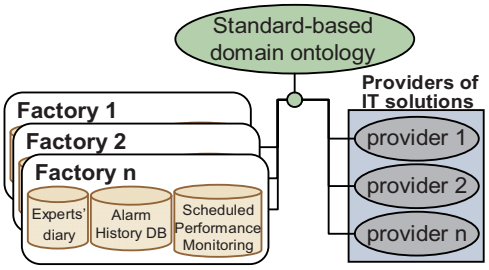

Figure 2. Inter-company standardization in paper industry

In the ideal case, companies would be able to sell information and analytic services to each other seamlessly with little or no programming effort. The services sold would be easily integrated into the company environment with the guaranteed compatibility. Furthermore, service consumers (factories) could become service providers too. Industry might share data and use it as learning samples for analytic services. The role of providers of IT solutions would shift from the integration aspects to those of intelligence. We, therefore, foresee the need for tools and capabilities in the UBIWARE platform that will simplify distributed querying and information integration.

3. UBIWARE platform

In this section we briefly introduce the UBIWARE agent-driven middleware platform, its agent engine, and S-APL – a Semantic Agent Programming Language for programming of software agents within the platform.

3.1. UBIWARE Platform Architecture

Central to the core platform is the architecture of a UBIWARE agent, depicted in Figure 3.

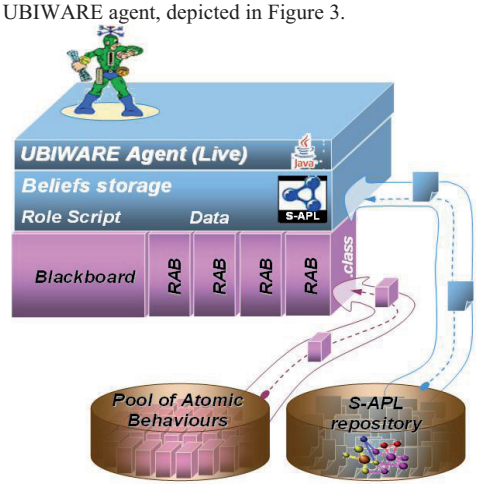

Figure 3. UBIWARE Agent

There is a *Live* behavior engine implemented in Java, a declarative middle layer, and a set of Java components, known as *Reusable Atomic Behaviors* (RABs). RABs can be considered sensors and actuators, i.e. components sensing or affecting the agent's environment, but are not restricted these. A RAB also can be a reasoner (data processor) if some of the logic needed is not efficient or possible to realize with the S-APL means, or if one wants to enable an agent to do some other kind of reasoning beyond the rule-based one.

The UBIWARE agent architecture implies that a particular UBIWARE-based software application will consist of a set of S-APL documents (data and

behavior models) and a set of specific atomic behaviors needed for this particular application. Since reusability is an important UBIWARE concern, it is reasonable that the UBIWARE platform provides some of those ready-made. Therefore, the UBIWARE platform, as such, can be seen as consisting of the following three elements:

- The Live behavior engine
- A set of "standard" S-APL models
- A set of "standard" RABs

The extensions to the platform are exactly some sets of such "standard" S-APL models and RABs that can be used by the developers to embed into their applications certain UBIWARE features.

3.2. S-APL platform language

In the UBIWARE Platform, behavior models are presented in a high-level, rule-based language, the *Semantic Agent Programming Language* (*S-APL)*. S-APL is based on the RDF (http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/) data model, i.e. the whole document can be seen as a set of subject-predicate-object triples. A behavior model specifies the initial beliefs (including knowledge, goals, commitments, and behavioral rules) of the agent in the role. Commitments and behavioral rules normally lead to adding/removing beliefs and executing various RABs. The notation that is selected for use in S-APL is a subset of Notation3 (http://www.w3.org/DesignIssues/Notation3.html). Notation3 was proposed by Tim Berners-Lee as an alternative to the dominant RDF/XML notation. There are namespaces in S-APL; in particular, the "sapl" namespace is used for the resources that are defined in the language's ontology. The default namespace is used for all the other resources in this paper.

In S-APL every statement is a belief of the agent. Simple belief would look like:

```
:John :Loves :Mary
```

Whereas a belief in a context is defined as:

```
{:John :Loves :Mary}
:since {:Year :Is 2005}
```

The unconditional commitment to an action (e.g. calling an RAB) is defined as follows:

```
{sapl:I sapl:do java:ubiware.shared.
MessageSenderBehavior}
sapl:configuredAs {
   p:receiver sapl:is :John.
   p:content sapl:is "bla bla".
   sapl:Success sapl:add {
     :John :was :notified }}
```

When the agent's engine finds a belief with the "java:" prefix in a general context G (active memory), it executes the specified action (RAB).

The sequential plan can be defined as:

```
{ sapl:I sapl:do ...}
sapl:configuredAs{ ...
  sapl:Success sapl:add {
    { sapl:I sapl:do ...}
     sapl:configuredAs {...} } }
```

meaning that, upon successful execution of the first commitment, the enclosed one should be added.

However, the central construct of the language is the conditional commitment:

```
{:John :Loves :Mary} =>
{{sapl:I sapl:do java:SendMail}
sapl:configuredAs {...}}
```

The interpretation is straightforward: Upon occurrence of a belief that satisfies the condition stated in the subject, the contents of the object are added to agent's general context G. Another key construct is matching with variables (querying). The commitment for querying is defined as follows:

```
{{:John :Loves ?x} :accordingTo ?y.
 ?x sapl:is :Girl
} =>
{sapl:I sapl:do java:SendMail}
sapl:configuredAs {
p:receiver sapl:is ?x ...}
```

which can be interpreted, then, as "If John loves ?x, according to someone's opinion, and ?x is a girl, then send an email to ?x".

Yet one more construct is a behavior rule:

```
{{...} => {...}} sapl:is sapl:Rule
```

The behavior rule differs from the commitment. Whereas a commitment is removed from the agent's beliefs upon execution, the rule stays and executes in agent's beliefs permanently. It must be removed explicitly.

In this section we have briefly introduced the core concepts of the UBIWARE. In the next section, we present an extension done beyond the core that makes an important step towards practical applicability of the platform in industrial applications.

4. Ontonuts Concept

We introduce here the concept of *Ontonut* to facilitate the presentation of modular scripts and plans within the UBIWARE platform. Ontonut is a semantic software component. Instances of the Ontonut concept generally represent a capability with known input and expected output. We then extend Ontonuts to solve the problem of distributed querying discussed in Section 2 of this paper.

4.1. Ontonuts in a nutshell

The Ontonuts technology is implemented as a combination of an S-APL script and RABs and, hence, can be dynamically added, removed or configured. Ontonuts allow componentization of S-APL code by introducing a semantic annotation to it. Such annotated

pieces of code are called *capabilities* (analog of function in procedural programming). The capabilities have S-APL descriptions with explicitly defined preconditions and effects:

```
Ontonut: {script, precondition, effect}
```

The capabilities can be dynamically combined further into plans and put into execution by the Ontonuts engine, which allows us to automatically compose the agent's actions to achieve a specified goal. The script part of the capability in general has an S-APL code that produces the effect once the precondition is satisfied. The whole data model of the UBIWARE platform is triple-based; therefore goals, preconditions and effects are defined as triple sets in S-APL. For example, we have an initial data set {*A A A*}, a goal *G1* defined as {*C C C*}, and we have two ontonuts *O1* and *O2*, defined as:

```
O1 rdf:type :Ontonut
O1 ont:precondition {A A A}
O1 ont:effect {B B B}
O1 ont:script {{A A A}=>... =>{B B B}}
O2 rdf:type :Ontonut
O2 ont:precondition {B B B}
O2 ont:effect {C C C}
O2 ont:script {{B B B}=>...=>{C C C}}
```

The appearance of the goal G1 will activate the Ontonuts engine that will match the G1 against available effects and then apply planning, which will result in an execution plan: *O1=>O2=>G1*.

Ontonuts reuse available scripts and RABs without any modifications to the platform. Architecturally, the Ontonuts engine consists of three main components: the *Triggering Rule*, *Action Planner* and *Plan Executor* (Figure 4). The Ontonuts Triggering Rule is a starting point of the engine work. The Triggering Rule is a MetaRule, i.e. it runs before other rules and commitments. On each iteration of the Live behavior, the rule checks whether there are any Ontonut calls to be handled and passes the activity to the Action Planner, which provides an execution plan for the Plan Executor.

Ontonut capabilities may include interaction with other agents or external resources, such as databases, files or web services. On the other hand, capabilities can perform local actions and do some computations on the data, e.g. statistical analysis.

4.2. Invoking Ontonuts

The Ontonuts engine supports three types of Ontonut calls:

- Explicit
- Goal-based
- Pattern-based

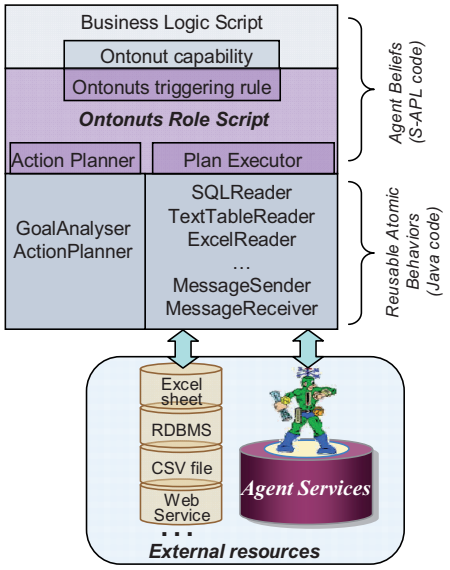

Figure 4. Architecture of Ontonuts

The *Explicit call* to Ontonut is defined as:

```
{sapl:I sapl:do <ontonutid>}
  sapl:configuredAs
{p:precondition sapl:is {<Input
statements>}}.
```

The result of the call is added to the G.

The *Goal-based call* is initiated by adding the following goal definition to the G:

```
sapl:I ont:haveGoal :id.
:id ont:goalDef{<goal statements>}
:id ont:initData {<initial data>}
```

The Ontonuts engine runs the planner to check if the plan can be produced for the goal using the initial data provided.

The third type – a *Pattern-based call* is triggered when the content of the active commitment in its left part matches the effect pattern of at least one Ontonut:

```
{A A ?a} => {<some action with ?a>}.
```

This call can be considered an abbreviated syntax for goal definition when the left part of the commitment is considered a goal. The Ontonuts engine intercepts such a commitment before it executes, removes it from G, and then uses the left part information to perform planning and execution. However, this type of call does not specify the initial data set as the second type of call does. Such goal definition is possible for those Ontonuts, for which precondition is always true within the goal specified. For example, an Ontonut can perform queries over a certain database and use a (sub)pattern of the goal to produce a query and execute it. After the goal is

achieved, and, hence, the variable values in the left part of the commitment can be assigned, the Ontonuts engine produces the result (the right part) of the commitment using the variable values. This type of Ontonuts targets mainly the task of distributed querying that is discussed in Section 4.6.

4.3. Planning the execution

The planning is organized as a goal-driven process. We apply a backward chaining algorithm to build an action plan, which may involve other Ontonuts and Rules. The planner performs a semantic inference over the set of initial data before the actual plan generation starts. Therefore, the semantic annotations of Ontonuts, as well as the corresponding domain ontology, are key success factors of the Ontonuts-based applications. The planner acts in a straightforward way – it matches the goal against Ontonut annotations by subtracting (operation over sets) these annotations from the goal. If a goal can be fulfilled by the available initial data and Ontonuts, the planner starts to check whether the preconditions of these Ontonuts can be fulfilled. If the preconditions may need to use other Ontonuts, they are checked as well. In such an iterative manner, the planner builds a solution tree. The planner then chooses the preferable solution using different criteria, e.g. utility-based selection.

4.4. Handling the execution

The Ontonuts engine does not execute the plan as a whole; rather, it generates a plan that is run by the agent's Live behavior engine. However, the plan is not straightforward: It includes additional handlers that allow the Ontonuts engine to observe the state of the execution and react if the execution cannot be successfully completed. The plan is sequential and therefore has steps or control points. At each control point, the plan produces the statements that represent the status of the execution. These statements are collected into a container that is attached to the plan:

```
:planid ont:execStatus {
     :01 ont:status ont:Success.
     …
     :nn ont:status ont:NoResponse.}.
```

The engine then can use the status information for re-planning if the current plan did not succeed.

There are two classes of Ontonuts executed in different ways:

- Self-running
- Engine-running

The latter type has a built-in script that runs in the agent's Live behavior as an independent code and returns the result to the G container. Meanwhile, the former is a description that is recognized and executed by the Ontonuts engine. The engine-running Ontonut calls are presented in the plan as explicit (see Section 4.2). In the current version, the engine supports one type of engine-running Ontonuts that simplify access to the databases.

4.5. Distributed querying with Ontonuts

There are two main viewpoints towards distributed querying in the UBIWARE: adapter-based and service-based. The former tackles the adaptation of the data sources that are external to the platform (databases, files, web services, etc.), while the latter deals with the agent-to-agent querying. Nevertheless, both target the same goal: to make distributed querying logic transparent (simple) to the UBIWARE agent (see Figure 5).

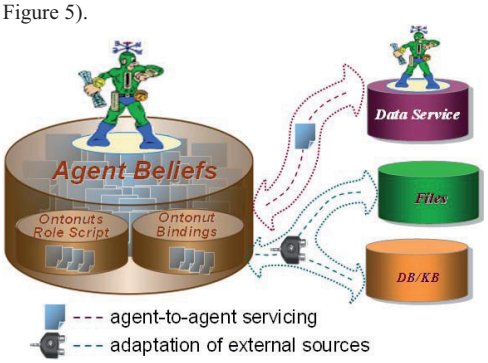

Figure 5. Distributed querying in UBIWARE

The agent-to-agent querying follows the servicing paradigm and, in particular, the data servicing discussed in [4]. The adaptation of external sources (e.g. RDF-based adaptation is discussed in [5]) resolves the problem of connectivity and interaction with those resources that are external to the platform, i.e. communicating with them in their native language.

However, from the business logic developer's point of view, any remote resource should be transparent in order to keep business logic script as modular and clear as possible. Ontonuts become the wrapper, adapter and connector in one place.

In a distributed querying task, every Ontonut is an interface to the data source that has an associated data query pattern (effect) it replies to. The Ontonuts engine introduces an extension for the data source-based Ontonuts. The extension allows for the Ontonut developer to not implement all the RAB calls and S-APL transformations from the scratch, but rather to define a description of the data source and

transformation mappings. We call this subclass of Engine-running Ontonuts *Donuts* (Database Ontonuts). The engine distinguishes the Donuts and treats them in a different way. The user-defined query can match several Donuts; therefore, the Triggering Rule invokes Action Planner. The Action Planner distinguishes sub-queries from the initial query and produces a distributed query plan. The plan is then passed to the Plan Executor. The executor handles the intermediate results of sub-queries and modifies subsequent sub-queries accordingly.

The structure of the Donuts is defined by Donuts Ontology (see Figure 6).

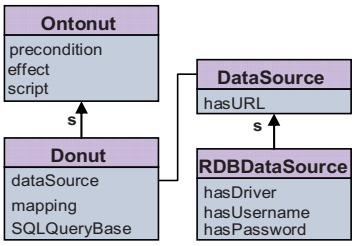

Figure 6. A fragment of Donuts ontology

The fragment of the ontology above describes the root classes *Ontonut* and *DataSource,* as well as their extensions for connectivity with relational databases (*Donut, RDBDataSource*). Similarly, other types of extensions will include type-specific facets in their descriptions.

The Plan Executor uses data source descriptions for fetching the sub-queries and applies mapping definitions to transform sub-query results into the semantic form.

4.6. An illustrative example

The example presented here is based on the usage scenario described in Section 2 of this paper. Suppose that a fault situation happened and the agent of the expert wants to extract the comment strings from the Expert's Diary database and align them with the performance indices from the Performance Monitoring database. Then the alarm limits and alarm values are extracted from the Alarm History database, based on the node-to-tag mappings. The time interval used for filtering is 10 days before the fault. The agent prints the collected values to the command line. The resources involved in the query and their tables are shown in Figure 7. Each resource has an associated Ontonut in the agent's beliefs.

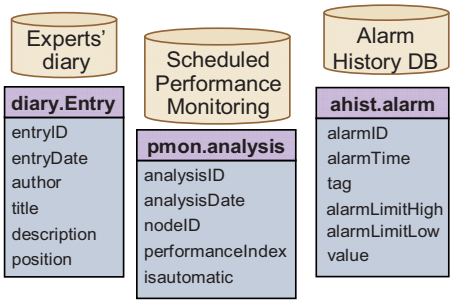

Figure 7. Sample database tables

The description of the Ontonut associated with the expert's diary and the datasource object (an instance of RDBDataSource) are shown below:

```
:DiaryEntryNut rdf:type ont:Donut.
:DiaryEntryNut ont:dataSource :entrydb.

:DiaryEntryNut ont:SQLQueryBase
"SELECT entryID, entryDate, author, title,
description, position FROM diary.Entry".

:DiaryEntryNut ont:mapping {
     ?entryId ont:mapsTo
     {ont:sqlentity sapl:is "entryID"}.
     ?entryDate ont:mapsTo
     {ont:sqlentity sapl:is "entryDate"}.
     ?author ont:mapsTo
     { ont:sqlentity sapl:is "author"}.
     ?title ont:mapsTo
     { ont:sqlentity sapl:is "title"}.
     ?description ont:mapsTo
     {ont:sqlentity sapl:is "description"}.
     ?position ont:mapsTo
     {ont:sqlentity sapl:is "position"}
}.

:DiaryEntryNut ont:effect {
     ?entry :entryId ?entryId.
     ?entry :entryDate ?entrydate.
     ?entry :author ?author.
     ?entry :title ?title.
     ?entry :description ?description.
     ?entry :position ?position
}.

:entrydb rdf:type ont:RDBDataSource.
:entrydb ont:hasURL http://host:80/diary.
:entrydb ont:hasDriver
            oracle.jdbc.OracleDriver.
:entrydb ont:hasUsername diaryuser.
:entrydb ont:hasPassword mypwd.
```

The `ont:SQLQueryBase` defines the SQL query that extracts the data that is used to produce the Ontonut instances. The mapping definitions use the column names from the SQLQueryBase property.

Other Ontonut descriptions are defined in a similar manner. The effect of the second Ontonut used in this example is defined as follows:

```
:PMAnalysisNut ont:effect {
```

```
    ?pmnode :analysisID ?aid.
    ?pmnode :analysisDate ?adate.
    ?pmnode :nodeID ?nodeid.
    ?pmnode :performanceIndex ?pindex
    ?pmnode :isautomatic ?isautom}.
```

The effect of the Ontonut for the Alarm History database is defined as:

```
:AHAlarmNut ont:effect {
    ?alarm :alarmID ?alid.
    ?alarm :alarmTime ?aldate.
    ?alarm :tag ?altag.
    ?alarm :alimitHigh ?ahigh
    ?alarm :alimitLow ?alow
    ?alarm :value ?avalue}.
```

The commitment (query) in the agent's beliefs is shown below:

```
{?entry  :entryDate ?edate.
 ?entry  :title ?ctitle.
 ?entry  :position ?pos.

 ?pos    :mapsTo_1 ?nodeid.

 ?pmnode :nodeId ?nodeid.
 ?pmnode :performanceIndex ?pindex.
 ?pmnode :analysisDate ?adate.

 ?pos    :mapsTo_2 ?tag.

 ?alarm  :tag ?tag.
 ?alarm  :alarmTime ?atime.
 ?alarm  :alarmValue ?value.
 ?alarm  :almLimitHigh ?ahigh.
 ?alarm  :almLimitLow ?alow.

 ?edate = "31.12.2008".
 ?adate < 31.12.2008.
 ?adate > 21.12.2008.
 ?atime < 23:59:59T31.12.2008.
 ?atime > 23:59:59T21.12.2008.
}=>
{ {gb:I gb:do :Print} gb:configuredAs
  {x:print gb:is "| ?ctitle | ?edate |
   ?pos | ?adate | ?pindex | ?atime |
   ?value | ?ahigh | ?alow | "}. }
```

The commitment does not explicitly refer to the type of the Ontonut by the *rdf:type* property, which would simplify the implementation of the triggering procedure, but we apply pattern-based matching, i.e. use subtract operation over available Ontonut effect patterns and the query.

In the particular query, the triple that matches the identifiers defined as:

```
?pos :mapsTo_1 ?nodeid.
```

It is a bridging property between the two property values of respective Ontonuts, which have physical data sources behind. It may belong to any of the Ontonuts it bridges or be an independent Ontonut: How it is modeled is domain-specific.

As soon as the matching with available Ontonuts has succeeded, the matchmaking rule passes the control to the Query Planner. In this particular case, the work of Query Planner is straightforward – to decide which Ontonut is to be queried first and apply the query parameters for the SQL query generation. The order of execution may depend e.g. on the average expected number of records of each independent sub-query. The methods of execution planning and optimization go beyond the scope of this paper. Further reading suggestions are in the next Section.

The result of the first executed sub-query is then used to limit the range of the variables in the subsequent sub-query. When all the query results are collected, they are printed to the command line (see Figure 8).

cti-tle	e-date	pos	adate	pin dex	atime	va-lue	hi gh	lo w
-	-	-	21.12 .2008	0.8	14:37 21.12.2008	19.7	50	20
-	-	-	-	-	16:23 23.12.2008	19.5	50	20
-	-	-	24.12 .2008	0.7	06:01 24.12.2008	18.0	50	20
-	-	-	-	-	16:30 25.12.2008	19.3	50	20
-	-	-	27.12 .2008	0.6	14:37 27.12.2008	17.7	50	20
-	-	-	-	-	18:17 28.12.2008	18.1	50	20
-	-	-	30.12 .2008	0.6	04:03 30.12.2008	15.4	50	20
-	-	-	-	-	14:37 30.12.2008	17.9	50	20
Paper jam	31.12. 2008	QT-123	-	-	12:59 31.12.2008	10.0	50	20

Figure 8. Query results

The table of results is compressed (duplicate rows are removed and repeating values substituted with dash sign) and arranged in a chronological order for a purpose of readability. The table allows e.g. an expert to analyze how the actual parameter values and performance indices were behaving before the fault happened.

5. Related work

The notion of semantic component composition is discussed in [10] and [11], but the authors focus on improving the programming paradigm as such, not the autonomic computing and semantic agent programming. The execution planning and optimization of queries are thoroughly researched in [6] and are a rather complementary part for our work dealing with query planning. The approach in [7] proposes a solution for management of corporate histories using multi-agent system and semantic data,

our work, in contrast, introduces an intra-agent feature, that simplifies the programming of an agent. A set of Semantic Web Service platforms and languages like OWL-S [8] and WSMF [9] externalize semantic components and allow planning and execution. However, the Ontonuts approach treats components as internal capabilities of an agent that can be externalized as semantic services. While the approach presented in [12] proposes an extension to the RDF language in order to allow modifications of the RDF content upon some triggered actions, we deal with the extension to the rule-based agent programming language, which allows componentization and data updates. Ontonuts allow semantic integration of both internal and external capabilities and stand for a semantic agent-driven workflow planning and execution engine.

6. Conclusions and future work

The approach presented in this work aims toward a quite specific target – to structure the S-APL code of the UBIWARE agent in order to simplify programming of the agent and allow automated goal-driven planning. Although the paper mainly covers a quite narrow domain of distributed querying, it involves the generic problems of agent planning and semantic programming. The emphasis of this paper is on the automated planning of distributed queries and is related to the notion of distributed RDF queries and so-called virtual graphs, when the graph being queried does not have RDF content beneath. The approach proposed uses patterns to define desired result and applies queries or any other code to fill the patterns requested.

At the moment, the first prototype is nearing its completion. We plan to extend functionality of the Ontonuts engine by introducing more engine-running Ontonut types, e.g. for web services and agent services. Another important step to be taken is to perform scalability tests over large data volumes and compare the results with purely SQL-based implementation alternatives. The planning procedure should also be improved to keep alternative pathways on each stage of the execution. Furthermore, in theory, agents can share Ontonuts as self-containing executable modules.

7. Acknowledgement

This research has been performed in the UBIWARE project, funded by TEKES, and the industrial consortium of Metso Automation, Fingrid, Nokia and Inno-W.

8. References

[1] Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *IEEE Computer*, 36(1):41–50.

[2] Katasonov A., Terziyan, V., Semantic Agent Programming Language (S-APL): A Middleware Platform for the Semantic Web, In: Proceedings of the Second IEEE International Conference on Semantic Computing (ICSC-2008) / International Workshop on Middleware for the Semantic Web, August 4-7, 2008, Santa Clara, CA, USA, IEEE CS Press, pp. 504-511.

[3] Katasonov A., Kaykova O., Khriyenko O., Nikitin S., Terziyan, V., Smart Semantic Middleware for the Internet of Things, In: Proceedings of the 5-th International Conference on Informatics in Control, Automation and Robotics, 11-15 May, 2008, Funchal, Madeira, Portugal, ISBN: 978-989-8111-30-2, Volume ICSO, pp. 169-178.

[4] Quilitz, B.; Leser, U., "Querying Distributed RDF Data Sources with SPARQL", *The Semantic Web: Research and Applications, 5th European Semantic Web Conference*, *ESWC 2008*, Tenerife, Canary Islands, Spain, June 1-5, 2008, pp.524-538.

[5] Langegger, A.; Blochl, M.; Woss, W., "Sharing Data on the Grid using Ontologies and distributed SPARQL Queries", *18th International Conference on Database and Expert Systems Applications, DEXA '07*. Regensburg, Germany, 3-7 Sept., 2007, pp.450-454.

[6] Obermeier, P., Nixon, L., *A Cost Model for Querying Distributed RDF Repositories*, Advanced Reasoning on the Web workshop, European Semantic Web Conference (ESWC) 2008, Tenerife, Spain.

[7] F. Gandon, L. Berthelot, R. Dieng-Kuntz., A Multi-Agent Platform for a Corporate Semantic Web, in: Proceedings of AAMAS'2002 (First International Joint Conference on Autonomous Agents and Multi-Agent Systems), Bologna, Italy, July 15-19 2002, p. 1025-1032.

[8] D. Martin et al., "Bringing Semantics to Web Services: The OWL-S Approach." *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)* 6-9, 2004, San Diego, California, USA.

[9] D. Fensel and C. Bussler., The web service modeling framework (WSMF), Electronic Commerce: Research and Applications, (1):113–137, 2002.

[10] Sjachyn, M. and Beus-Dukic, L. 2006. Semantic Component Selection — SemaCS. In *Proceedings of the Fifth international Conference on Commercial-off-the-Shelf (Cots)-Based Software Systems* (February 13 - 16, 2006). ICCBSS. IEEE Computer Society, Washington, DC, 83.

[11] Liu, X., Wang, B., and Kerridge, J. 2005. Achieving seamless component composition through scenario-based deep adaptation and generation. *Sci. Comput. Program.* 56, 1-2 (Apr. 2005), 157-170.

[12] G. Papamarkos, A. Poulovassilis, and P. T.Wood. RDFTL: An Event-Condition-Action Language for RDF. In *Proc. 3rd Int.Workshop on Web Dynamics (in conjunction with WWW2004)*, 2004.

JYVÄSKYLÄ STUDIES IN COMPUTING

1 Ropponen, Janne, Software risk management - foundations, principles and empirical findings. 273 p. Yhteenveto 1 p. 1999.
2 Kuzmin, Dmitri, Numerical simulation of reactive bubbly flows. 110 p. Yhteenveto 1 p. 1999.
3 Karsten, Helena, Weaving tapestry: collaborative information technology and organisational change. 266 p. Yhteenveto 3 p. 2000.
4 Koskinen, Jussi, Automated transient hypertext support for software maintenance. 98 p. (250 p.) Yhteenveto 1 p. 2000.
5 Ristaniemi, Tapani, Synchronization and blind signal processing in CDMA systems. - Synkronointi ja sokea signaalinkäsittely CDMA järjestelmässä. 112 p. Yhteenveto 1 p. 2000.
6 Laitinen, Mika, Mathematical modelling of conductive-radiative heat transfer. 20 p. (108 p.) Yhteenveto 1 p. 2000.
7 Koskinen, Minna, Process metamodelling. Conceptual foundations and application. 213 p. Yhteenveto 1 p. 2000.
8 Smolianski, Anton, Numerical modeling of two-fluid interfacial flows. 109 p. Yhteenveto 1 p. 2001.
9 Nahar, Nazmun, Information technology supported technology transfer process. A multi-site case study of high-tech enterprises. 377 p. Yhteenveto 3 p. 2001.
10 Fomin, Vladislav V., The process of standard making. The case of cellular mobile telephony. - Standardin kehittämisen prosessi. Tapaustutkimus solukkoverkkoon perustuvasta matkapuhelintekniikasta. 107 p. (208 p.) Yhteenveto 1 p. 2001.
11 Päivärinta, Tero, A genre-based approach to developing electronic document management in the organization. 190 p. Yhteenveto 1 p. 2001.
12 Häkkinen, Erkki, Design, implementation and evaluation of neural data analysis environment. 229 p. Yhteenveto 1 p. 2001.
13 Hirvonen, Kullervo, Towards better employment using adaptive control of labour costs of an enterprise. 118 p. Yhteenveto 4 p. 2001.
14 Majava, Kirsi, Optimization-based techniques for image restoration. 27 p. (142 p.) Yhteenveto 1 p. 2001.
15 Saarinen, Kari, Near infra-red measurement based control system for thermo-mechanical refiners. 84 p. (186 p.) Yhteenveto 1 p. 2001.
16 Forsell, Marko, Improving component reuse in software development. 169 p. Yhteenveto 1 p. 2002.
17 Virtanen, Pauli, Neuro-fuzzy expert systems in financial and control engineering. 245 p. Yhteenveto 1 p. 2002.
18 Kovalainen, Mikko, Computer mediated organizational memory for process control. Moving CSCW research from an idea to a product. 57 p. (146 p.) Yhteenveto 4 p. 2002.
19 Hämäläinen, Timo, Broadband network quality of service and pricing. 140 p. Yhteenveto 1 p. 2002.
20 Martikainen, Janne, Efficient solvers for discretized elliptic vector-valued problems. 25 p. (109 p.) Yhteenveto 1 p. 2002.
21 Mursu, Anja, Information systems development in developing countries. Risk management and sustainability analysis in Nigerian software companies. 296 p. Yhteenveto 3 p. 2002.
22 Seleznyov, Alexandr, An anomaly intrusion detection system based on intelligent user recognition. 186 p. Yhteenveto 3 p. 2002.
23 Lensu, Anssi, Computationally intelligent methods for qualitative data analysis. 57 p. (180 p.) Yhteenveto 1 p. 2002.
24 Ryabov, Vladimir, Handling imperfect temporal relations. 75 p. (145 p.) Yhteenveto 2 p. 2002.
25 Tsymbal, Alexey, Dynamic integration of data mining methods in knowledge discovery systems. 69 p. (170 p.) Yhteenveto 2 p. 2002.
26 Akimov, Vladimir, Domain decomposition methods for the problems with boundary layers. 30 p. (84 p.). Yhteenveto 1 p. 2002.
27 Seyukova-Rivkind, Ludmila, Mathematical and numerical analysis of boundary value problems for fluid flow. 30 p. (126 p.) Yhteenveto 1 p. 2002.
28 Hämäläinen, Seppo, WCDMA Radio network performance. 235 p. Yhteenveto 2 p. 2003.
29 Pekkola, Samuli, Multiple media in group work. Emphasising individual users in distributed and real-time CSCW systems. 210 p. Yhteenveto 2 p. 2003.
30 Markkula, Jouni, Geographic personal data, its privacy protection and prospects in a location-based service environment. 109 p. Yhteenveto 2 p. 2003.
31 Honkaranta, Anne, From genres to content analysis. Experiences from four case organizations. 90 p. (154 p.) Yhteenveto 1 p. 2003.
32 Raitamäki, Jouni, An approach to linguistic pattern recognition using fuzzy systems. 169 p. Yhteenveto 1 p. 2003.
33 Saalasti, Sami, Neural networks for heart rate time series analysis. 192 p. Yhteenveto 5 p. 2003.
34 Niemelä, Marketta, Visual search in graphical interfaces: a user psychological approach. 61 p. (148 p.) Yhteenveto 1 p. 2003.
35 You, Yu, Situation Awareness on the world wide web. 171 p. Yhteenveto 2 p. 2004.
36 Taatila, Vesa, The concept of organizational competence – A foundational analysis. - Perusteanalyysi organisaation kompetenssin käsitteestä. 111 p. Yhteenveto 2 p. 2004.

37 LYYTIKÄINEN, VIRPI, Contextual and structural metadata in enterprise document management. - Konteksti- ja rakennemetatieto organisaation dokumenttien hallinnassa. 73 p. (143 p.) Yhteenveto 1 p. 2004.
38 KAARIO, KIMMO, Resource allocation and load balancing mechanisms for providing quality of service in the Internet. 171 p. Yhteenveto 1 p. 2004.
39 ZHANG, ZHEYING, Model component reuse. Conceptual foundations and application in the metamodeling-based systems analysis and design environment. 76 p. (214 p.) Yhteenveto 1 p. 2004.
40 HAARALA, MARJO, Large-scale nonsmooth optimization variable metric bundle method with limited memory. 107 p. Yhteenveto 1 p. 2004.
41 KALVINE, VIKTOR, Scattering and point spectra for elliptic systems in domains with cylindrical ends. 82 p. 2004.
42 DEMENTIEVA, MARIA, Regularization in multistage cooperative games. 78 p. 2004.
43 MAARANEN, HEIKKI, On heuristic hybrid methods and structured point sets in global continuous optimization. 42 p. (168 p.) Yhteenveto 1 p. 2004.
44 FROLOV, MAXIM, Reliable control over approximation errors by functional type a posteriori estimates. 39 p. (112 p.) 2004.
45 ZHANG, JIAN, Qos- and revenue-aware resource allocation mechanisms in multiclass IP networks. 85 p. (224 p.) 2004.
46 KUJALA, JANNE, On computation in statistical models with a psychophysical application. 40 p. (104 p.) 2004.,
47 SOLBAKOV, VIATCHESLAV, Application of mathematical modeling for water environment problems. 66 p. (118 p.) 2004.
48 HIRVONEN, ARI P., Enterprise architecture planning in practice. The Perspectives of information and communication technology service provider and end-user. 44 p. (135 p.) Yhteenveto 2 p. 2005.
49 VARTIAINEN, TERO, Moral conflicts in a project course in information systems education. 320 p. Yhteenveto 1p. 2005.
50 HUOTARI, JOUNI, Integrating graphical information system models with visualization techniques. - Graafisten tietojärjestelmäkuvausten integrointi visualisointitekniikoilla. 56 p. (157 p.) Yhteenveto 1p. 2005.
51 WALLENIUS, EERO R., Control and management of multi-access wireless networks. 91 p. (192 p.) Yhteenveto 3 p. 2005.
52 LEPPÄNEN, MAURI, An ontological framework and a methodical skeleton for method engineering – A contextual approach. 702 p. Yhteenveto 2 p. 2005.
53 MATYUKEVICH, SERGEY, The nonstationary Maxwell system in domains with edges and conical points. 131 p. Yhteenveto 1 p. 2005.
54 SAYENKO, ALEXANDER, Adaptive scheduling for the QoS supported networks. 120 p. (217 p.) 2005.
55 KURJENNIEMI, JANNE, A study of TD-CDMA and WCDMA radio network enhancements. 144 p. (230 p.) Yhteenveto 1 p. 2005.
56 PECHENIZKIY, MYKOLA, Feature extraction for supervised learning in knowledge discovery systems. 86 p. (174 p.) Yhteenveto 2 p. 2005.
57 IKONEN, SAMULI, Efficient numerical methods for pricing American options. 43 p. (155 p.) Yhteenveto 1 p. 2005.
58 KÄRKKÄINEN, KARI, Shape sensitivity analysis for numerical solution of free boundary problems. 83 p. (119 p.) Yhteenveto 1 p. 2005.
59 HELFENSTEIN, SACHA, Transfer. Review, reconstruction, and resolution. 114 p. (206 p.) Yhteenveto 2 p. 2005.
60 NEVALA, KALEVI, Content-based design engineering thinking. In the search for approach. 64 p. (126 p.) Yhteenveto 1 p. 2005.
61 KATASONOV, ARTEM, Dependability aspects in the development and provision of location-based services. 157 p. Yhteenveto 1 p. 2006.
62 SARKKINEN, JARMO, Design as discourse: Representation, representational practice, and social practice. 86 p. (189 p.) Yhteenveto 1 p. 2006.
63 ÄYRÄMÖ, SAMI, Knowledge mining using robust clustering. 296 p. Yhteenveto 1 p. 2006.
64 IFINEDO, PRINCELY EMILI, Enterprise resource planning systems success assessment: An integrative framework. 133 p. (366 p.) Yhteenveto 3 p. 2006.
65 VIINIKAINEN, ARI, Quality of service and pricingin future multiple service class networks. 61 p. (196 p.) Yhteenveto 1 p. 2006.
66 WU, RUI, Methods for space-time parameter estimation in DS-CDMA arrays. 73 p. (121 p.) 2006.
67 PARKKOLA, HANNA, Designing ICT for mothers. User psychological approach. – Tieto- ja viestintätekniikoiden suunnittelu äideille. Käyttäjäpsykologinen näkökulma. 77 p. (173 p.) Yhteenveto 3 p. 2006.
68 HAKANEN, JUSSI, On potential of interactive multiobjective optimization in chemical process design. 75 p. (160 p.) Yhteenveto 2 p. 2006.
69 PUTTONEN, JANI, Mobility management in wireless networks. 112 p. (215 p.) Yhteenveto 1 p. 2006.
70 LUOSTARINEN, KARI, Resource , management methods for QoS supported networks. 60 p. (131 p.) 2006.
71 TURCHYN, PAVLO, Adaptive meshes in computer graphics and model-based simulation. 27 p. (79 p.) Yhteenveto 1 p.
72 ZHOVTOBRYUKH, DMYTRO, Context-aware web service composition. 290 p. Yhteenveto 2 p. 2006.

73 KOHVAKKO, NATALIYA, Context modeling and utilization in heterogeneous networks. 154 p. Yhteenveto 1 p. 2006.
74 MAZHELIS, OLEKSIY, Masquerader detection in mobile context based on behaviour and environment monitoring. 74 p. (179 p). Yhteenveto 1 p. 2007.
75 SILTANEN, JARMO, Quality of service and dynamic scheduling for traffic engineering in next generation networks. 88 p. (155 p.) 2007.
76 KUUVA, SARI, Content-based approach to experiencing visual art. - Sisältöperustainen lähestymistapa visuaalisen taiteen kokemiseen. 203 p. Yhteenveto 3 p. 2007.
77 RUOHONEN, TONI, Improving the operation of an emergency department by using a simulation model. 164 p. 2007.
78 NAUMENKO, ANTON, Semantics-based access control in business networks. 72 p. (215 p.) Yhteenveto 1 p. 2007.
79 WAHLSTEDT, ARI, Stakeholders' conceptions of learning in learning management systems development. - Osallistujien käsitykset oppimisesta oppimisympäristöjen kehittämisessä. 83 p. (130 p.) Yhteenveto 1 p. 2007.
80 ALANEN, OLLI, Quality of service for triple play services in heterogeneous networks. 88 p. (180 p.) Yhteenveto 1 p. 2007.
81 NERI, FERRANTE, Fitness diversity adaptation in memetic algorithms. 80 p. (185 p.) Yhteenveto 1 p. 2007.
82 KURHINEN, JANI, Information delivery in mobile peer-to-peer networks. 46 p. (106 p.) Yhteenveto 1 p. 2007.
83 KILPELÄINEN, TURO, Genre and ontology based business information architecture framework (GOBIAF). 74 p. (153 p.) Yhteenveto 1 p. 2007.
84 YEVSEYEVA, IRYNA, Solving classification problems with multicriteria decision aiding approaches. 182 p. Yhteenveto 1 p. 2007.
85 KANNISTO, ISTO, Optimized pricing, QoS and segmentation of managed ICT services. 45 p. (111 p.) Yhteenveto 1 p. 2007.
86 GORSHKOVA, ELENA, A posteriori error estimates and adaptive methods for incompressible viscous flow problems. 72 p. (129 p.) Yhteenveto 1 p. 2007.
87 LEGRAND, STEVE, Use of background real-world knowledge in ontologies for word sense disambiguation in the semantic web. 73 p. (144 p.) Yhteenveto 1 p. 2008.
88 HÄMÄLÄINEN, NIINA, Evaluation and measurement in enterprise and software architecture management. - Arviointi ja mittaaminen kokonais- ja ohjelmistoarkkitehtuurien hallinnassa. 91 p. (175 p.) Yhteenveto 1 p. 2008.
89 OJALA, ARTO, Internationalization of software firms: Finnish small and medium-sized software firms in Japan. 57 p. (180 p.) Yhteenveto 2 p. 2008.
90 LAITILA, ERKKI, Symbolic Analysis and Atomistic Model as a Basis for a Program Comprehension Methodology. 321 p. Yhteenveto 3 p. 2008.
91 NIHTILÄ, TIMO, Performance of Advanced Transmission and Reception Algorithms for High Speed Downlink Packet Access. 93 p. (186 p.) Yhteenveto 1 p. 2008.
92 SETÄMAA-KÄRKKÄINEN, ANNE, Network connection selection-solving a new multiobjective optimization problem. 52 p. (111p.) Yhteenveto 1 p. 2008.
93 PULKKINEN, MIRJA, Enterprise architecture as a collaboration tool. Discursive process for enterprise architecture management, planning and development. 130 p. (215 p.) Yhteenveto 2 p. 2008.
94 PAVLOVA, YULIA, Multistage coalition formation game of a self-enforcing international environmental agreement. 127 p. Yhteenveto 1 p. 2008.
95 NOUSIAINEN, TUULA, Children's involvement in the design of game-based learning environments. 297 p. Yhteenveto 2 p. 2008.
96 KUZNETSOV, NIKOLAY V., Stability and oscillations of dynamical systems. Theory and applications. 116 p. Yhteenveto 1 p. 2008.
97 KHRIYENKO, OLEKSIY, Adaptive semantic Web based environment for web resources. 193 p. Yhteenveto 1 p. 2008.
98 TIRRONEN, VILLE, Global optimization using memetic differential evolution with applications to low level machine vision. 98 p. (248 p.) Yhteenveto 1 p. 2008.
99 VALKONEN, TUOMO, Diff-convex combinations of Euclidean distances: A search for optima. 148 p. Yhteenveto 1 p. 2008.
100 SARAFANOV, OLEG, Asymptotic theory of resonant tunneling in quantum waveguides of variable cross-section. 69 p. Yhteenveto 1 p. 2008.
101 POZHARSKIY, ALEXEY, On the electron and phonon transport in locally periodical waveguides. 81 p. Yhteenveto 1 p. 2008.
102 AITTOKOSKI, TIMO, On challenges of simulation-based globaland multiobjective optimization. 80 p. (204 p.) Yhteenveto 1 p. 2009.
103 YALAHO, ANICET, Managing offshore outsourcing of software development using the ICT-supported unified process model: A cross-case analysis. 91 p. (307 p.) Yhteenveto 4 p. 2009.
104 KOLLANUS, SAMI, Tarkastuskäytänteiden kehittäminen ohjelmistoja tuottavissa organisaatioissa. - Improvement of inspection practices in software organizations. 179 p. Summary 4 p. 2009.
105 LEIKAS, JAANA, Life-Based Design. 'Form of life' as a foundation for ICT design for older adults. - Elämälähtöinen suunnittelu. Elämänmuoto ikääntyville tarkoitettujen ICT tuotteiden ja palvelujen suunnittelun lähtökohtana. 218 p. (318 p.) Yhteenveto 4 p. 2009.

106 Vasilyeva, Ekaterina, Tailoring of feedback in web-based learning systems: Certitude-based assessment with online multiple choice questions. 124 p. (184 p.) Yhteenveto 2 p. 2009.

107 Kudryashova, ElenaV., Cycles in continuous and discrete dynamical systems. Computations, computer assisted proofs, and computer experiments. 79 p. (152 p.) Yhteenveto 1 p. 2009.

108 Blackledge, Jonathan, Electromagnetic scattering and inverse scattering solutions for the analysis and processing of digital signals and images. 297 p. Yhteenveto 1 p. 2009.

109 Ivannikov, Andriy, Extraction of event-related potentials from electroencephalography data. - Herätepotentiaalien laskennallinen eristäminen EEG-havaintoaineistosta. 108 p. (150 p.) Yhteenveto 1 p. 2009.

110 kalyakin, Igor, Extraction of mismatch negativity from electroencephalography data. - Poikkeavuusnegatiivisuuden erottaminen EEG-signaalista. 47 p. (156 p.) Yhteenveto 1 p. 2010.

111 Heikkilä, Marikka, Coordination of complex operations over organisational boundaries. 265 p. Yhteenveto 3 p. 2010.

112 Fekete, Gábor, Network interface management in mobile and multihomed nodes. 94 p. (175 p.) Yhteenveto 1 p. 2010.

113 Kujala, Tuomo, Capacity, workload and mental contents - Exploring the foundations of driver distraction. 146 p. (253 p.) Yhteenveto 2 p. 2010.

114 Lugano, Giuseppe, Digital community design - Exploring the role of mobile social software in the process of digital convergence. 253 p. (316 p.) Yhteenveto 4 p. 2010.

115 Kampylis, Panagiotis, Fostering creative thinking. The role of primary teachers. - Luovaa ajattelua kehittämässä. Alakoulun opettajien rooli. 136 p. (268 p.) Yhteenveto 2 p. 2010.

116 Toivanen, Jukka, Shape optimization utilizing consistent sensitivities. - Muodon optimointi käyttäen konsistentteja herkkyyksiä. 55 p. (130p.) Yhteenveto 1 p. 2010.

117 Mattila, Keijo, Implementation techniques for the lattice Boltzmann method. - Virtausdynamiikan tietokonesimulaatioita Hila-Boltzmann -menetelmällä: implementointi ja reunaehdot. 177 p. (233 p.) Yhteenveto 1 p. 2010.

118 Cong, Fengyu, Evaluation and extraction of mismatch negativity through exploiting temporal, spectral, time-frequency, and spatial features. - Poikkeavuusnegatiivisuuden (MMN) erottaminen aivosähkönauhoituksista käyttäen ajallisia, spektraalisia, aikataajuus - ja tilapiirteitä. 57 p. (173 p.) Yhteenveto 1 p. 2010.

119 Liu, Shenghua, Interacting with intelligent agents. Key issues in agent-based decision support system design. 90 p. (143 p.) Yhteenveto 2 p. 2010.

120 Airaksinen, Tuomas, Numerical methods for acoustics and noise control. - Laskennallisia menetelmiä akustisiin ongelmiin ja melunvaimennukseen. 58 p. (133 p.) Yhteenveto 2 p. 2010.

121 Weber, Matthieu, Parallel global optimization Structuring populations in differential evolution. - Rinnakkainen globaalioptimointi. Populaation rakenteen määrittäminen differentiaalievoluutiossa. 70 p. (185 p.) Yhteenveto 2 p. 2010.

122 Väärämäki, Tapio, Next generation networks, mobility management and appliances in intelligent transport systems. - Seuraavan sukupolven tietoverkot, liikkuvuuden hallinta ja sovellutukset älykkäässä liikenteessä. 50 p. (111 p.) Yhteenveto 1 p. 2010.

123 Viukari, Leena, Tieto- ja viestintätekniikka-välitteisen palvelun kehittämisen kolme diskurssia. - Three discourses for an ICT-service development . 304 p. Summary 5 p. 2010.

124 Puurtinen, Tuomas, Numerical simulation of low temperature thermal conductance of corrugated nanofibers. - Poimutettujen nanokuitujen lämmönjohtavuuden numeerinen simulointi matalissa lämpötiloissa . 114 p. Yhteenveto 1 p. 2010.

125 Hiltunen, Leena, Enhancing web course design using action research . - Verkko-opetuksen suunnittelun kehittäminen toimintatutkimuksen keinoin . 192 p. Yhteenveto 2 p. 2010.

126 Aho, Kari, Enhancing system level performance of third generation cellular networks through VoIP and MBMS services. 121 p. (221 p.). Yhteenveto 2 p. 2010.

127 Häkkinen, Markku, Why alarms fail. A cognitive explanatory model. 102 p. (210 p.). Yhteenveto 1 p. 2010.

128 Pennanen, Anssi, A graph-based multigrid with applications. - Graafipohjainen monihilamenetelmä sovelluksineen. 52 p. (128 p.). Yhteenveto 2 p. 2010.

129 Ahlgren, Riikka, Software patterns, organizational learning and software process improvement. 70 p. (137 p.). Yhteenveto 1 p. 2011.

130 Nikitin, Sergiy, Dynamic aspects of industrial middleware architectures 52 p. (114 p.). Yhteenveto 1 p. 2011.