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

Editors
Timo Männikkö
Department of Mathematical Information Technology, University of Jyväskylä
Pekka Olsbo
Publishing Unit, University Library of Jyväskylä

ISBN 978-951-39-4230-4
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
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

LIST OF TABLES

CONTENTS

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.

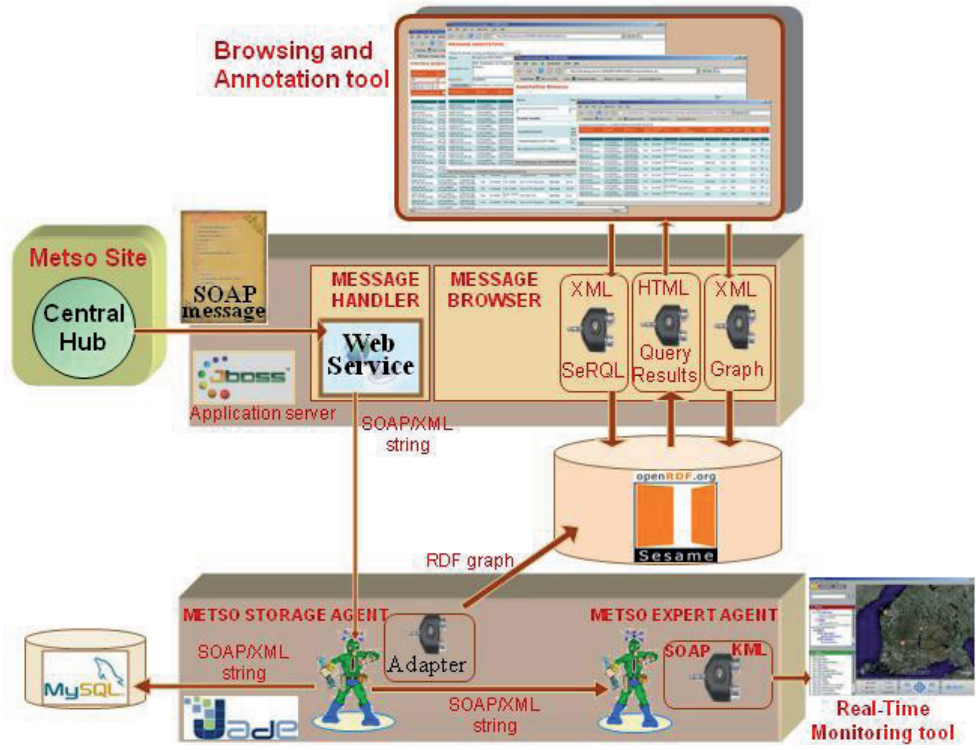

FIGURE 1 Alarm message integration tool

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.

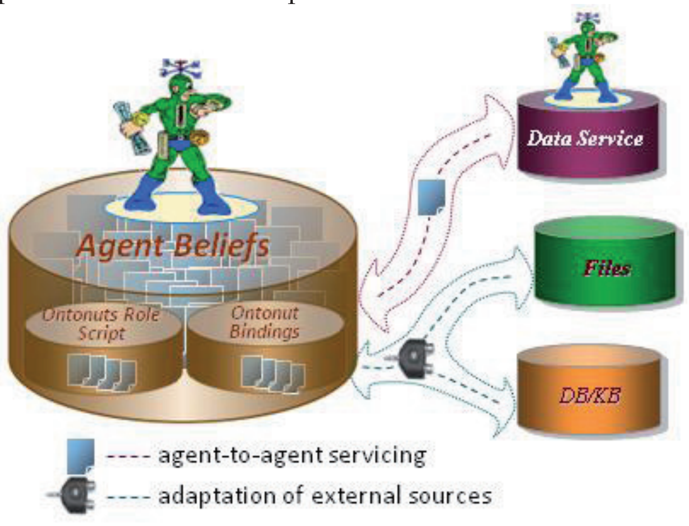

FIGURE 2 Data integration using Ontonuts

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.

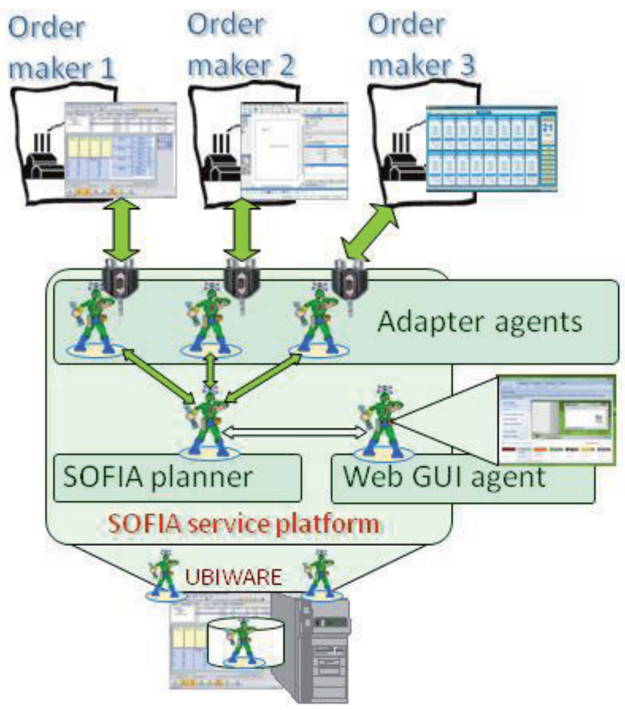

FIGURE 3 SOFIA platform architecture

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.

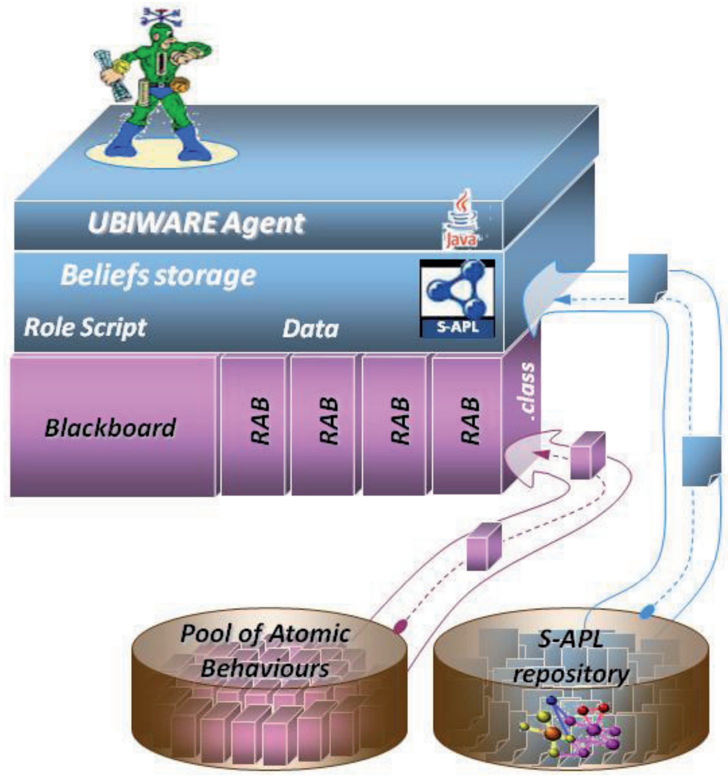

FIGURE 4 Ubiware Agent architecture

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

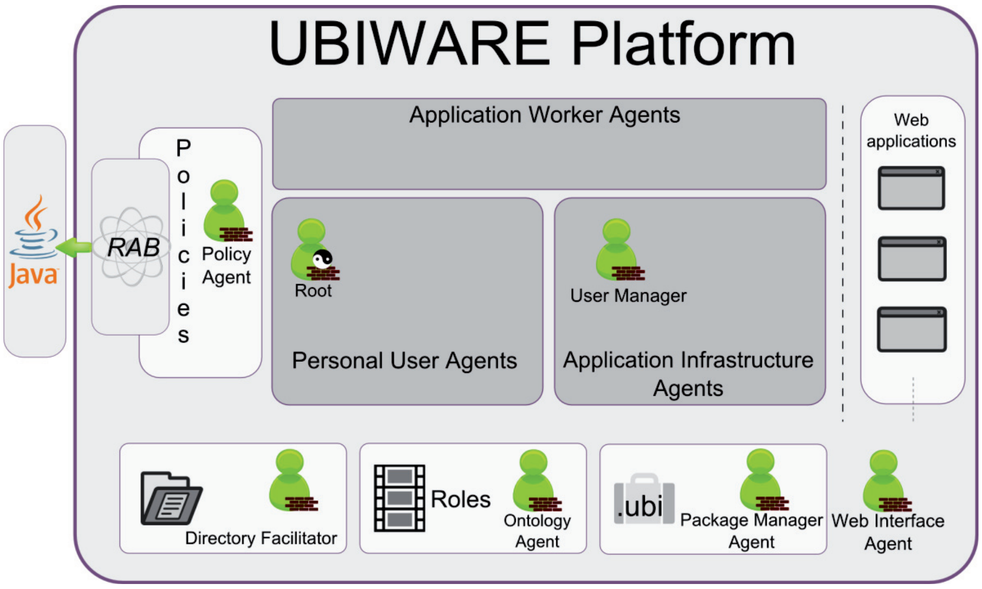

FIGURE 5 UBIWARE platform architecture

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.

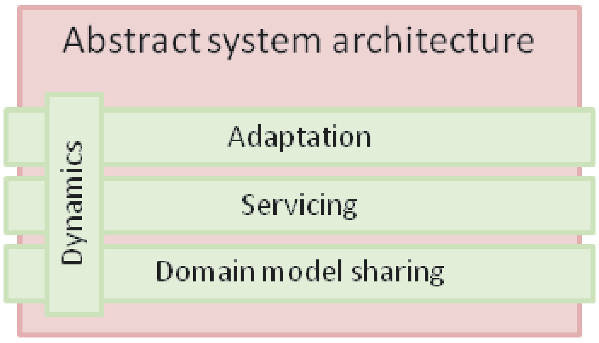

FIGURE 6 Aspects of the abstract system architecture

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.

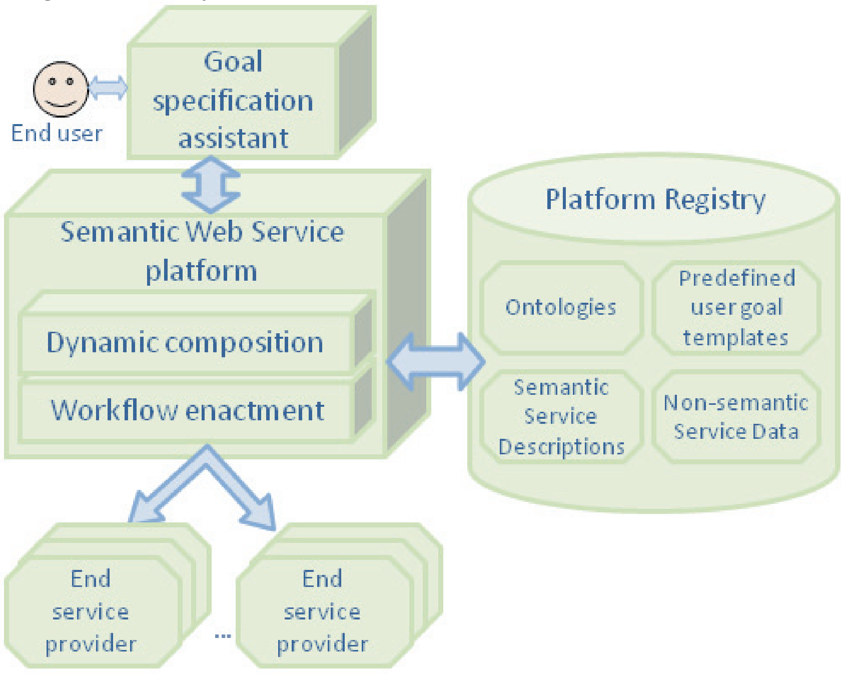
End user
Goal specification assistant
Semantic Web Service platform
Dynamic composition
Workflow enactment
Platform Registry
Ontologies
Predefined user goal templates
Semantic Service Descriptions
Non-semantic Service Data
End service provider
...
End service provider

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.

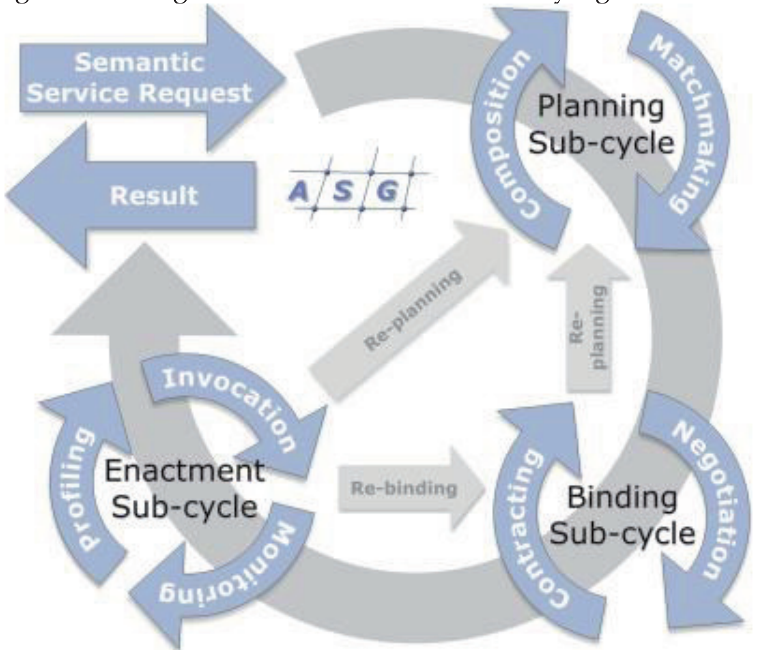

FIGURE 8 Service delivery process (adopted from (ASG, 2004))

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).

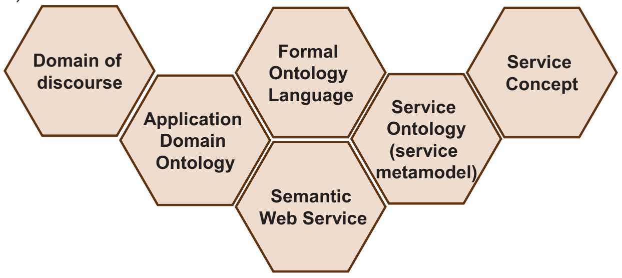

FIGURE 9 Foundation Models for Semantic Web Services

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).

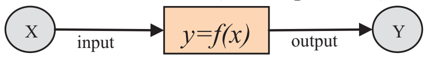

FIGURE 10 Mathematical function as a capability

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.

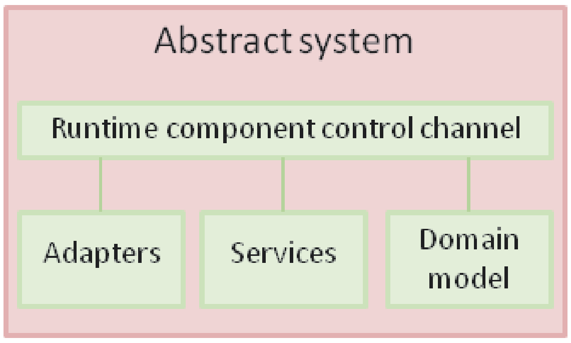

FIGURE 11 Dynamics as a component control channel

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 existing 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/

ORIGINAL ARTICLES

I

QUERYING DYNAMIC AND CONTEXT-SENSITIVE METADATA IN SEMANTIC WEB

by

Sergiy Nikitin, Vagan Terziyan, Yaroslav Tsaruk and Andriy Zharko 2005

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.

Querying Dynamic and Context-Sensitive Metadata in Semantic Web

Sergiy Nikitin, Vagan Terziyan, Yaroslav Tsaruk, and Andriy Zharko

Industrial Ontologies Group, Agora Center, University of Jyväskylä,
P.O. Box 35, FIN-40014 Jyväskylä, Finland
senikiti@cc.jyu.fi

Abstract. RDF (core Semantic Web standard) is not originally appropriate for context representation, because of its initial focus on the ordinary Web resources, such as web pages, files, databases, services, etc., which structure and content are more or less stable. However, on the other hand, emerging industrial applications consider e.g. machines, processes, personnel, services for condition monitoring, remote diagnostics and maintenance, etc. to be specific classes of Web resources and thus a subject for semantic annotation. Such resources are naturally dynamic, not only from the point of view of changing values for some attributes (state of resource), but also from the point of view of changing "status-labels" (condition of the resource). Thus, context-awareness and dynamism appear to be new requirements to the existing RDF. This paper discusses the issues of representing the contexts in RDF and constructions coming with context representation. We discover certain representation patterns and their classification towards development of the general approach of querying dynamic and context-sensitive metadata in Semantic Web by autonomous agents.

1 Introduction

Emerging Semantic Web technology offers a Resource Description Framework (RDF) as a standard for semantic annotation of Web resources. It is expected that Web content with RDF-based metadata layer and ontological basis for it will be enough to enable interoperable and automated processing of Web data by various applications. RDF-based tools, e.g. Hewlett-Packard's Jena [14] and Stanford's Protégé [15] provide a base for reasoning about metadata and about situated data (entities situated in time and space) that is superior to alternatives such as relational databases or object-oriented databases. However, according e.g. to [10] essential representational ability is missing from the current generation of Semantic Web tools and languages. When that ability is added, the resulting capabilities offer a combination of novelty and flexibility that may usher in a wave of commercial Semantic Web tool-based applications. Evidently the existing RDF tools should be extended to support contexts to enable querying a set of RDF statements having common temporal, spatial or other metadata attributes. In [10] it was concluded that the "clear winners" for possible solution can be quads (i.e. adding a fourth field of type 'context' to each RDF triple) and a context mechanism that references individuals instead of statements. Another attempt has been made recently to add C-OWL (Context OWL), an extended language with an enriched semantics which allows us to contextualize ontologies, namely, to localize their contents (and, therefore, to make them not visible to the outside) and to allow for explicit mappings (bridge rules). The core open issue is the tension between how much knowledge should be shared and globalized (via ontologies) and how much should be localized with limited and controlled forms of globalization (via contexts) [11]. In [12] the usage of context- and content-based trust mechanisms have been proposed and the cRDF trust architecture was presented which allows the formulation of subjective and task-specific trust policies as a combination of reputation-, context- and content-based trust mechanisms. There exist different ways how to understand and use context information for RDF data. In [13] these different ways have been summarized and the RDF-Source related Storage System (RDF-S3) has been proposed. RDF-S3 aimed to keep track of the source information for each stored RDF triple. On top the RDF-S3 has an extended version of *easy RQL* (eRQL) that makes use of the source information supported by RDF-S3. Therefore queries can be restricted to trusted sources and results can be viewed inside their RDF graph context. Two main arguments are stated in [13] for using context nodes instead of quads. First, quads are not compatible with the RDF model and second, the distinction between the given RDF information and information that is given in addition, like external context information, is much more complicated when using quads, whereas additional context nodes can be easily distinct from RDF triples. Therefore context nodes were used instead of context parts (quads).

There is not yet clear vision, which way is better (triples or quads) for representing contextual metadata in RDF. Another issue is for what kind of resources such descriptions will be required. On one hand the ordinary Web resources, such as web pages, files, databases, services, etc., which structure and content are more or less stable, probably do not need a specific way of context representation. However, on the other hand, emerging industrial applications consider e.g. machines, processes, personnel, services for condition monitoring, remote diagnostics and maintenance, etc. represent specific classes of Web resources and thus a subject for semantic annotation. Such resources are naturally dynamic, not only from the point of view of changing values for some attributes (state of resource) but also from the point of view of changing "status-labels" (condition of the resource). In our former effort within SmartResource project [16] we presented Resource State/Condition Description Framework (RscDF), as an extension to RDF, which introduces upper-ontology for describing such characteristics of resources as states and corresponding conditions, dynamics of state changes, target conditions and historical data about previous states. These descriptions are supposed to be used by external Web-services (e.g. condition monitoring, remote diagnostics and predictive maintenance of the resources). We presented RscDF as temporal and contextual extensions of RDF and discussed a State-Symptom-Diagnosis-Decision-Maintenance model as the basis for RscDF schema.

RSCDF is a unified representation format for resource state and condition description (encoding). RscDF-language formalizes context definition structure. RscDF-Schema defines main concepts and structure of the language. The structure is highly flexible, thus allowing definition of different complex constructions over the basic statements. Different definitions being used for resource description must refer to or define instances of classes from Industrial Maintenance Ontology. Detailed description of RscDF-language is not in a scope of this paper, so we refer to [17]. Figure 1 shows the key element of RscDF – “SR_Statement”.

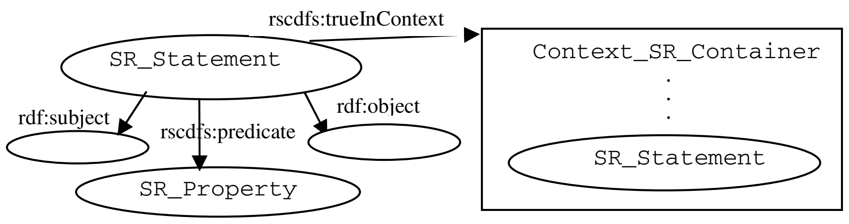

Fig. 1. SR_Statement structure

The SR_Statement defines the basic structure of statements being used in RscDF. The combinations of statements and references to statements and statement containers may form highly structured semantic description. The important semantics are represented by SR_Property class and its subproperties. The property in the rscdfs:predicate container defines the type and structure of rdf:object of current SR_Statement. However, the property specification defines only domain and range. So to know the structure of the statement, we have to attach some pattern description to SR_Property.

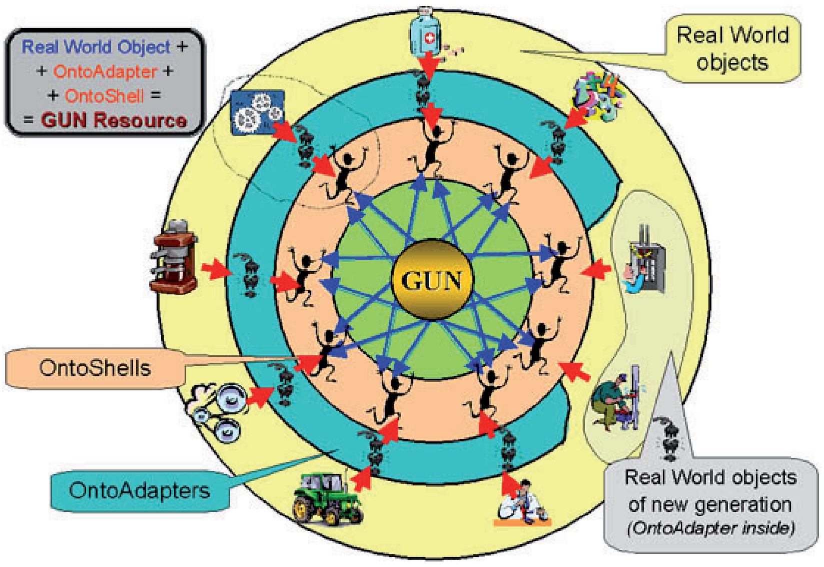

Fig. 2. GUN concept illustrated (adopted from [1])

The RscDF language was designed to serve the concept of a Global Understanding Environment [1]. GUN concept utilises Semantic Web approach for resource annotation and ontology-based semantic representation and describes communities of interacting Smart Resources. GUN provides a framework for making resources smart, for interaction, collaboration, coordination of these resources and resource discovery support. Types of resources are not restricted to traditional web content, but can be physical resources from real world, such as humans and devices (see Figure 2).

GUN paradigm provides every participant with common structured data representation format, allowing explicit and unambiguous knowledge sharing. In order to become GUN participant certain steps of adaptation should be taken. In GUN development our research group focuses on industrial case study that is concerned with large-scale platforms for automated management of industrial objects. The adaptation process to GUN environment is described in General Adaptation Framework [18]. "General adaptation" assumes a design of a sufficient framework for an integration of different (by structure and nature) resources into the GUN environment. This environment will provide a mutual interaction between heterogeneous resources. Adaptation assumes elaboration of a common mechanism for new resource integration, and its provision with a unified way of interaction. The main idea of adaptation is based on a concept of "adapter", which plays role of a bridge between an internal representation of resource and a unified environment.

Adapter is a software component, which provides a bidirectional link between a resource interface and an interface of the environment. GUN assumes interoperability of Smart Resources. Smart Resource is a conjunction of Real World Resource (RWR), Adapter and Agent. By extending RWR within Adapter and Agent we make it GUN compatible. General Adaptation includes development of Adapter for RWR. Adaptation to GUN is not just syntactical transformation from one representation format to another. The key element of adaptation is mapping of concepts being used by "Real-World-Resource" to Industrial Maintenance Ontology (IMO) elements. The role of IMO lies in unification and structuring of data being represented in such way that every resource description taking part in GUN must refer to it.

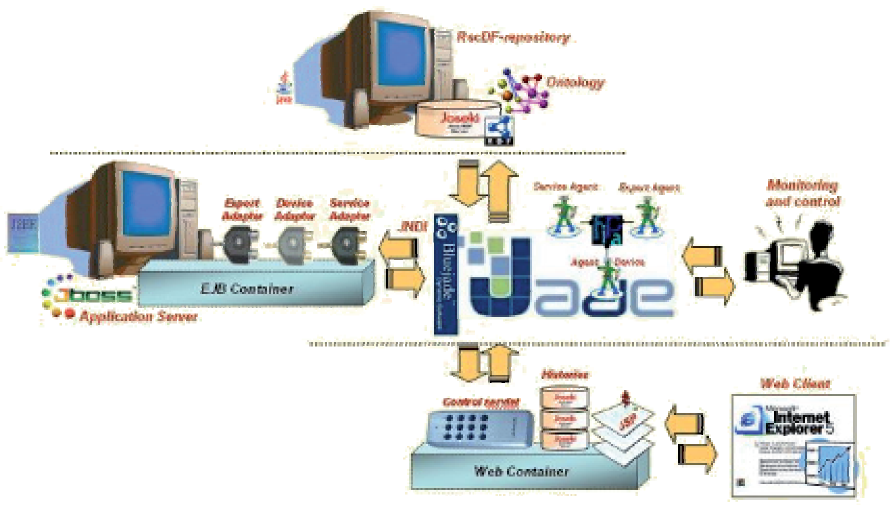

Fig. 3. SmartResource as a Multi-Agent System

Semantic Web standards are not yet supporting semantic descriptions of resources with proactive behavior. However, as the research within the SmartResource project shows [16], to enable effective and predictive maintenance of an industrial device in distributed and open environment, it will be necessary to have autonomous agent based monitoring over device state and condition and also support from remote diagnostics Web-Services (see Figure 3).

This means that the description of a device as a resource will require also the description of proactive behavior of autonomous condition monitoring applications (agents, services) towards effective and predictive maintenance of the device. For that we plan to develop in 2005 another extension of RDF, which is Resource Goal/Behavior Description Framework (RGBDF) to enable explicit specification of maintenance goals and possible actions towards faults monitoring, diagnostics and maintenance. Based on RSCDF and RGBDF and appropriate ontological support, we also plan to design RSCDF/RGBDF platforms for smart resources (devices, Web-services and human experts) equipped by adapters and agents for proactivity, and then to apply several scenarios of communication between the platforms towards learning Web-services based on device data and expert diagnostics to enable automated remote diagnostics of devices by Web-services.

In this paper we present our solution how to manage (according to the structure of the paper Section 2 describes about storing and Section 3 is dedicated to querying) the context-sensitive metadata for applications compatible with Semantic Web and GUN concepts by utilising existing technologies and tools. Some examples with industrial metadata are also provided.

2 Storing RDF-Based Metadata

Nowadays there are a lot of proposals related to storing RDF data in RDF databases, each with different client-server protocols and different client APIs. For our purposes we surveyed a number of most popular RDF-storages (Kowari[1], Sesame[2], Joseki[3]) and selected Joseki storage as most suitable allowing access to RDF-data through HTTP.

2.1 Joseki

Joseki has been proposed and maintained by Semantic Web group at HP Labs. Joseki is a web application for publishing RDF models on the web and realized useful access to models through HTTP protocol. This allows getting easy access to model from anywhere you want. It is built on Jena and, via its flexible configuration, allows a Model to be made available on a specified URL and queried using a number of languages. Results can be returned as RDF/XML, RDF/N3, or NTriples. The query languages, result formats, and model sources can be extended to produce new alternatives tailored to the user's needs.

[1] http://www.kowari.org/
[2] http://www.openrdf.org/
[3] http://www.joseki.org/

2.2 Storing and Extracting Data in Joseki

Information stored in Joseki are presented in a format of models. The client application has an access to a specified model and executes operation on this model. Operations that can be done upon the remote model:

- add new model or statement
- remove model or statement
- extract data from storage

New model can be appended to already existing model on the Joseki server. This operation also allows appending new statement to the predefined model. Each model or statement can be removed from the storage by using the remove operation.

Data extraction from Joseki storage can be implemented by using different mechanisms:

- fetch the whole model
- SPO query (single triple match language)
- RDQL query

Information from the storage can be extracted partly or as a whole model. To extract the whole model the fetch mechanism is used. For extracting just specified information, SPO and RDQL queries are available. SPO (also known as "Triples") is an experimental minimal query language. An SPO query is a single triple pattern, with optional subject (parameter "s"), predicate (parameter "p"), and object (parameter "o", if an URIref or parameter "v" for a string literal). Absence of the parameter implies "any" for matching that slot of the triple pattern.

RDQL is a query language, which is similar to SQL (Structured Query Language) and allows specifying the set of conditions, which should suite the extracted set of statements.

The architecture of Joseki is presented in Figure 4.

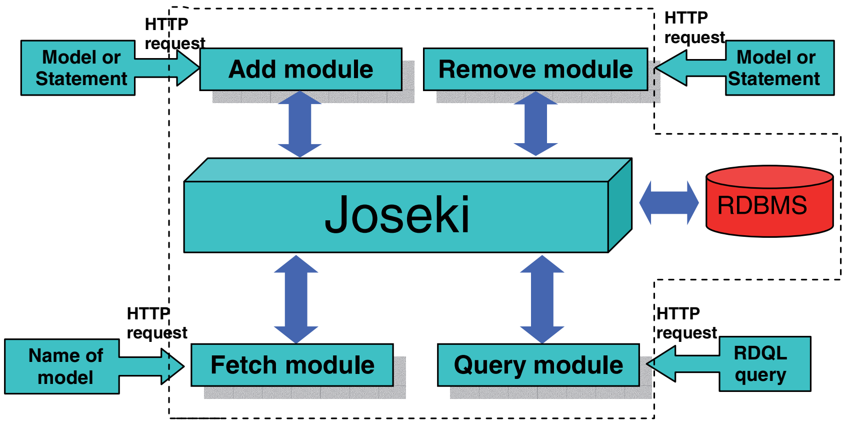

Fig. 4. Architecture of Joseki storage

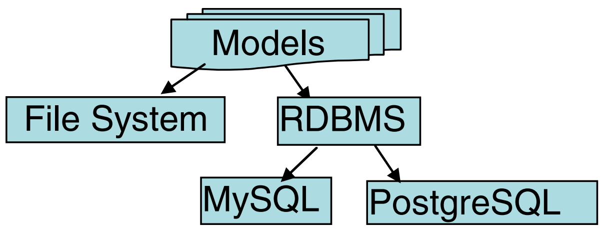

Fig. 5. Types of storing models in Joseki

It consists of the core module and modules, which execute specialized functions on the remote model (fetching, adding, removing, querying). Interaction between the client and the Joseki server is implemented through HTTP query. The type of the query depends on the length of the query. It could be GET if query is not longer then 200 characters, otherwise POST method is used. Each model in Joseki server has a predefined set of operations, which could be executed upon it. When server gets query to one of the defined models, it checks the list of operations which could be executed and if the operation is not specified it responds by a fail message. Each operation is executed by a specified module. As an input each module requires special parameters. For example, Addition and Remove modules need as an input model or statement, which have to be added or removed. As a response Joseki sends empty model, if the operation was successful.

One more optional component is RDBMS assigned for storing models in a persistent storage. The models in Joseki can be saved in two ways (See Figure 5):

- to a file
- to a RDBMS.

The target RDBMS is specified in the configuration file joseki-db.n3.

2.3 RDQL

Resource Description Query Language (RDQL) is a query language for RDF. RDQL is an implementation of the SquishQL RDF query language and is similar to SQL. It borrows basic set of words for specifying the set of data, which should be returned (e.g. SELECT, WHERE, FROM, etc). As a condition for extracting, RDQL provides, the "WHERE" clause followed by a list of triples (subject, predicate, object). These triples define the pattern for a search. RDQL has one more key word for defining a space of URI identifiers. It allows avoiding long names.

In the sample query presented below (SELECT-query), as a result, two values will be returned: Matthew and Jones. At the beginning of query we specify the values, which should be returned: "`?family`" and "`?given`". The first condition determines a statement, which has `vcard:FN` property value "Matt Jones". Then we extract data from property `vcard:N` to the variable "name". Basing on this information, we extract values of the property `vcard:Family` and `vcard:Given`.

```
SELECT ?family, ?given
WHERE (?vcard vcard:FN "Matt Jones")
```

```
      (?vcard vcard:N ?name)
      (?name vcard:Family ?family)
      (?name vcard:Given ?given)
USING vcard FOR <http://www.w3.org/2001/vcard-rdf/3.0#>
```

Scheme on Figure 6 shows the steps of the query execution. The names of nodes are presented as names of variables to make picture clearer. The values of variables "`vcard`" and "`name`" are used as an input to the next condition statements.

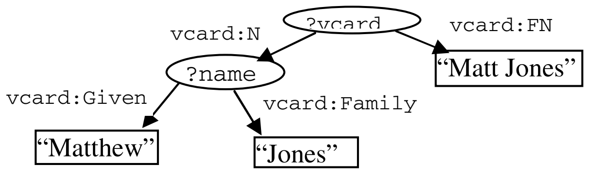

Fig. 6. Query description scheme

3 RscDF Data Management in GUN

The capabilities GUN provides rely on the common data representation format (RSCDF) and the common understanding of domain (Industrial Maintenance Ontology). As far as RSCDF is RDF-compatible, we reuse already existing RDF-databases to store RSCDF data.

.

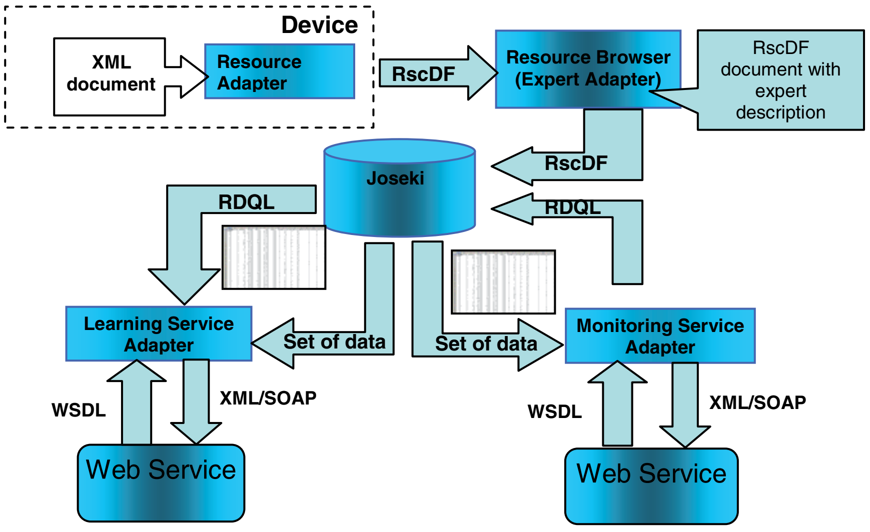

Fig. 7. Presentation scenario

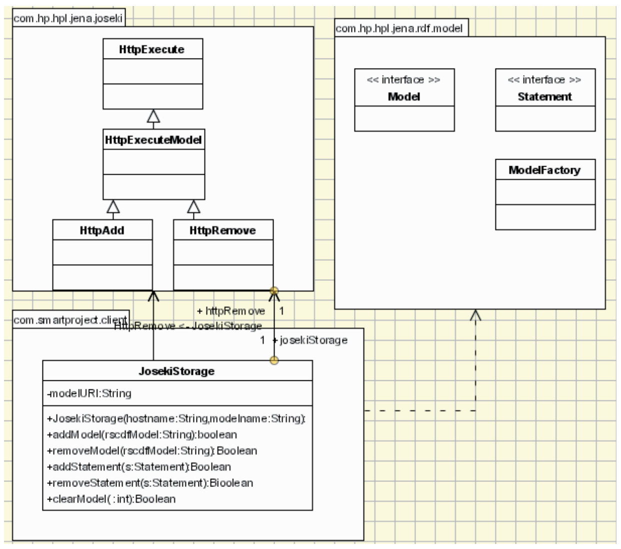
com.hp.hpl.jena.joseki
HttpExecute
HttpExecuteModel
HttpAdd
HttpRemove
com.hp.hpl.jena.rdf.model
<< interface >>
Model
<< interface >>
Statement
ModelFactory
com.smartproject.client
+ httpRemove
1
HttpRemove <- JosekiStorage
1
+ josekiStorage
JosekiStorage
-modelURI:String
+JosekiStorage(hostname:String,modelname:String):
+addModel(rscdfModel:String):boolean
+removeModel(rscdfModel:String):Boolean
+addStatement(s:Statement):Boolean
+removeStatement(s:Statement):Bioolean
+clearModel(: int):Boolean

Fig. 8. Class diagram showing classes needed for interaction with the Joseki server

3.1 Applying RDQL to RscDF Querying

When querying RscDF data we deal with Statement objects that has the additional property `rscdfs:trueInContext`. When selecting a Statement about an object having certain property, we have to consequently apply queries, specifying the rdf:object, rscdfs:predicate or rdf:subject property values, so the query may look like:

```
SELECT ?stmts
WHERE
(?stmts,<rdf:subject>,<papmDescr:123456XZ24>),
(?stmts,<rscdfs:predicate>,<measureOnt:surfacelevel>)
USING
papmDescr  FOR
<http://www.cc.jyu.fi/~olkhriye/rscdfs/resource/resourc
eInstanceDescription#>,
rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
rscdfs FOR
<http://www.cc.jyu.fi/~olkhriye/rscdfs/0.3/rscdfs#>,
measureOnt FOR
<http://www.cc.jyu.fi/~olkhriye/rscdfs/0.3/ontologies/m
easurementOntology#>
```

The resulting variable `stmts` will contain the set of Statements, whose subject and predicate properties satisfy the condition presented in Figure 9 in a form of a graph.

However, when the query contains context-related parts, we meet a problem of representing it in the RDQL language. The query becomes difficult to read, because of additional constructions. For example, when the statements describing different object properties at the certain moment of time, should be selected, we have to specify the value of time-statement lying in the context container.

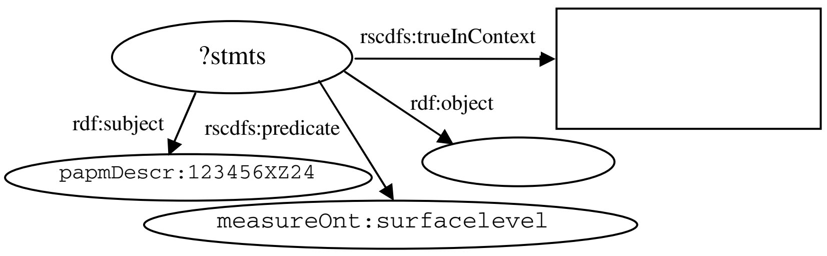

Fig. 9. Visual query representation

3.2 Querying Patterns

RscDF language provides a facility to select Statements by a certain template. The template Statement is put to the context container of Statement, wrapping the Statements selected according to the template. Figure 10 shows the structure of Statement, being created as a result of data collection according to a certain template.

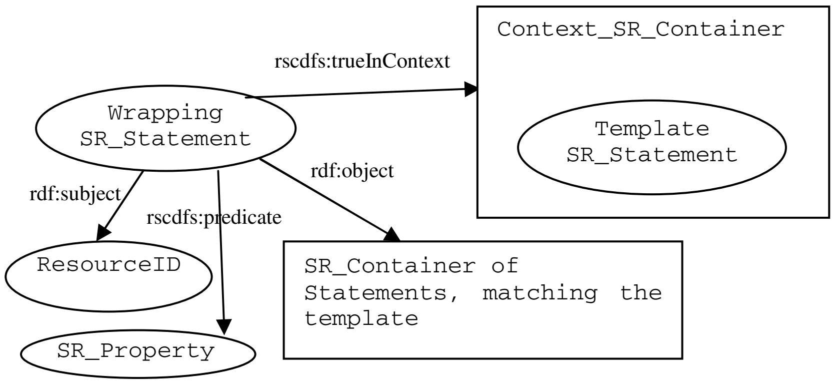

Fig. 10. Data Collection Statement selecting data according to a template

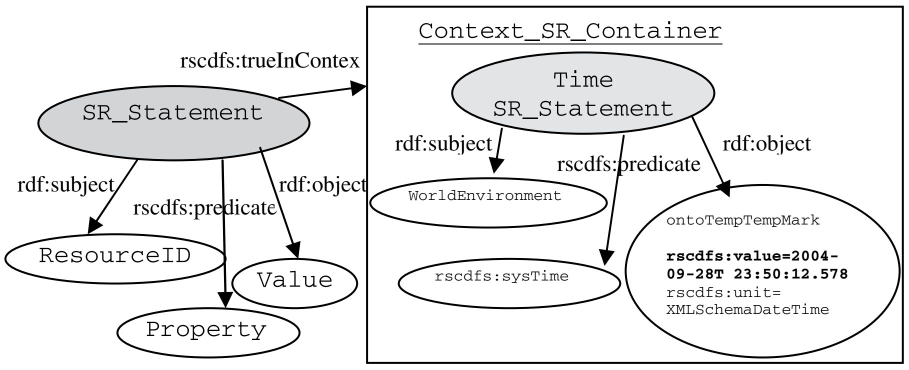

Fig. 11. Statement with time context

The most vivid example of the template context-dependent data collection is State-template data collection. For example, we have a certain resource, logging a track of its states. Different Statements about resource states are marked with time. So, the Statements will contain Statement about time in context container (Figure 11).

Figure 12 shows the data collection (subhistory) statement. The `rdf:object` property contains reference to container with Statements, matching the data collection template. The data collection template Statement is placed to the Context_SR_Container of the State Statement.

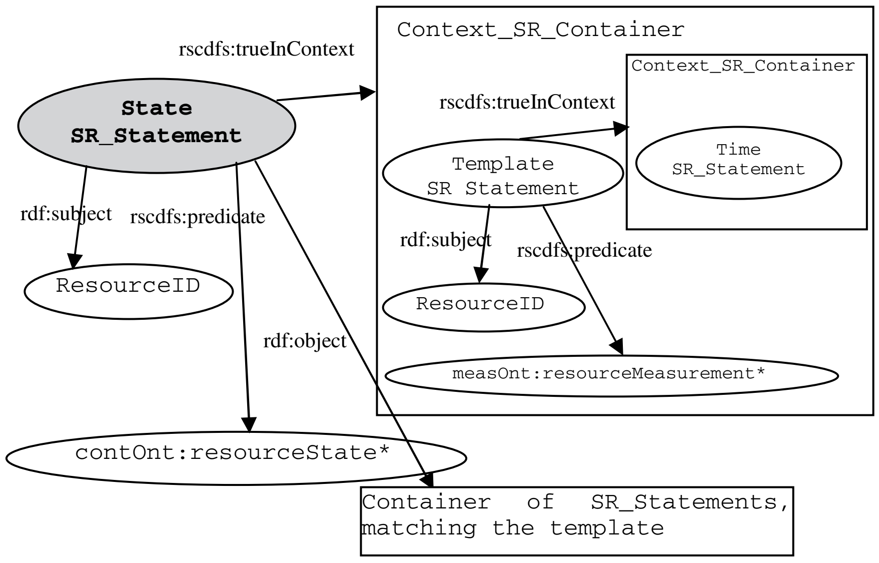

Fig. 12. Collecting Statement of State template

All the names marked with (*) are not actually present in Industrial Maintenance Ontology, but mean more generic classes or properties.

To apply the query, to data collection structures residing in RDF storage via RDQL language we have to write a number of routine triple queries, so it is reasonable to discover certain RDQL templates combining the operations into blocks and asking only start input data for further query execution. In case of State template we have discovered following routines:

– State RscDF-Statement To “attribute-value pairs” Routine:

Input	**Output**
Pointer to State Statement	Set of attribute-value pairs of one state

– Subhistory Statement to “Set of State Records”

Input	**Output**
Pointer to Subhistory Statement	Set of attribute-value pairs of correspondent states

Further on we omit namespaces definition and `USING` clause. For the first case the RDQL query looks like:

```
SELECT ?ValueStatements, ?NumUnits, ?NumValues
WHERE
(<StateStmtID>, <rdf:object>, ?StateContainer),
(?StateContainer, <rscdfs:member>, ?ValueStatements),
(?ValueStatements, <rdf:object>, ?NumValueInstances),
(?NumValueInstances, <rscdfs:value>,?NumValues),
(?NumValueInstances, <rscdfs:unit>, ?NumUnits)
```

The output of the query is a plain 3-column table with a set of rows. It is implied that every record in the table belongs to State, hence here we have 3 output variables, but in cases, when this routine is used as a subroutine, we have to return also the State Statement identifiers in order to be able to identify then relationships of values to states. Table 1 illustrates a possible output of the query:

Table 1. RDQL Query output

Statement ID	Units	Value
somens:valueStatementID_1	measureUnitsOnt:temperatureCelsius	70
somens:valueStatementID_2	measureUnitsOnt:roundsPerMinute	1500

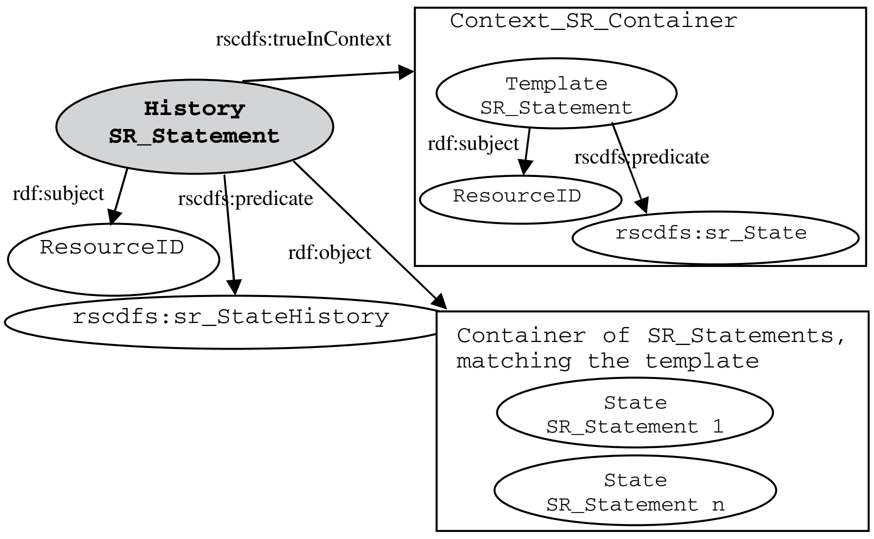

Fig. 13. History Statement

We put Statement ID to query output, because it uniquely identifies the belonging of values and units and allows further inference upon received results. As far as

RDQL query result is displayed in one non-normalized table, we store redundant data, but save the semantics.

The routine logic can be wrapped as a method, whose input is the name of Statement and output - RDQL subroutine. Example in Figure 13 shows more complex logic. It reuses previous example of State data selection, but provides a Set of States.

Below is the RDQL query:

```
SELECT?StateStmts,?ValueStatements,?NumUnits, ?NumVaues
WHERE
(<HistoryStmtID>, <rdf:object>, ?StatesCont),
(?StatesCont, <rscdfs:member>, ?StateStmts),
(?StateStmts, <rdf:object>, ?StateContainers),
(?StateContainers, <rscdfs:member>, ?ValueStatements),
(?ValueStatements, <rdf:object>, ?NumValueInsts),
(?NumValueInsts, <rscdfs:value>,?NumValues),
(?NumValueInsts, <rscdfs:unit>, ?NumUnits)
```

The five last strings of the query above are almost equivalent to the State query template. The only difference is presence of variable `?StateStmts` instead of static given value `StateStmtID`. Query doesn't contain any references to types of properties being used in statements because we know beforehand what kind of data we deal with. In general, when the statement's ID is not known, we should first look for it, specifying as a search criteria Resource's ID and property type, for example:

```
SELECT ?stmts
WHERE
(?stmts,<rdf:subject>,<resourceID>),
(?stmts,<rscdfs:predicate>,<rscdfs:sr_StateHistory>)
```

Basically, the SR_Property being pointed by `rscdfs:predicate`, specifies the data template. So it makes sense to develop the ontology of data templates and associate it with SR_Properties.

4 Conclusions

In this paper we tried to analyze the problems of storing and managing context-enabled data via RDF storages. Finally, Joseki RDF storage and querying engine has been chosen as the most appropriate for integration to the prototype platform for adaptation of industrial resources to Semantic Web – pilot system, result of the Adaptation Stage of the SmartResource project. The approach based on the RDQL-patterns has been applied in the logic of the part of General Semantic Adapter, responsible for querying RscDF storages – dynamic and context-sensitive histories of industrial resources (experts, web services and devices). The flexibility of the RDQL-patterns has allowed to design a unified semantic adapter – a mediator between software agents (which implement proactive goal-driven behavior of originally passive industrial resources) and RDF-based storage of the history data of the corresponding industrial resources.

Further, it is planned to apply the developed method based on the RDQL-patterns in the design of querying mechanism for goal/behavior rule storages, which will utilize RGBDF – Resource Goal/Behavior Description Framework. The latter will be designed during the Proactivity Stage of the SmartResource activities as a part of the Pro-GAF – General Proactivity Framework.

Acknowledgements

This research has been performed as part of the SmartResource ("Proactive Self-Maintained Resources in Semantic Web") project in Agora Center (University of Jyväskylä, Finland) and funded by TEKES and industrial consortium of following companies: Metso Automation, TeliaSonera, TietoEnator and Science Park of Jyväskylä.

References

1. 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) 279–291
2. Online Jena API tutorial by B. McBride, "An Introduction to RDF and the Jena RDF API", August 2003, http://jena.sourceforge.net/tutorial/RDF_API/
3. Online Jena tutorial by A. Seaborne, Hewlett Packard, "Jena Tutorial. A Programmer's Introduction to RDQL", April 2002, http://www.hpl.hp.com/semweb/doc/tutorial/RDQL/
4. Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds: Efficient RDF Storage and Retrieval in Jena2, In: I. F. Cruz, V. Kashyap, S. Decker, R. Eckstein (Eds.): Proceedings of SWDB'03, The first International Workshop on Semantic Web and Databases, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7–8 (2003) 131–150
5. Barnell, A.: RDF Objects, Technical Report, Semantic Web Applications Group, Hewlett Packard Laboratories Bristol, Avon, England, November 2002
6. Webpage of Jena on the official website of B. McBride, Hewlett Packard, "Jena, An RDF API in Java", http://www.uk.hpl.hp.com/people/bwm/rdf/jena
7. Lee, R.: Scalability Report on Triple Store Applications, Technical Report, SIMILE project (2004)
8. Beckett, D.: Semantic Web scalability and storage: survey of free software/open source RDF storage systems, Deliverable 10.1 report, SWAD-Europe project (IST-2001-34732) (2002)
9. B. McBride, Jena: Implementing the RDF Model and Syntax Specification, HP Labs in proceedings of the Second International Workshop on the Semantic Web, WWW10, Hong Kong, 1st May 2001
10. MacGregor, R.: In-Young Ko, Representing Contextualized Data using Semantic Web Tools, In Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems, ISWC 2003, October 2003, Sanibal Island, Florida, USA
11. Bouquet, P., Giunchiglia, F., Harmelen, F., Serafini, L. and Stuckenschmidt, H.: Contextualizing Ontologies, *Journal of Web Semantics* **26** (2004) 1–19
12. Bizer, C., Oldakowski, R.: Using Context- and Content-Based Trust Policies on the Semantic Web. In 13th World Wide Web Conference, WWW2004 (Poster) (2004)
13. Official website of RDF-S3 — RDF Source related Storage System, http://www.dbis.informatik.uni-frankfurt.de/~tolle/RDF/RDFS3/

14. Official website of JENA – a Semantic Web Framework for Java, http://jena.sourceforge.net/
15. Official website of the Protégé ontology management tool, http://protege.stanford.edu/
16. Webpage of the SmartResource project, http://www.cs.jyu.fi/ai/OntoGroup/SmartResource_details.htm
17. Kaykova, O., Khriyenko, O., Naumenko, A., Terziyan, V., Zharko, A.: "RSCDF: Resource State/Condition Description Framework", Deliverable 1.1 report, September 2004, SmartResource project, (http://www.cs.jyu.fi/ai/IJWGS-2004_v2.doc)
18. Kaykova, O., Khriyenko, O., Kovtun, D., Naumenko, A., Terziyan, V., Zharko, A.: "GAF: General Adaptation Framework", Deliverable 1.2 report, October 2004, SmartResource project (http://www.cs.jyu.fi/ai/SJIS-2005.doc)

II

SERVICE MATCHING IN AGENT SYSTEMS

by

Anton Naumenko, Sergiy Nikitin and Vagan Terziyan 2006

International Journal of Applied Intelligence, Special Issue on Agent-Based Grid Computing, Vol. 25, No. 2, 2006, ISSN: 0924-669X, pp. 223-237.

Appl Intell (2006) 25:223–237
DOI 10.1007/s10489-006-9655-4

Service matching in agent systems

Anton Naumenko · Sergiy Nikitin · Vagan Terziyan

Abstract The problem of service and resource matching is being actively discussed currently as a new challenging task for the next generation of semantic discovery approaches for Web services and Web agents. A significant advantage is expected when using an ontological approach to semantically describe and query services. A matchmaking problem arises when a service is being queried and it includes the distance measure between the required service description and the one from the service registry. We realized the need to analyze the applicability of different matchmaking methods to agent development tools when implemented according to agent technology specifications such as FIPA. We consider three main groups of cases: matchmaking between classes of service profiles in pure taxonomies, matchmaking between classes in faceted taxonomies, and matchmaking between instances of faceted taxonomies.

Keywords Agent technology · Ontology · Service matching · Similarity

1. Introduction

The problem of service and resource matching is being hotly discussed now as a new challenging task for the next generation of annotation approaches to Web services and resources. The most significant outcome in matching was reached using an ontological approach to the description of a domain. The ontologies and ontology description languages are currently being developed by different standardization organizations such as W3C consortium.

We have met the problem of resource (device, expert, service) matchmaking in the SmartResource project [1, 2]. The SmartResource project, led by the Industrial Ontologies Group, is aimed at developing a framework for the adaptation of different heterogeneous resources to the common so-called Global Understanding Environment (GUN) and provisioning, based on this framework, a higher level agent-based interoperability and resource description management.

A similar problem was met in the Adaptive Services Grid project [3], which concentrated on the integration of two IT worlds such as Semantic Web Services and Grid Computing by an open generic software platform for adaptive services discovery, creation, composition, and enactment to offer new applications on Grid Services providing a well-defined range of quality of service. The matchmaking of resources and services is one of the most challenging tasks of the project. The matchmaking problem addresses the question of the distance measure between objects and because the SmartResource project raised the task of enforcing the GUN platform with agents, we realized the need to analyze the applicability of different matchmaking methods to agent development tools that are compliant to agent technology specifications such as FIPA [4].

There are many approaches to defining distance between any two entities (attributes, terms) based on their numerical or semantic closeness. For example, Tailor and Tudhope [5] have presented a hypermedia architecture that is supported by a classification schema. Semantic closeness measures have been developed to measure the closeness of terms in a schema

A. Naumenko · S. Nikitin (✉) · V. Terziyan
Industrial Ontologies Group, MIT Department,
University of Jyvaskyla, P.O. Box 35 (Agora),
FIN-40014 Jyvaskyla, Finland
e-mail: senikiti@cc.jyu.fi

A. Naumenko
e-mail: annaumen@cc.jyu.fi

V. Terziyan
e-mail: vagan@it.jyu.fi

which provides a platform for high-level navigation tools and which can provide flexible access tools to a collection of material. Two higher level navigation tools, navigation via media similarity and best-fit generalization, also have been developed. The similarity coefficients are extended in that similarity is judged on the "semantic closeness" of the sets of classification terms that are attached to the media nodes. The similarity coefficient therefore needs to be able to handle sets of classification terms with varying lengths, with non-exact matches of terms, and where the pairing of terms between media nodes may not be immediately obvious.

Brooks reports two experiments that investigated the semantic distance model (SDM) of relevance assessment [6]. In the first experiment, graduate students of mathematics and economics assessed the relevance relationships between bibliographic records and hierarchies of terms composed of classification headings and help-menu terms. The relevance assessments of the classification headings, but not the help-menu terms, exhibited both a semantic distance effect and a semantic direction effect as predicted by the SDM. Topical subject expertise enhanced both these effects. The second experiment investigated whether the poor performance of the help-menu terms was an experimental design artifact reflecting the comparison of terse help terms with verbose classification headings. In the second experiment, the help menu terms were compared to a hierarchy of single-word terms where they exhibited both a semantic distance and semantic direction effect.

Foo et al. [7] propose and define a modification of Sowa's metric on conceptual graphs. The metric is computed by locating the least subtype which subsumes the two given types, and then adding the distance from each given type to the subsuming type.

The distance metric used by Rada et al. [8] represents the conceptual distance between concepts. Rada et al. use only the path length to determine this conceptual distance, with no consideration of node or link characteristics. Distance is measured as the length of the path representing the traversal from the first classification term to the second. The closeness of terms ranges from 1 (identical terms) to 0 (which represents that terms are not semantically close, although it does not mean that they are disjoint in the classification schema).

Instance-based learning techniques typically handle continuous and linear input values well, but often do not handle nominal input attributes appropriately. The Value Difference Metric (VDM) was designed by Wilson and Martinez [9] to find reasonable distance values between nominal attribute values, but it largely ignores continuous attributes, requiring discretization to map continuous values into nominal values. Wilson and Martinez propose new heterogeneous distance functions, called the Heterogeneous Value Difference Metric (HVDM), the Interpolated Value Difference Metric (IVDM), and the Windowed Value Difference Metric (WVDM). These new distance functions are designed to handle applications with nominal attributes, continuous attributes, or both. As was mentioned in the Wilson and Martinez review [9], there are many learning systems that depend upon a good distance function to be successful. A variety of distance functions are available for such uses, including the Minkowsky, Mahalanobis, Camberra, Chebychev, Quadratic, Correlation, and Chi-square distance metrics; the Context-Similarity measure; the Contrast Model; hyperrectangle distance functions; and others.

The problem of service and resource matching (or measuring distance between service query and registered service profiles) is being actively discussed currently as a new challenging task for the next generation of annotation approaches to Web services and Web agents. A significant advantage is expected when using an ontological approach to semantically describe and query services. A matchmaking problem arises when a service is being queried and the query includes the distance measure between the required service description and the one from the service registry. We realized the need to analyze the applicability of different matchmaking methods to agent development tools implemented according to agent technology specifications such as FIPA (Section 2). We also consider three main groups of cases: matchmaking between classes of service profiles in pure taxonomies (Section 3); matchmaking between classes in faceted taxonomies (Section 4); and matchmaking between instances of faceted taxonomies (Section 5). Some samples of recent related work are given in Section 6. We conclude in Section 7.

2. FIPA matchmaking algorithm

One of the crucial components of a multi-agent system is a Registry to provide support for service registering and discovering. According to FIPA specifications [10], the Service Directory service is in charge of providing such functionality within the Agent System. The Service Directory service stores information of a service as an entry of its Service Description. The Directory Facilitator [11] is a reification of the Service Directory Service to provide a yellow page directory service for agents who have services to advertise. The Directory Facilitator operates with the Service Descriptions, which correspond to the structure of Fig. 1. The Service Description consists of the name of the service, its type, its supported interaction protocols, a list of ontologies, a list of content languages, the owner of the service, and a list of additional descriptive properties.

FIPA specifies a concrete matching criteria. The algorithm has to perform syntactic and structural matching based on a service template and a registered service description in the Directory Facilitator. The service template does not match the registered service description if:

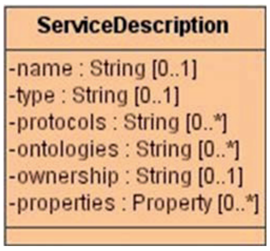
ServiceDescription
-name : String [0..1]
-type : String [0..1]
-protocols : String [0..*]
-ontologies : String [0..*]
-ownership : String [0..1]
-properties : Property [0..*]

Fig. 1 Structure of Service Description

1. Any parameter of the service template does not exist in the registered service description, or,
2. Any parameter of the service template does not match to a corresponding parameter of the registered service.

A parameter of the service template matches a parameter of registered service if both names are equal and their values match.

For the Service Description it means that

- The name, type and ownership parameters match if their values are equal; and
- Protocols, ontologies and properties parameters match if each element of the set of the service template is matched by an element of the set of the registered service.

A possible service description as part of the Agent Description for Registration:

```
(service-description
          :name BusTicketBookingService1
          :type BusTicketBookingService
          :ontologies(set ServiceOntology)
          :properties(set
           (property
              :name Country
              :value Finland)
           (property
              :name typeOfConnection
              :value local)))
```

Service Template for matching:

```
(service-description
          :type BusTicketBookingService
          :ontologies (set ServiceOntology)
          :properties (set
           (property
              :name Country
              :value Finland)))
```

The main disadvantage is that the algorithm performs only syntactic matching and does not utilize even easy-to-implement and obvious possibilities of semantic matching. For instance, the algorithm checks only the syntactic case-insensitive equality of a string value of the type of the service template with type of the registered service. The algorithm takes into account only existing parameters and values in the service template, and does not take into account any information about semantic relation of the parameters nor their values defined in correspondent ontology. Thus the definition area of the algorithm is restricted by the possible combination of optional parameters, user defined properties, and the ranges of their values. As a result, the algorithm provides a fewer number of possible matching services because of the absence of analysis of the correspondent ontology. Last but not least, the algorithm is a binary function with "yes" and "no" answers instead of a more flexible approach of a distance measure.

Different characteristics of multi-agent systems influence the matching algorithms in the sense of what accessible information to utilize during the matching process. For instance, the Directory Facilitator is a registry of instances of services. Agents can register services in arbitrary Directory Facilitators. Instances of the service have a reference to ontology through the type of service parameter. But ontology does not have references to instances of classes of services. The absence of these references means that ontology cannot provide a number of services of some type. But matching algorithms can utilize numbers of instances to measure distances between classes more precisely.

3. Taxonomy-based distance measure

Having a service type in a search query, according to current FIPA implementations, we can compare a query string to service type[1] strings of available services. Such a comparison may lose efficiency in a large and highly distributed environment. However, new services will require additional new service types, and it will be quite complex to bind all services to a finite number of service types in order to save unambiguity and thus allow for search efficiency.

The simplest solution for a dynamic environment is to provide taxonomy of service types. The taxonomy provides its entities with a class-subclass relationship. This relationship is transitive, so if `classA` has subclass `classB` and `classB` has subclass `classC`, then `classC` is a subclass of `classA` also. Such a hierarchy will break the limitation in the number of service types and will save—and even gain—in search efficiency by providing unambiguous search. The simple example of service type taxonomy (see Fig. 2) can demonstrate the benefits of such approach.

The simplest hierarchy structure allows an arbitrary number of subclasses to be created, while saving the semantics of

[1] Service type refers to a class in an ontology.

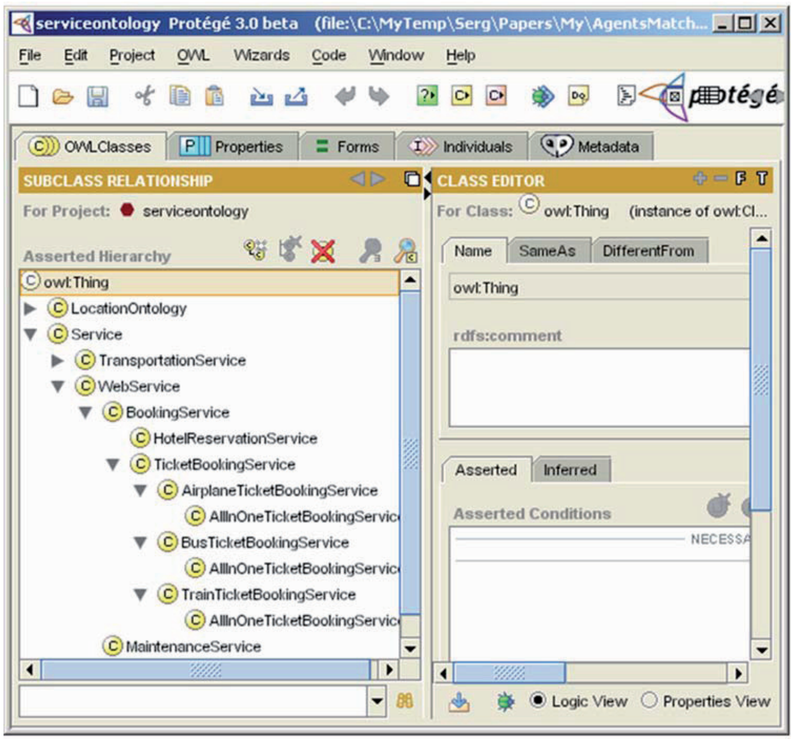
serviceontology Protégé 3.0 beta (file:\C:\MyTemp\Serg\Papers\My\AgentsMatch...
File Edit Project OWL Wizards Code Window Help
OWLClasses Properties Forms Individuals Metadata
SUBCLASS RELATIONSHIP
For Project: serviceontology
Asserted Hierarchy
owl:Thing
LocationOntology
Service
TransportationService
WebService
BookingService
HotelReservationService
TicketBookingService
AirplaneTicketBookingService
AllInOneTicketBookingServic
BusTicketBookingService
AllInOneTicketBookingServic
TrainTicketBookingService
AllInOneTicketBookingServic
MaintenanceService
CLASS EDITOR
For Class: owl:Thing (instance of owl:Cl...
Name SameAs DifferentFrom
owl:Thing
rdfs:comment
Asserted Inferred
Asserted Conditions
NECESSA
Logic View Properties View

Fig. 2 Simple service hierarchy

the upper ones. For example, if we extend `TicketBookingService` class to four more specific subclasses, they all still belong to `TicketBookingService` yet their instances should be returned upon a search request looking for `TicketBookingService`. Of course it is possible to look for only direct instances of a certain class, but this limitation can be used for a more precise search. For example, if we search for only direct instances of `AirplaneTicketBookingService`, we will not receive instances of `AllInOneTicketBookingService` which could provide us with the facility we look for.

Information that an instance of a class is also an instance of all super classes of the class gives us the possibility to implement the simplest reasoning on taxonomy. For example, a search request,

```
(service-description
:type TicketBookingService
:ontologies (set ServiceOntology))
```

will not provide any response in the case when the FIPA matching algorithm is used and there are no registered services with such type in the Directory Facilitator.

But there are several services registered with types of subclasses (`BusTicketBookingService`, `TrainTicketBookingService`, etc.) of `TicketBookingService`. The Directory Facilitator has to return such services in response, as they are also a type of the requested service that is seen in Fig. 3.

It is easy to implement the functionality of the Directory Facilitator to perform reasoning over a taxonomy utilizing the class-subclass relationship among types of services. There are a number of open source libraries providing reasoning API for ontologies encoded in different languages. For instance, JENA is a JAVA library for the implementation of Semantic Web features. It can operate with RDF- and OWL-based ontologies. Concrete implementation of the matching algorithm has to create the JENA model of the ontology.

Then, instead of comparing syntactical equality of names of direct classes of services, JENA gets all the subclasses of a class of the requested service from the model and compares them further using the FIPA algorithm of syntactic match. This source code below allows the forming of a list of subtypes of a type of the requested service:

```
ArrayList subClasses = new ArrayList();
OntModel model = ModelFactory.createOntologyModel();
model.read("http://sample.domain/ServiceOntology.rdf","
RDF/XML");

OntClass requestedServiceClass =
model.getOntClass("http://sample.domain" +
"/ServiceOntology.rdf#TicketBookingService");

ExtendedIterator iterator =
requestedServiceClass.listSubClasses(false);

while ((iterator!=null)&&(iterator.hasNext())){

     OntClass subClass = (OntClass) iterator.next();

subClasses.add(subClass.getLocalName());}
```

There are two possibilities for a deployment of the subclass relation-based matching algorithm. The most appropriate one is to realize it as a part of the Directory Facilitator, as Fig. 4 illustrates.

Such an architectural solution requires changes in the FIPA specifications. But there is no need to change the protocol of request of agents to the Directory Facilitator. Basically the Directory Facilitator embeds the source code mentioned above to get a list of the types of services that are subclasses of the requested service type. Using the list of subtypes, the Directory Facilitator can perform the regular FIPA algorithm for the syntactic matching of type and subtypes of the requested service and for the types of the services registered in the directory. As a result, the Directory Facilitator responds to all services which have the requested type. Figure 5 shows a sequence diagram of the described interaction between the Agent, Directory Facilitator, and Ontology. The main functional characteristics are

– an agent forms one request with a service type,
– the Directory Facilitator requests a list of subclasses for the specified type of a service from the Ontology, then
– the Directory Facilitator performs internal iterations of the syntactic match over a list of requested types of services.

Main nonfunctional properties are

– an agent operates according to the regular protocol , and
– the response of the Directory Facilitator is now semantically full.

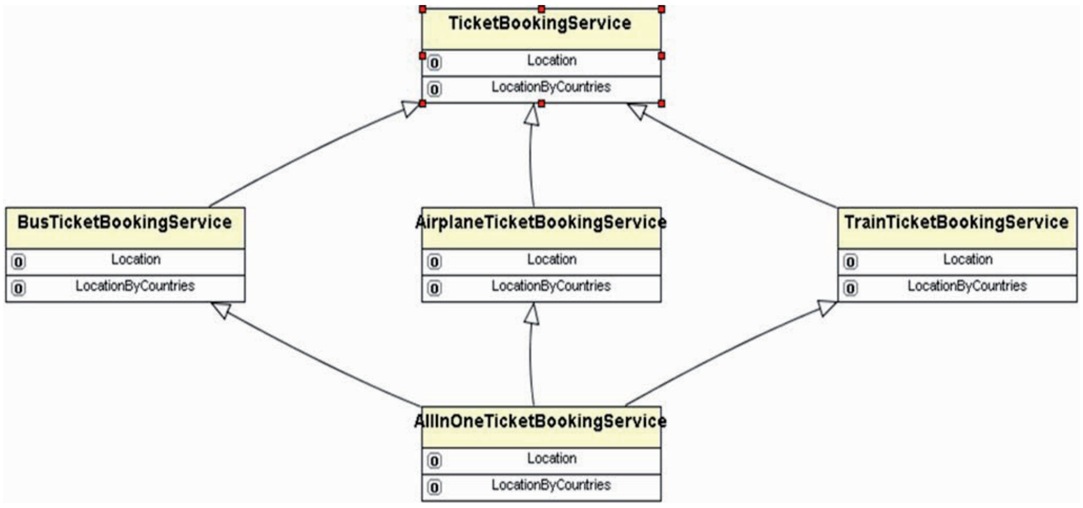

Fig. 3 Subclasses of `TicketBookingService`

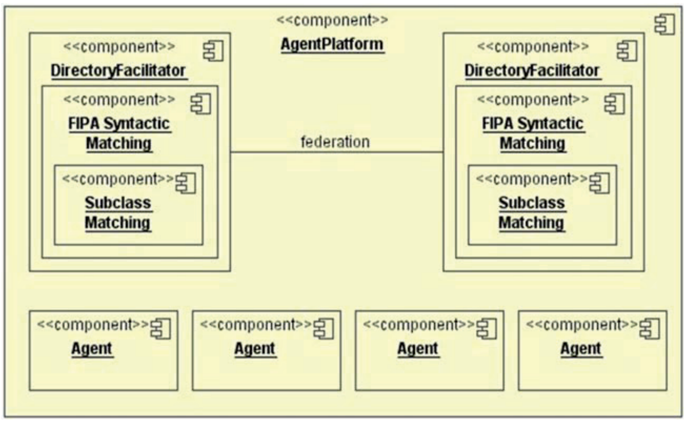

Fig. 4 Component diagram when the subclass matching component belongs to the Directory Facilitator

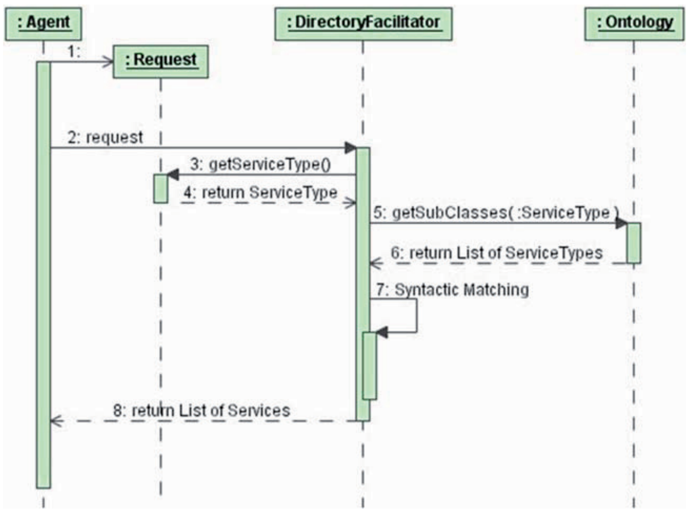

Fig. 5 Sequence diagram of an interoperation of the Agent, Ontology and Directory Facilitator

The solution should be used as a possible elaboration of FIPA specification. Implementing it out of the FIPA specification could cause a misunderstanding and a lack of interoperability of the enhanced Directory Facilitators with the existing FIPA compliant components.

The second approach is to leave the Directory Facilitator as is, while implementing a subclass matching as the Agent's functionality. Figure 6 depicts the architecture of this solution.

There is no need to change the FIPA specifications in this instance. The agents could utilize this functionality with the already existing implementations of the multi-agent systems. Figure 7 illustrates a sequence diagram of an interaction among the Agent, the Ontology and the Directory Facilitator for this architectural solution. The main functional characteristics are

– the agent requests a list of subclasses for the specified type of service from the Ontology,
– the agent iteratively forms requests over the list of requested types of services, then
– the Directory Facilitator provides a syntactic matching service according to the FIPA specification.

The agent collects the responses after the syntactic match. The main nonfunctional properties are

– if needed, the Agent can itself form a semantic matching,
– the Directory Facilitator stays unchanged, and
– inefficient network utilization.

In some cases, it is reasonable to return in a search result not only instances of the class being searched but also instances of classes close to that being searched. For example,

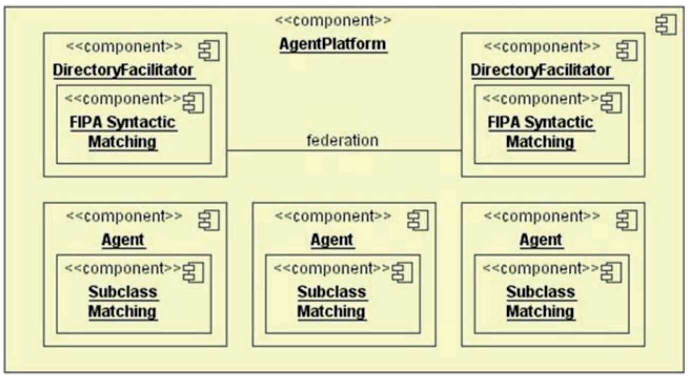

Fig. 6 Component diagram when the subclass matching is part of the Agent functionality

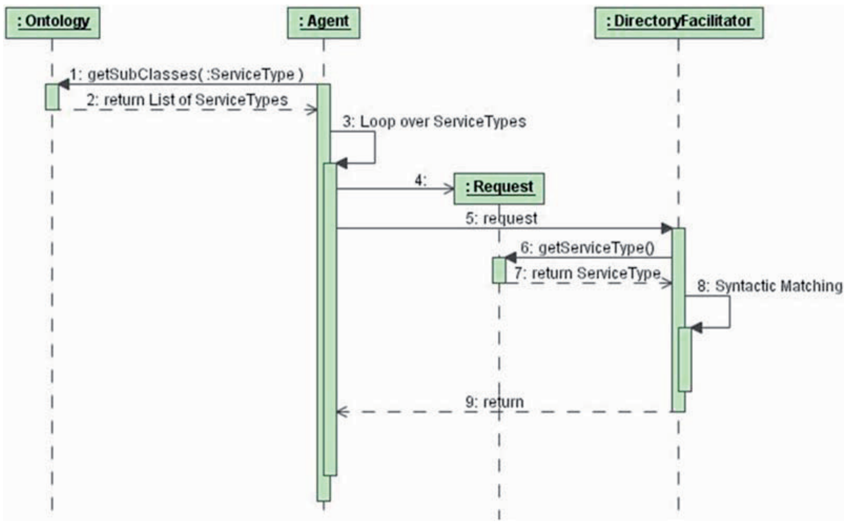

Fig. 7 Sequence diagram of the interoperation of the Agent, Ontology and Directory Facilitator

if we are looking for `BusTicketBookingService` and no instances are available, then, probably, the user wouldn't mind getting instances of a sibling service, `TrainTicketBookingService`, as a result. Considering a general case, we need to provide a distance metric to select the closest classes for response when there are no instances of a requested type.

One of the metrics could reflect generalization of classes. The utilization of this semantic relation depends on the interests of a domain and a task. For a search of some service type, the answer can contain instances of a superclass in the case where there are no instances of the requested service class. It means, for example, that on a request for the `BusTicketBookingService`, the user may get instances of `TicketBookingService`, which are all instances of the other subclasses of `TicketBookingService`. If for some domain or task such functionality is reasonable, then the distance between the classes is 0 in the case of a subclass relation and 1 for each instance of a superclass relation. For instance, Fig. 8 shows that `TicketBookingService` has the distance of 0 to all its subclasses, the distance of 1 to the direct superclass `BookingService`, the distance of 2 to `WebService`, and the distance of 3 to Service. According to the distance measure described above, the implementation of a matching algorithm based on the subclass relation is a particular case of the distance measure based on the subclass and superclass relations.

The implementation of the matching algorithm based on the generalization of classes requires changes of the FIPA specifications. FIPA defines an object of the search constraints to limit the function of searching within a directory. The object consists of three parameters: `max-depth`, the

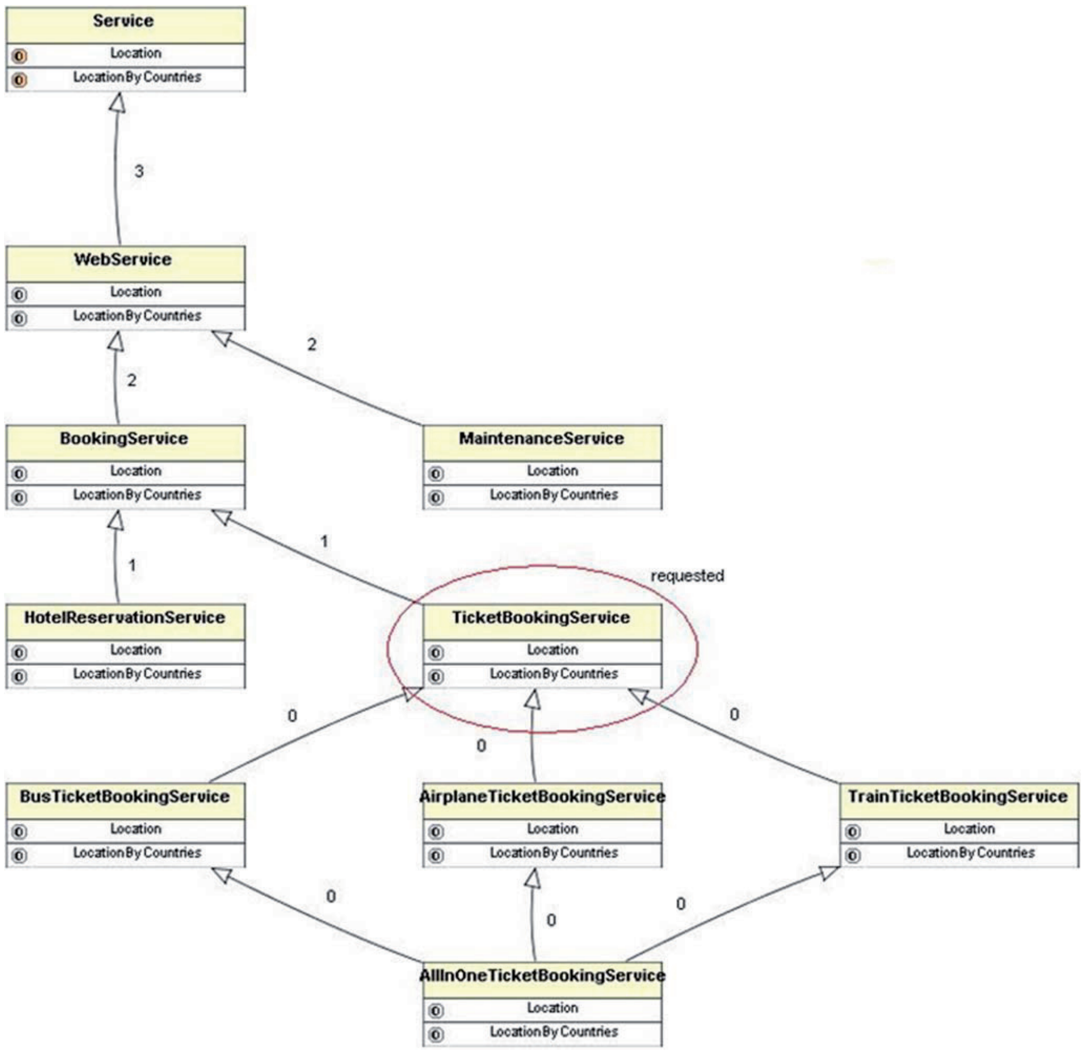

Fig. 8 Distances between `TicketBookingService` and other services

maximum depth of propagation of the search to the federated directories; `max-results`, the maximum number of results to return for the search; and `search-id`, a globally unique identifier for a search. An agent has to pass an additional parameter, `max-distance`, to support the implementation of the generalization-based matching algorithm. `Max-distance` is a positive integer, including zero, that defines an expansion of the search over the ontology. The value 0 requires services which are of the type of the requested class and its subclasses. The value 1 includes the direct superclasses and their subclasses in consideration. Generally, the value of the `max-distance` defines the level of the superclass to which search could be extended if there are no services on lower levels.

The federation of directories adds a complexity to the implementation of this algorithm because the directory has to meet the requirements of the `max-results` parameter of the search constraints. One of the possible solutions is to collect the matching services of the farthest classes in respect to `max-distance` parameter. Then the search is propagated with respect to the `max-depth` parameter while integrating the resulting sets of services from the previous

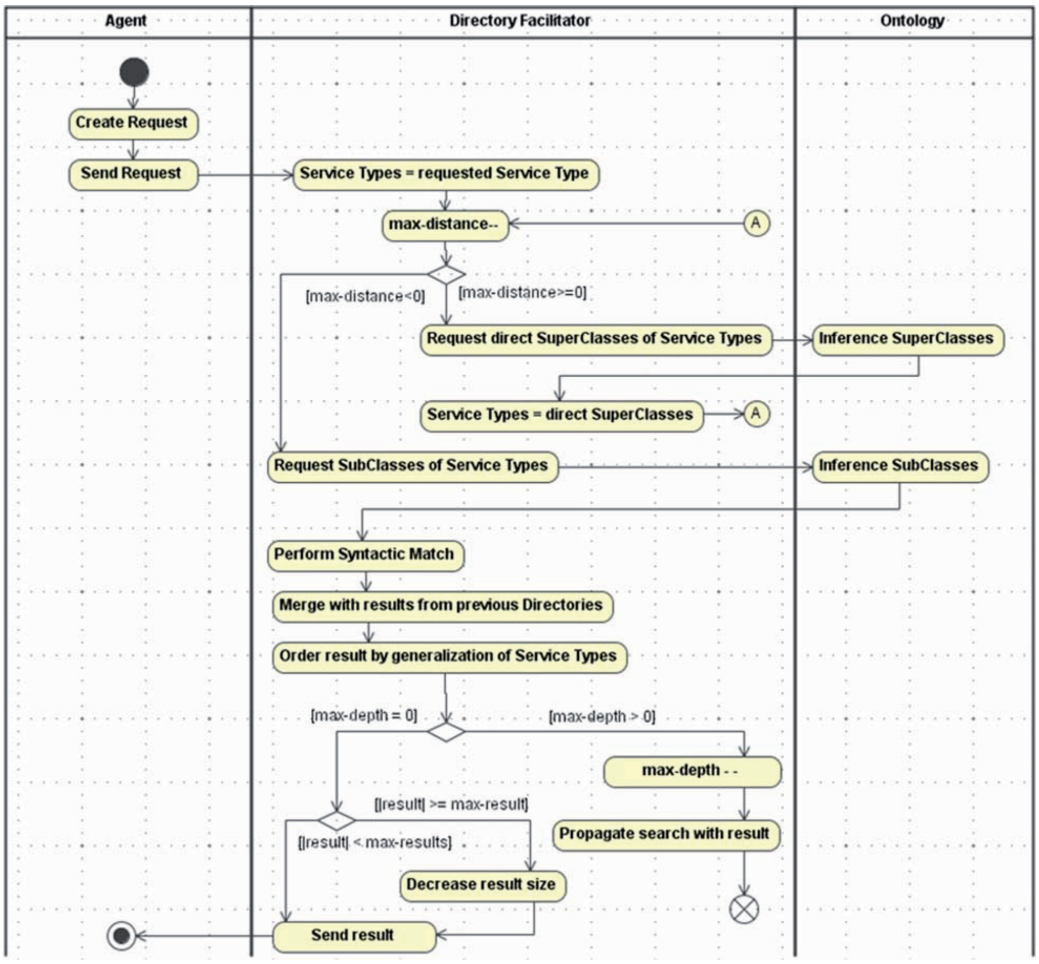

Fig. 9 Activity diagram of a matching algorithm based on generalization distance

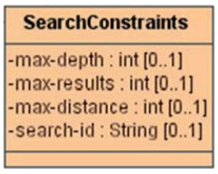
SearchConstraints
-max-depth : int [0..1]
-max-results : int [0..1]
-max-distance : int [0..1]
-search-id : String [0..1]

Fig. 10 Extended structure of the Search Constraints

directories. The response consists of a `max-result` or a smaller number of the most specialized services. Figure 9 illustrates the algorithm described above.

Figure 10 shows the extended structure of the object of the search constraints. Architectural considerations are the same as for the matching algorithm based on the subclass relation.

The Object Match algorithm [12] is more sophisticated. It calculates the similarity of two concepts based on a hierarchy of concepts. The algorithm uses the concept of upwards cotopy (UC) defined as follows:

$$UC(O_j, H) = \{O_j \mid H(O_i, O_j) \cup O_j = O_i\} \qquad (1)$$

where taxonomy is given by an irreflexive, acyclic, transitive relation class-subclass H and O is a class in a taxonomy.

The intersection of upwards cotopies of two classes is

$$I(O_1, O_2, H) = UC(O_1, H) \cap UC(O_2, H) \qquad (2)$$

The union of upwards cotopies of two classes is

$$U(O_1, O_2, H) = UC(O_1, H) \cup UC(O_2, H) \tag{3}$$

And according to the Object Match algorithm, the similarity of two classes equals to:

$$S(O_1, O_2, H) = \frac{|I(O_1, O_2, H)|}{|U(O_1, O_2, H)|} \tag{4}$$

The algorithm reflects the fact that more specialized neighbors are closer to each other than the more general ones. Basically, the algorithm takes into account the quantity of common and different parents with a sensitivity to the depth of classes within the taxonomy.

This algorithm could be enhanced by taking into account the width as well as the depth of the taxonomy. Similarity is defined as follows:

$$S_e(O_1, O_2, H) = \frac{\sum_{O_i \in I(O_1, O_2, H)} |\{O_j | H'(O_i, O_j) \cup O_i = O_j\}|}{\sum_{O_i \in U(O_1, O_2, H)} |\{O_j | H'(O_i, O_j) \cup O_i = O_j\}|} \tag{5}$$

where H' is a relation class—direct subclass.

Let us consider an example of the abilities of the algorithm to reflect the depth and width of taxonomy. Figure 11 shows three hierarchies of services.

We have to calculate the upwards cotopies in order to measure the distance between D and E in the first hierarchy H_1, using formula 1:

$$UC(D, H_1) = \{D, B, A\} \tag{6}$$

$$UC(E, H_1) = \{E, B, A\} \tag{7}$$

The intersection and union of the cotopies are calculated using formulas 2 and 3:

$$I(D, E, H_1) = \{B, A\} \tag{8}$$

$$U(D, E, H_1) = \{D, E, B, A\} \tag{9}$$

The similarity according to formula 4 equals

$$S(D, E, H_1) = \frac{|\{B, A\}|}{|\{D, E, B, A\}|} = \frac{1}{2} \tag{10}$$

The enhanced similarity by formula 5 is equal to

$$S_e(D, E, H_1) = \frac{|\{D, E, F, B\}| + |\{B, C, A\}|}{|\{D, E, F, B\}| + |\{B, C, A\}| + |\{D\}| + |\{E\}|} = \frac{7}{9} \tag{11}$$

The second hierarchy H_2 extends the first hierarchy H_1 by introducing a new subclass of B, thus raising the width of the hierarchy. The upwards cotopies for D and E in such a case are the same as in the first hierarchy. The intersection and union of cotopies are also the same. The value of similarity of D and E by formula 4 stays unchanged. The enhanced similarity shows that D and E are semantically closer in the second hierarchy:

$$S_e(D, E, H_2) = \frac{|\{G, D, E, F, B\}|}{|\{G, D, E, F, B\}| + |\{B, C, A\}| + |\{D\}| + |\{E\}|} + \frac{|\{B, C, A\}|}{|\{G, D, E, F, B\}| + |\{B, C, A\}| + |\{D\}| + |\{E\}|} = \frac{8}{10} \tag{12}$$

The calculation of the similarity of D and E for the third hierarchy H_3 is analogical to the calculation for the first hierarchy. The similarities by formula 4 and the enhanced formula 5 are equal to:

$$S(D, E, H_3) = \frac{|\{B, A, G\}|}{|\{D, E, B, A, G\}|} = \frac{3}{5} \tag{13}$$

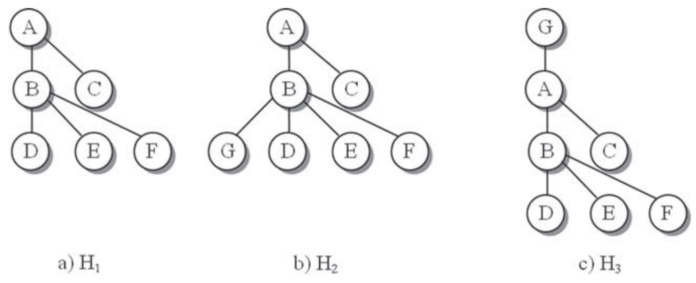

Fig. 11 Example on Hierarchies

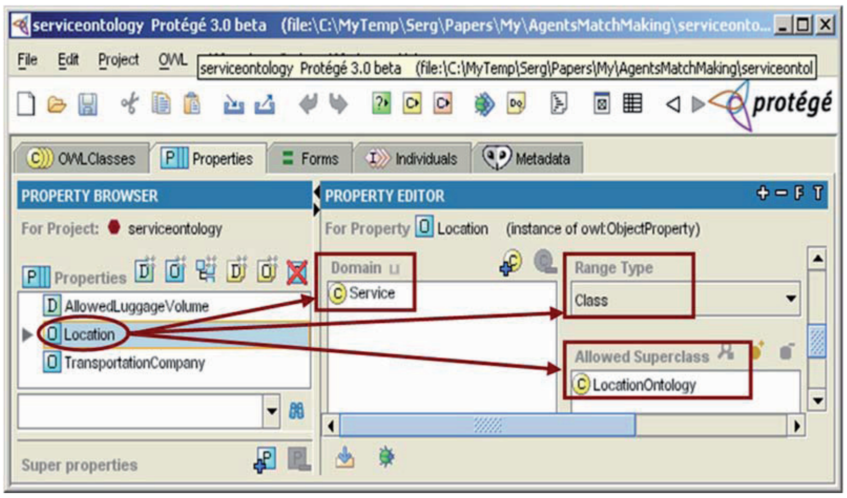

Fig. 12 Property domain and range definition in Protégé

$$S_e(D, E, H_3) = \frac{|\{D, E, F, B\}| + |\{B, C, A\}| + |\{A, G\}|}{|\{D, E, F, B\}| + |\{B, C, A\}| + |\{A, G\}| + |\{D\}| + |\{E\}|} = \frac{9}{11} \qquad (14)$$

As we can see, D and E are semantically closer by both formulas 4 and 5 in the third hierarchy as compared to the first one.

4. Distance measure for ontology with facets

Faceted Classification [13] is a sophisticated alternative to the traditional classification schemes and modern web directories, which put one item in only one place. Faceted classifications are based on the Colon Classification scheme of Indian library scientist S.R. Ranganathan developed in the 1930s. Ranganathan created a set of properties or characteristics or attributes of any subject, ideally mutually exclusive (orthogonal) and exhaustive (complete), which means that any object being classified could be assigned one of these descriptions, which he called a facet. The outstanding advantage is that a hierarchy can be built starting with the facet considered as the most important. A facet-based classification has much more extended capabilities in comparison to a taxonomy-based one. In ontological modeling it is quite usual for a certain hierarchy to be selected as a backbone for the whole Ontology, which is then powered by additional properties assigned to the classes. The main hierarchy consists of the objects representing the goal of a description. For example, if we consider services as our description goal, then we can organize them in the main class-subclass relationship by their service type and then augment with additional properties, such as location, type of payment accepted, etc.

For example, we can add the property `Location` to the `BookingService` class (Fig. 12).

We then provide the property `Location` with `domain` and `range` characteristics. The `domain` defines the set of classes allowed to have this property and the `range` defines the allowed values for the property.

In our case, we set the `domain` value to the `Service` class. This means that any instance of a `Service` class or its subclasses may have the property `Location`. As a range value type we set `Class`, and as an allowed superclass for value—`LocationOntology`. In this way we define a controlled vocabulary for the property values. A *controlled vocabulary*, or managed vocabulary, is an attempt to limit the number of terms that will be admitted into a discourse in order to improve communication. However, the applicability of a controlled vocabulary in the quickly changing environments of P2P is still a matter of discussion and research.

When comparing classes with properties, the distance can be measured based on a number of discriminate properties. The more identical properties the classes have, the more similar classes are. The simple distance measure for two classes, A and B with facets n_A and $n_{B,}$ can be calculated by formula:

$$D = \frac{|n_A \cap n_B|}{|n_A \cup n_B|} \qquad (15)$$

So we divide the number of identical facets into the number of facets used for the description of both classes (identical facets of both classes are counted only once).

For example, if class A has 5 facets $\{f1, f2, f3, f4, f5\}$ and class B has 7 facets $\{f1, f2, f3, f6, f7, f8, f9\}$, then distance can be calculated as:

$$\begin{aligned} |n_A \cap n_B| &= |\{f_1, f_2, f_3\}| = 3 \\ |n_A \cup n_B| &= |\{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\}| = 9 \\ D &= \frac{3}{9} = \frac{1}{3} \end{aligned} \tag{16}$$

5. Distance measure between instances in ontology

Faceted classification schemes may vary. The limited set of allowed values (controlled vocabulary) for certain facet provides the ability to measure distance between two instances having the same facet. If the allowed values are arranged in taxonomy, it extends the distance measure precision. However, property (attribute) values can be from non-controlled vocabulary, which means that only a Boolean match can be applied for returning 1 if values are equal and 0 if not. If the values are numerical then the distance can be measured using a standard measure (e.g. Euclidean distance) however values obtained should be normalized.

When an instance of a certain class is requested and can not be found (e.g. we ask for the implementation of an online train ticket service in Finland but there is no service currently running), then it makes sense to return a bus ticket service located also in Finland. Hence, we need to deal with instances and assigned property values.

In our opinion, similarity (or distance) between instances can be measured only using common properties and, thus, their comparable values. However, the closest class to class of instance should be found first. After the closest class is found, its instances can be taken and the distance can be measured based on different metrics.

Assume that we have two interpreted profiles with the same set of attributes, which have numerical or nominal values. Assume also that the first profile is taken from the "service requests" database and the second one from the "service offerings" database. The distance between these two profiles can be measured according to [14, 15] as follows:

$$E(X, Y) = \sqrt{\sum_{\forall i, x_i \in X, y_i \in Y} \omega_i \cdot d(x_i, y_i)^2} \tag{17}$$

where X and Y are two vectors of the values of the attributes of the two profiles. The component distance $d(x_i, y_i)$ for every attribute is normalized by the range of the previously known values of the attribute so that it is mostly within the range [0, 1], and weighted by weights ω_i according to the importance of the attribute. The weight ω_i may be the probability that a client, whom a request (interpreted profile) belongs to, will not be satisfied by an offering (interpreted profile of the same structure) if the i-th attribute of these profiles is not taken into account.

The Heterogeneous Euclidean-Overlap Metric is used for both nominal and numerical features. This function defines the component distance between two values of an attribute as:

$$d(x_i, y_i) = \begin{cases} \text{if } i\text{-th attribute is nominal } - \begin{cases} 0, & \text{if } x_i = y_i \\ 1, & \text{otherwise} \end{cases} \\ \text{else} : \dfrac{|x_i - y_i|}{range_i} \end{cases} \tag{18}$$

where $range_i$ is the range of the attribute i.

If the attribute is nominal but has values from the controlled vocabulary organized in taxonomy, then the distance between the two attribute values can be measured using distance measure techniques applicable to taxonomy, as described in Section 3. So the distance can be defined as:

$$d(x_i, y_i) = \begin{cases} \text{if } i\text{-th attribute is nominal} - \text{distance} \\ \quad \text{between ontology concept } x_i \text{ and } y_i \\ \text{else} : \dfrac{|x_i - y_i|}{range_i} \end{cases} \tag{19}$$

The Interpolated Value Difference Metric defines the following component distance between the two values of an attribute:

$$d(x_i, y_i) = \begin{cases} \sqrt{\displaystyle\sum_{j=1}^{k} |P(j \mid i \in [x_i, x_i + \Delta]) - P(j \mid i \in [y_i, y_i + \Delta])|^2}, \\ \quad \text{if } i \text{ numerical (continuous) attribute} \\ \sqrt{\displaystyle\sum_{c_i=1}^{k} |P(j \mid i = x_i) - P(j \mid i = y_i)|^2}, \quad \text{otherwise,} \end{cases} \tag{20}$$

where k is the number of classes of profiles; $P(j \mid i = x_i)$ is the conditional probability that a profile belongs to class j if its attribute i has the value x_i, and $P(j \mid i \in [x_i + \Delta])$ is the interpolated conditional probability that a profile belongs to class j if its discreted attribute i has the value x_i, and Δ is the discretization step.

Table 1 Services separated by facet values

Object	Quantity
Web Services Total	10
`PlaneTicketBookingService`	3
`TrainTicketBookingService`	2
`BusTicketBookingService`	5
Services accepting Web money	4
Services not accepting Web money	6
`PlaneTicketBookingService` *accepting Web money*	1
`TrainTicketBookingService` *accepting Web money*	1
`BusTicketBookingService` *accepting Web money*	2
`PlaneTicketBookingService` *not accepting Web money*	2
`TrainTicketBookingService` *not accepting Web money*	1
`BusTicketBookingService` *not accepting Web money*	3

A more simple interpretation of the previous formula might be:

$$d(x_i, y_i) = \begin{cases} \sqrt{\sum_{c=1}^{C} [P(c|x_i) - P(c|y_i)]^2}, \\ \text{if } i\text{-th attribute is nominal;} \\ \frac{|x_i - y_i|}{range_i}, \quad \text{if } i\text{-th attribute is numerical.} \end{cases} \tag{21}$$

A probabilistic approach requires prior knowledge of the total number of instances of a certain class but, in the P2P environment, it may not always be easy to obtain statistics about instances and range values of numerical attributes. But by having this data the calculations gain in accuracy. For example, we have a preference to find a train ticket booking service that accepts Web money (payment via the Internet) and has a service load of not more than 60%. The importance of service load is estimated as 0.2 and importance of transport type (train, bus or plane) is 0.8 respectively. To calculate the distance to the closest service profile we need the following data:

– The total number of ticket booking service instances and the number of service instances of each class;
– The total number of service instances accepting Web money and the number of service instances of each class that accept Web money;
– the service load value.

For example, we have 10 ticket booking service instances, among them:

– three instances of `PlaneTicketBookingService`
– two instances of `TrainTicketBookingService`
– five instances of `BusTicketBookingService`

Among our 10 service instances only 4 accept Web money.
The four instances accepting Web money comprise a `PlaneTicketBooking Service`, a `TrainTick-etBookingService` and two instances of `BusTicket BookingService`. The six remaining instances are: two instances of `PlaneTicketBookingService`, one `TrainTicketBookingService` and three instances of `BusTicketBookingService`. The summary is provided in Table 1.

Let's consider two service profiles (Table 2).

In order to calculate the distance from our preference to the Service Profiles, we need conditional probability values calculated according to Eq. (12) (see Table 3).

Table 2 Service profiles attribute values

	Service Profile 1	Service Profile 2
ServiceType	`TrainTicketBookingService`	`BusTicketBookingService`
Load	80%	20%

Table 3 Conditional probability values for the calculation of the distance between `TrainTicket-BookingService` and `BusTicketBooking-Service` values

Probability	Value
P(*Accepts Web money* | *ServiceType* = `TrainTicketBookingService`)	1/2
P(*Accepts Web money* | *ServiceType* = `BusTicketBookingService`)	2/5
P(*Not accepts Web money* | *ServiceType* = `TrainTicketBookingService`)	1/2
P(*Not accepts Web money* | *ServiceType* = `BusTicketBookingService`)	3/5

So the distance can be calculated as:

$$d(\text{“}Train\text{”}, \text{“}Bus\text{”}) = \sqrt{\begin{pmatrix} (P(AcceptsWM \mid ServiceType = Train) \\ - P(AcceptsWM \mid ServiceType = Bus))^2 \\ + (P(NotAccWM \mid ServiceType = Train) \\ - P(NotAccWM \mid ServiceType = Bus))^2 \end{pmatrix}} \tag{22}$$

Substituting the values in Eq. (22), we have:

$$d(\text{“}Train\text{”}, \text{“}Bus\text{”}) = \sqrt{\left(\frac{1}{2} - \frac{2}{5}\right)^2 + \left(\frac{1}{2} - \frac{3}{5}\right)^2} \approx 0,141 \tag{23}$$

Now we can calculate the distance of our preferences to Service Profiles. Here we have to mention that the distance of attribute having numerical value as a marginal value should be calculated with an additional condition. In our case, we assume that “60% load” parameter should be *not more* than 60%, hence the distance can be calculated as:

$$d(x_i, y_i) = \begin{cases} 0, & \text{if } x_i \le y_i \\ \dfrac{|x_i - y_i|}{range_i}, & \text{otherwise,} \end{cases} \tag{24}$$
$$\text{where } range_i = \max(y_i) - y_i$$

The maximum range value for the Service Load attribute is 100 because its value is a percentage.

Now, the distance to the Service Profiles is calculated using Eq. (17).

$$D(UserP, SP1) = \sqrt{0.8 \cdot 0 + 0.2 \cdot \left(\frac{80 - 60}{100 - 60}\right)^2} = 0.1$$
$$D(UserP, SP2) = \sqrt{0.8 \cdot (0.141)^2 + 0.2 \cdot 0} \approx 0.126 \tag{25}$$

where $D(UserP, SP1)$ is a distance between User Preferences and Service Profile 1 and $D(UserP, SP2)$ is a distance between Preferences and Service Profile 2 respectively.

As we can notice from the equations above, it is possible to vary the weights so that the initially preferred `TrainTicketBookingService` might be farther from the `BusTicketBookingService` because of undesirable attribute values.

6. Related work

The application of semantic technology in different distributed environments is an open issue for many research projects. Peer service discovery and matching, based on semantic profiles described in [16], provides an ontology-based matching and a distance measure based on the shortest path between ontology nodes. However, the full power of the ontology is not used and more attention should be paid to overall system architecture and interoperability of nodes.

Another recent activity [17], aimed at service discovery within a grid environment, proposes a similarity metric for measuring the distance between a service request and service descriptions available in the registry. The metric is based on a similarity function which is a weighted sum of matching attributes, description and metadata. The weights are calculated as probabilistic functions.

A Semantic Similarity Measure for Semantic Web Services is proposed in [18]. The formula for semantic similarity is defined as follows: $sim(a, b) = \frac{f_{common}(a, b)}{f_{desc}(a, b)}$, where f_{common} is the common function measuring the information value of the description that is shared between a and b, and f_{desc} is the description function giving the value of the total information content of a and b. But the key feature is how the functions f_{common} and f_{desc} are calculated. Deep analysis is done towards the formation of the description sets. The authors use OWL Lite as a descriptive language and give examples of the applicability on OWL-S descriptions.

7. Conclusions

In this paper, we tried to analyze the applicability of the ontology-based models for the improvement of the searching capabilities in Agent Systems. Firstly, we have demonstrated the drawbacks of the matching algorithm of the Directory Facilitator, compliant with the FIPA specification [11]. The algorithm responds to an incorrect set of services from the ontological point of view because of an instance of a class is also an instance of all superclasses of this class. Secondly, we demonstrated through the examples that matching algorithms based on a distance or similarity measure are more flexible and appropriate in the task of a services search because they provide responses even if an exact match does not exist. With a matching algorithm based on distance measure, there is a possibility for a user to prepare and efficiently execute requests based on uncertain or incomplete information.

The main conclusion is that the Agent, Grid Services, and Web Service technologies can be effectively integrated with each other. However such integration requires correct ontology-based matching tools, which can be considerably improved by similarity measure methods. Analysis shows that some of the algorithms can be easily implemented without radical changes within the existing tools and standards while giving more sophisticated results of matching.

Although an elaboration of the existing standards requires deeper analysis of the quality characteristics of the distance measure-based matching algorithms, we consider these changes as inevitable.

Acknowledgment This research has been supported by the "Proactive Self-Maintained Resources in Semantic Web" (SmartResource) project, funded by TEKES and the industrial consortium of Metso Automation, TeliaSonera, and TietoEnator.

References

1. Kaikova H, Khriyenko O, Kononenko O, Terziyan V, Zharko A (2004) Proactive self-maintained resources in semantic web. Eastern-European J Enter Technol, 2(1):4–16, ISSN: 1729-3774
2. SmartResource project, http://www.cs.jyu.fi/ai/OntoGroup/Smart–Resource_ details.htm
3. Adaptive Services Grid, Integrated project supported by the European Commission, http://asg-platform.org/
4. FIPA, Foundation for Intellegent Physical Agents, http://www.fipa.org/
5. Tailor C, Tudhope D (1996) Semantic closeness and classification schema based hypermedia access. In: Proceedings of the 3-rd International Conference on Electronic Library and Visual Information Research (ELVIRA'96), Milton, Keynes
6. Brooks T (1995) Topical subject expertise and the semantic distance model of relevance assessment. J Doc, 51(4):370–387
7. Foo N, Garner B, Rao A, Tsui E (1992) Semantic distance in conceptual graphs. In: Gerhotz L (ed) Current directions in conceptual structure research. Ellis Horwood, pp 149–154
8. Rada R, Mili H, Bicknell E, Blettner M (1989) Development and application of a metric on semantic nets. IEEE Tran Sys, Man, and Cybernetics 19(1):17–30
9. Wilson D, Martinez T (1997) Improved heterogeneous distance functions. J Art Intell Rese 6:1–34
10. FIPA Abstract Architecture Specification, http://fipa.org/specs/fipa00001/
11. FIPA Agent Management Specification, http://fipa.org/specs/fipa00023/
12. Stojanovic N, Meadche A, Staab S, Studer R, Sure Y (2001) SEAL: A framework for developing semantic PortALs. In: Proceedings of the International Conference on Knowledge Capture, Victoria, British Columbia, Canada, ACM Press. pp 155–162
13. Glossary of Content Management Professionals, http:www. cms-glossary.com/
14. Cost S, Salzberg S (1993) A weighted nearest neighbor algorithm for learning with symbolic features. Mach Learn 10(1):57–78
15. Puuronen S, Tsymbal A, Terziyan V (2000) Distance functions in dynamic integration of data mining techniques. In: Dasarathy BV (ed) data mining and knowledge discovery: Theory, tools and technology II, Proceedings of SPIE, vol. 4057, The Society of Photo-Optical Instrumentation Engineers, USA, pp 22–32
16. Haase P, Agarwal S, Sure Y (2004) Service-oriented semantic peer-to-peer systems, Lecture Notes in Computer Science, vol. 3307, pp 46–57
17. Ludwig SA, Reyhani SMS (2005) Semantic approach to service discovery in a grid environment. J Web Semant, 3(4) Elsevier
18. Hau J, Lee W, Darlington J (2005) A Semantic Similarity Measure for Semantic Web Services, Web Service Semantics Workshop

Vagan Terziyan Professor in Software Engineering since 1994 and the Head of the Artificial Intelligence Department since 1997 in Kharkiv National University of Radioelectronics (Ukraine). Currently he is working in Agora Center (University of Jyvaskyla, Finland) as a project leader of the SmartResource TEKES Project and Head of Industrial Ontologies Group. He is a member of IFIP WG 12.5 ("Artificial Intelligence Applications"), PC Chair of IFIP International Conference on Industrial Applications of Semantic Web. His research and teaching profile is design of Intelligent Web Applications, which utilise and integrate emerging Knowledge-, Agent-, Machine-Learning- and Semantic Web- Based technologies and tools. For more details please refer to his homepage at http://www.cs.jyu.fi/ai/vagan/

Sergiy Nikitin (1982) is currently working as a researcher in Agora Center (University of Jyväskylä, Finland) on the SmartResource TEKES project. Sergiy graduated form the Kharkiv National University of Radioelectronics (KNURE), Ukraine with the Engineer's degree in 2004. At the end of the same year he obtained a Master of Science degree from the University of Jyväskylä, Finland. In 2005 he entered a Ph.D.-program at the Faculty of Information Technology of the University of Jyväskylä. Among other research activities Sergiy has taken part in Adaptive Services Grid (ASG) and SCOMA projects.

His research interests include semantic web services, semantic configuration, agent technology and ontology engineering areas. Sergiy Nikitin has been a member of Industrial Ontologies Group since 2004. For more details please refer to his homepage at http://www.cc.jyu.fi/ senikiti

Anton Naumenko (1980) is a researcher at the Department of Mathematical Information Technology, University of Jyväskylä, Finland (JYU). He has Master of Science degrees in Information Technology from the JYU and in Computer Science from the Kharkov National University of Radioelectronics, Ukraine (KhNURE). He also earned a Bachelor of Economics degree from the KhNURE. In 2004, he has started his postgraduate education in the Faculty of Information Technology, JYU. Anton's professional career started in the KhNURE with a position of technician (2000) in a research project, where he became a researcher and leaded database designers and software developers (2001–2002). He was a leader of a software development team and a researcher in the SmartResource project (2004). After that he participated in the Adaptive Services Grid (2005) and Mobile Design Patterns and Architectures (2005–2006) research projects. His expertise and interests consist of Semantic Web, Agent Technologies, Object-Oriented Design, and Access Control. For more details please visit his homepage at www.cc.jyu.fi/ annaumen.

III

DATA INTEGRATION SOLUTION FOR PAPER INDUSTRY - A SEMANTIC STORING, BROWSING AND ANNOTATION MECHANISM FOR ONLINE FAULT DATA

by

Sergiy Nikitin, Vagan Terziyan and Jouni Pyötsiä 2007

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.

DATA INTEGRATION SOLUTION FOR PAPER INDUSTRY
A Semantic Storing Browsing and Annotation Mechanism for Online Fault Data

Sergiy Nikitin, Vagan Terziyan
Agora Center, University of Jyväskylä, Mattilanniemi 1, Jyväskylä, Finland
senikiti@cc.jyu.fi, vagan@it.jyu.fi

Jouni Pyötsiä
Metso Automation
Jouni.Pyotsia@metso.com

Keywords: Semantic Web, Semantic Storage, Information Integration, Paper Industry.

Abstract: A lot of IT solutions exist for simplification and time saving of industrial experts' activities. However, due to large diversity of tools and case-by-case software development strategy, big industrial companies are looking for an efficient and viable information integration solution. The companies have realized the need for an integrated environment, where information is ready for extraction and sophisticated querying. We present here a semantic web-based solution for logging and annotating online fault data, which is designed, and implemented for a particular business case of a leading paper machinery maintenance and automation company.

1 INTRODUCTION

Rapid changes and discontinuities in the 21st century business environment will challenge companies with the growing demand for competitive advantages within their business solutions. To ensure high flexibility, sustainable growth and profitability, companies have to search for new innovative approaches to products and services development. New innovative business solutions call for strong integration of automation technology, information and communication technology (ICT), and business processes. At the same time, embedded intelligence in different machines and systems gives new possibilities for automated business process operation over the network throughout the machines and systems life cycles.

The new emerging remote service solutions imply that products transform into life cycle services and these services, in turn, transform into customers' service processes. Business messages coming from intelligent machines and systems drive these processes, utilizing embedded intelligence and ICT solutions.

In the future, a variety of collaborative resources, like intelligent machines, systems and experts, will create a huge amount of new information during the life cycles of machines and systems. Message flow management and compression to on-line knowledge are already a demanding issue for the logging of product-related activities. On the other hand, optimization requirements demand more effective knowledge utilization and the speeding up of network-based learning in the process of collaboration between different resources.

Industry challenges the IT-sector with the new requirements that are dictated by the need to offer essentially new services to customers in order to be competitive in the market. These requirements may become hard to meet using conventional tools and approaches. The growth in the information volumes we want to store and process by integrating data from different sources leads to an unprecedented level of complexity. Modern Enterprise Resource Planning (ERP) systems are trying to provide integrated solutions for large companies. However the installation and adjustment of such systems may take a half a year, involving hundreds of consultants and subcontractors.

The current trend towards more open service-based computing environments is a new approach to componentization and components distribution. Service-oriented architecture aims to achieve a new level of reusability and business process flexibility; however, to ensure the interoperability between the components we need a common “glue” that would adjust the semantics of the data being exchanged. The Semantic Web technology (Berners-Lee, T., et al., 2001) introduces a set of standards and languages for representation of a domain model with the explicit semantics. The main instrument of domain model construction is ontology, which allows for domain data representation in a formalized and unified way.

In this paper we present a solution that utilizes Semantic Web technology to provide a tool for online maintenance data browsing, analysis and annotation. We utilize experience obtained in the SmartResource project (SmartResource, 2006) and apply the General Adaptation Framework (Kaykova et al., 2005) to align data with the domain ontology. The paper is organized as follows: In the next section we describe the paper machinery ICT infrastructure, and Section 3 presents the solution we have developed using Semantic Web tools and standards. We end with conclusions and future work.

2 IT INFRASTRUCTURE IN PAPER INDUSTRY

Metso Corporation is a global supplier of process industry machinery and systems as well as know-how and aftermarket services. The corporation's core businesses are fiber and paper technology, rock and minerals processing, and automation and control technology. Metso's strategy is based on an in-depth knowledge of its customers' core processes, close integration of automation and ICT, and a large installed base of machines and equipment. Metso's goal is to transform into a long-term partner for its customers. Based on the remote service infrastructure, it develops solutions and services to improve efficiency, usability and quality of customers' production processes throughout their entire life cycles.

2.1 Remote Service Infrastructure

Metso's remote service infrastructure consists of a service provider's Central Hub and several customers' Site Hubs, which are integrated over the network (see Figure 1).

The key issues in a Site Hub solution are: open standards, information security, reliability, connectivity and manageability.

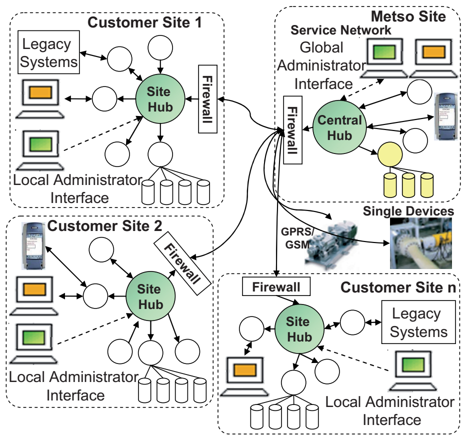

Figure 1: Site Hub network architecture.

Message Center is the main component of the Site Hub. It checks the validity of messages and routes them to the correct receivers. Messaging in Site Hub is based on Web Services technology.

Hub-based integrated infrastructure combined with secure connectivity allows easy incorporation of new business logic on both customer and Metso sites. A messaging mechanism between customers and Metso provides a very flexible medium for information exchange and new service provisioning.

3 LOGGING AND ANNOTATION OF MAINTENANCE DATA

The main purpose of the system we present here is to store alarm data, generated by paper machine monitoring systems. When an alarm happens, a SOAP/XML message (SOAP, 2003) is generated and sent to the Site Hub, which then forwards it to the Central Hub. We have established a message flow from the Central Hub to the computer at the university network, where messages are processed by our system.

3.1 Architecture of the System

The system can be divided into two main subcomponents – *Message Handler* and *Message Browser* (see Figure 2).

Message Handler receives and processes SOAP/XML messages from customers. It invokes the *Adapter* to transform the XML content into an

RDF-graph (RDF, 2004) object and store it in the Sesame RDF storage (Sesame).

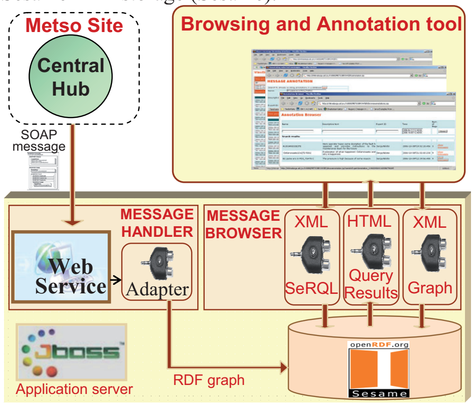

Figure 2: Architecture of the system.

The RDF storage contains an *Ontology* that plays the role of a schema for all data within the storage. Based on the analysis of SOAP/XML messages, we have defined main concepts (classes) with the corresponding properties (see Figure 3).

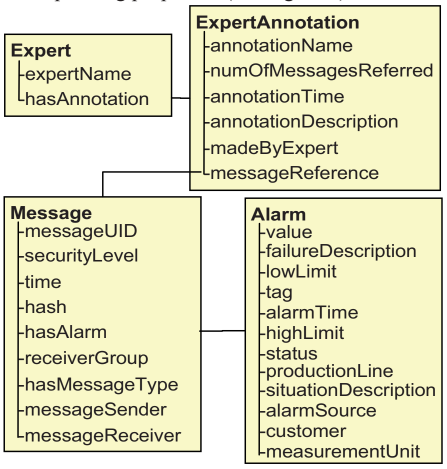

Figure 3: Ontology classes.

The *Message* class describes such message properties as message sender and receiver, message reception time, etc. The *Message* class also refers to an *Alarm* class, which contains information about the reason for message generation, such as measurements of sensors, status data and exact module of the production line where the alarm happened. The *ExpertAnnotation* class defines the structure for labelling groups of messages with an expert's decision and has references to instances of the *Message* class.

The *Message Browser* component provides a web-based interface for browsing and filtering messages stored in the RDF-storage, according to user-defined filtering criteria.

The purpose of message filtering is to distinguish the groups of messages leading to exceptional situations. The expert provides annotations for message groups which are stored to the RDF-storage and that can be used as samples for machine learning algorithms. The client-server interaction is implemented using AJAX technology (Garrett, 2005), which provides a more dynamic script-based interaction with the server (see Figure 4). When a user performs any action that requires invocation of server functionality, the script on a client side wraps the required parameters into XML format and sends it to the server. For example, in order to filter the messages, a user selects the needed parameters and specifies parameter values within the corresponding textboxes (see Figure 4).

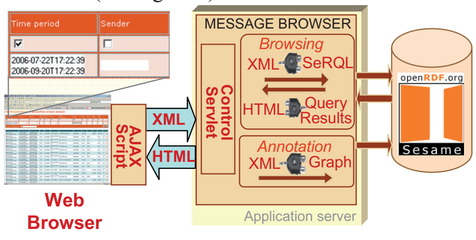

Figure 4: Client-server interaction.

On the server side, the *Control Servlet* handles the XML document. For filtering, it generates a SeRQL query (Broekstra, 2004) and executes it. On the client side, a dedicated callback script function processes the response and shows the result in a web browser.

3.2 Integration with the Agent Platform

We realize that the extension of the system will challenge the complexity of development and maintenance. That is why, following the autonomic computing paradigm (Kephart, 2003), we have tested agent-based scenario (see Figure 5) implemented on a JADE agent platform (Bellifemine; 2001). We have assigned an agent to manage RDF-storage activities (*Metso Storage Agent*) and provided a *Metso Expert Agent* to interact with a maintenance expert.

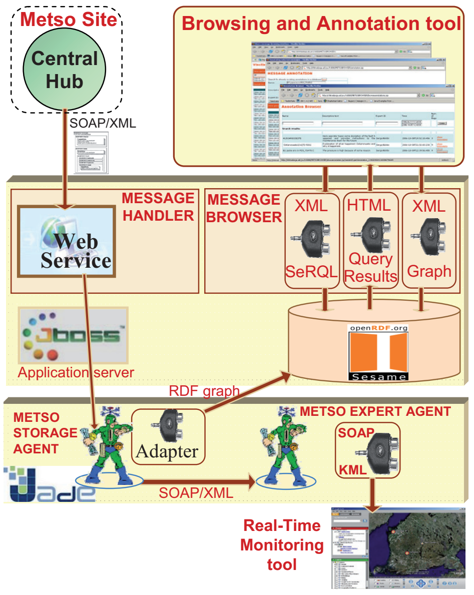

Figure 5: Agent-enabled system.

The messages coming from customers are handled by the *Metso Storage Agent*, which incorporates *Adapter* to perform transformation and storage. Then, the *Metso Storage Agent* sends the message to the *Metso Expert Agent*, which updates the situation on a Real-time Monitoring Tool and provides an expert with the message content and a link to the browsing and annotation tool.

4 CONCLUSIONS

Although we have succeeded with the implementation of the solution presented here, there are still many issues to cope with in order to meet key industrial requirements, such as scalability, maintainability and robustness. RDF-storages can handle billions of triples, but there are no mature semantic storage-oriented development patterns or guidelines. Nevertheless, the simplicity and efficiency of querying, as well as model extending, provide incontestable arguments in favour of semantic data storages. The ontological domain model brings more benefits to customers when there are more sources integrated. However, the complexity of such a system, if developed using conventional approaches, will be too burdensome to maintain and extend. In order to distribute the complexity, we introduce self-manageable entities in the agent-based communication scenario.

ACKNOWLEDGEMENTS

This research has been supported by the SmartResource project, funded by TEKES, and the industrial consortium of Metso Automation, TeliaSonera and ABB. The preparation of this paper was partially funded by the COMAS graduate school.

REFERENCES

Bellifemine, F., Poggi, A., and Rimassa, G. 2001. JADE: a FIPA2000 compliant agent development environment. In Proceedings of the Fifth international Conference on Autonomous Agents (Montreal, Quebec, Canada). AGENTS '01. ACM Press, New York, NY, 216-217. DOI= http://doi.acm.org/10.1145/375735.376120

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

Broekstra, J., Kampman A., and F. van Harmelen. Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. In D. Fensel, J. Hendler, H. Lieberman, and W.Wahlster, editors, Semantics for the WWW. MIT Press, 2001.

Broekstra, J., Kampman, A. SeRQL: An RDF query and transformation language. In Proceedings of the International Semantic Web Conference, ISWC 2004, Hiroshima, Japan.

Garrett, J., Ajax: A New Approach to Web Applications (white paper).,http://www.adaptivepath.com/publications/essays/archives/000385.php, February 2005.

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.

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

RDF – Resource Description Framework, A W3C Recommendation, Feb 2004, http://www.w3.org/RDF/

SmartResource – a TEKES funded project, http://www.cs.jyu.fi/ai/OntoGroup/SmartResource_details.htm

SOAP – Simple Object Access Protocol, A W3C Recommendation, 2003, http://www.w3.org/TR/soap/

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.

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.

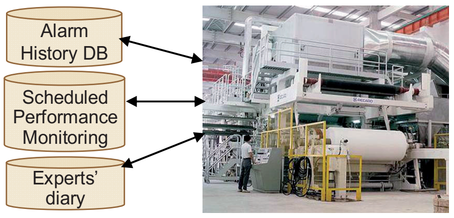

Figure 1. The IT-infrastructure of the paper machine

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).

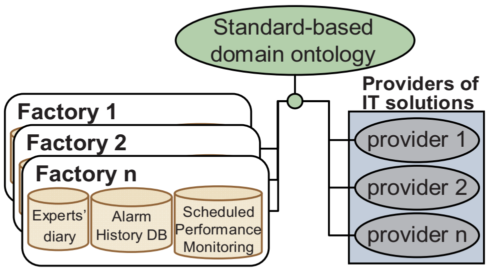
Standard-based domain ontology
Providers of IT solutions
Factory 1
Factory 2
Factory n
Experts' diary
Alarm History DB
Scheduled Performance Monitoring
provider 1
provider 2
provider n

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.

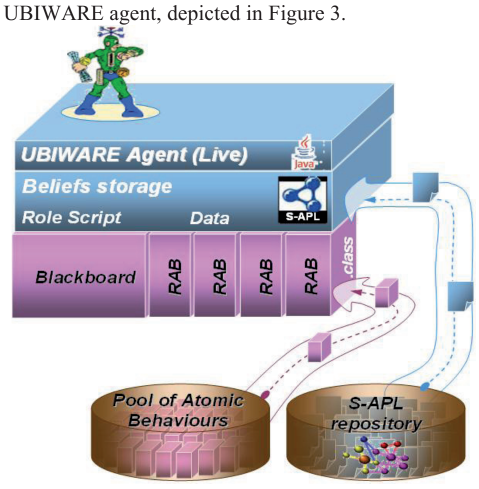

Figure 3. UBIWARE Agent

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

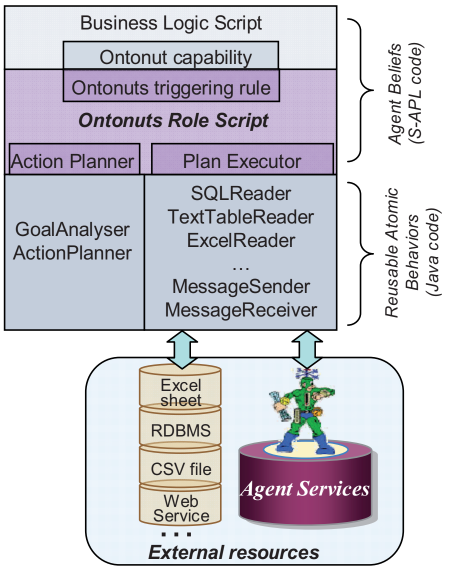

Figure 4. Architecture of Ontonuts

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).

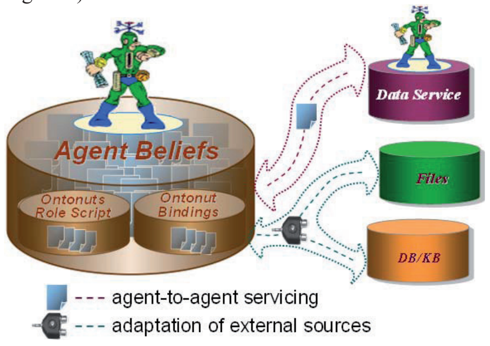

Figure 5. Distributed querying in UBIWARE

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).

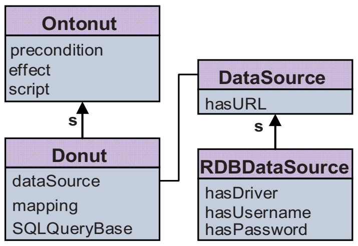

Figure 6. A fragment of Donuts ontology

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.

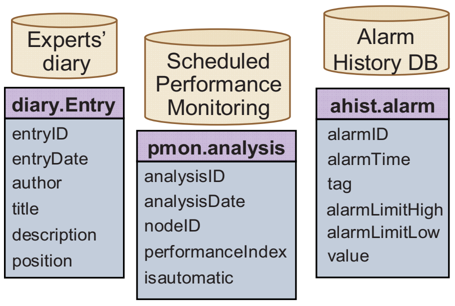

Figure 7. Sample database tables

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.

V

SOFIA: AGENT SCENARIO FOR FOREST INDUSTRY

by

Sergiy Nikitin, Vagan Terziyan and Minna Lappalainen 2010

Proceedings of the 12th International Conference on Enterprise Information Systems (ICEIS 2010), Funchal, Madeira, Portugal, 8-12 June, 2010, pp. 15-22.

SOFIA: AGENT SCENARIO FOR FOREST INDUSTRY
Tailoring UBIWARE Platform Towards Industrial Agent-driven Solutions

Sergiy Nikitin, Vagan Terziyan
Industrial Ontologies Group, University of Jyväskylä, Mattilanniemi 1, Jyväskylä, Finland
{sergiy.nikitin, vagan.terziyan}@jyu.fi

Minna Lappalainen
Fixteri Oy, Finland
minna.lappalainen@fixteri.fi

Keywords: Agent Technology, Distributed Applications, Semantic Web, Forestry Services.

Abstract: Current economical situation in Finnish forest industry desperately calls for higher degree of efficiency in all stages of the production chain. The competitiveness of timber-based products directly and heavily depends on the raw material cost. At the same time, the successes of companies, that use timber, determine the volumes of the raw wood consumption and, therefore, drive forest markets. However, wood consuming companies (e.g. paper producers) can not unilaterally dictate logging and transportation prices to their contractors, because profitability of those, has already reached its reasonable margins (Vesterinen, 2005, Penttinen, 2009). Recent research conducted in 2005-2008 shows 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; however privacy and trust issues prohibit those companies from taking the open environment solutions. Therefore, the researchers have suggested new mediator-based business models that leverage the utilization and preserve current state of affairs at the same time. New business solutions for logistic optimization can be built, when a unified view on the market players is possible. Nevertheless, with fast development of communications, RFID and sensor technologies, forest industry sector is experiencing a technological leap. The adoption of innovative technologies opens possibilities for enactment of new business scenarios driven by bleeding edge ICT tools and technologies. We introduce an application scenario of the semantic agent platform called UBIWARE to the forest industry sector of Finland.

1 INTRODUCTION

Current economical situation in Finnish forest industry desperately calls for higher degree of efficiency in all stages of the production chain. The competitiveness of timber-based products directly and heavily depends on the raw material cost. At the same time, the successes of companies, that use timber, determine the volumes of the raw wood consumption and, therefore, drive forest markets. However, wood consuming companies (e.g. paper producers) can not unilaterally dictate logging and transportation prices to their subcontractors, because profitability of those, has already reached its reasonable margins (Vesterinen, 2005, Penttinen, 2009). Recent research conducted in 2005-2008 shows extremely high degree of inefficiency in logistic operations amongst wood 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; however privacy and trust issues prohibit those companies from taking the open environment solutions. Moreover, the logistic optimization within one company is complicated due to heterogeneity of information systems used in harvesters and timber trucks. The same harvester, in order to perform logging for e.g. three different clients, needs to use three distinct systems. Those systems are not integrated, thus the logger has to learn three different interfaces still not having a composite view. Same applies to the subcontractor's office desktop systems, where, operator needs to manage e.g. 5 harvesters having different ordering systems from its clients. Although, an increasing number of logging and transportation subcontractors have or control two or more machines, still the logistic plan is mainly done manually or requires manual work to align the data from different systems.

The research performed in 2005-2009 (Lappalainen, 2009) has suggested new mediator-based business models that leverage the utilization and preserve current state of affairs at the same time. New business solutions for logistic optimization can be built, when a unified view on the market players is possible. Although, the companies involved in the forestry sector have a high degree of the ICT infrastructure, yet they do not utilize it to improve the situation cooperatively. The ICT solutions used in a majority of cases are developed as black box standalone applications, therefore the integration of those raises technological challenges. Traditional system integration, if applied here, would become an expensive task involving changes to the existing solutions on the companies' site or building a new system from the scratch. According to the surveys conducted in Finland, currently, forest market players are more or less satisfied with the existing ICT solutions and are neither interested, nor capable to spend resources for new information systems and technologies. The innovative ICT solution, if it takes place, should seamlessly penetrate into the existing infrastructure. The revolutionary changes would not be accepted, unless dictated by market leaders in wood consumption. Those, however, are tied by the contracts with their ICT solution providers.

In this paper we present an outcome of the preparatory project – a proposal of innovative ICT solution for forest industry. In Section 2 we explore the problem domain and define a driving use case that calls for a new ICT solution. Section 3 presents the architecture of the semantic middleware platform that can be considered as a construction tool for a new solution. The extension of the middleware platform to the forestry domain is discussed in Section 4. In Section 5 we present related work and conclude in Section 6.

2 ICT IN FORESTRY

The business environment considered in this work involves wood buyers, forest owners, forest owner associations, and forest and transportation contractors. The interactions are automated, i.e. the wood purchase and cutting orders are done via information systems.

2.1 State of the Art: Wood Purchase Scenario

A forest owner either finds a forest buyer or contacts forest owners association with the request to sell forest in a certain area on the owner's behalf. The forest counsellor (either from the association or from the forest buyer) goes to that area to estimate what will be an approximate outcome and of what quality (classified by dimensions).

Forest counsellor is equipped with the handheld device with the GPS-receiver and her/his conclusion automatically goes either to the association or to the buyer database. Upon counsellor's decision, the operator of the association or buyer enterprise makes an order to a forest contractor for harvesting service. The order includes the amount and optimized sizes of the logs to be cut. The forest contractor loads the order to the information system of the harvester and starts felling. After the felling is done, the timber is forwarded to the roadside storage place where it can be accessed by the timber truck. The operator of the association or buyer's enterprise then does next order to the transportation contractor to deliver the wood to the destination place (e.g. sawmill, pulp mill, power plant, etc.).

2.2 Contractor's Viewpoint

A large share of forest logging or transportation contractors usually own two or more harvesters or timber trucks. Furthermore, increasing number of them receives orders from two or more order makers (wood buyers, forest management associations) (see Figure 1). In order to receive these orders, the contractors have to install respective information systems. Each buyer and/or association has its own tailored solution, which is incompatible with the solutions from others. The contractor has to learn peculiarities of each system, such as different internal codes for wood types, different user interfaces and principles of functioning, etc. *The system heterogeneity makes it impossible for the contractor to integrate the data from those systems and obtain an integrated view of his/her own business.* At the moment the integration is done manually by reading and inserting data into one table, or, sometimes, calculations on the paper are used. In the following subsection we present a desired functionality of the logistics management platform for harvesting and transportation SME's in a form of a use case.

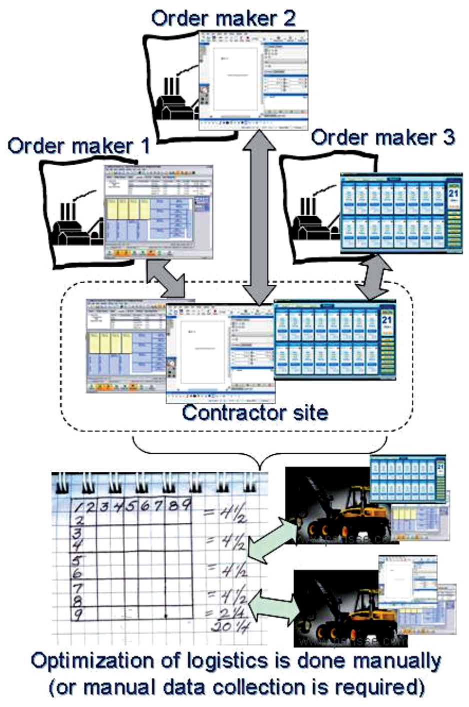
Order maker 2
Order maker 1
Order maker 3
Contractor site
Optimization of logistics is done manually
(or manual data collection is required)

Figure 1: Contractor's view point.

2.3 Driving Use Case: A Platform for Integrated Logistics Optimization

A logging contractor company called KORJUUBEST Oy has 5 harvesters and three different order makers (customers). Timo Saarinen is a company owner and CEO. Timo likes to keep things controlled in his own hands; therefore, he also does the operator job when he is free from traveling and meetings. Timo has double backup, if he is busy, then another operator in the office can substitute him, or Timo can turn on the automatic mode in his new logistics control platform called SOFIA. Every morning Timo comes to his office and after a morning coffee he starts his laptop and logs into the SOFIA platform. The system shows current situation of Timo's harvesters and their status (e.g. *working*, *stopped*, *short maintenance break*, or *out of order*). Timo chooses the harvester icon and browses the tasks assigned and planned. He can also browse the history. After a short look on the harvesters' status Timo opens the bookmark called *orders*. He sees the integrated currently pending order list as well as the orders already planned, based on the long term contracts. The system proposes the optimized order assignment table for the next week, where Timo can reassign tasks to other harvesters if he thinks it is needed and press "Approve" button. The system will send new (not yet sent) logging tasks to the harvesters and the operators will immediately see the new task information in the operator's web-based view. The operators can download order files attached to the task and load them to the harvester's native system. In the evening, upon the completion of the work, the operators can send progress reports or, if the task was completed, a closing report.

SOFIA platform can be configured so, that it automatically sends closing reports to the order maker, or, it may wait for an approval and manual submission from Timo or the operator in charge.

After a half a year of successful operation and optimized utility, Timo has realized that his company can serve at least one more order maker (customer) and luckily, he has met a potential client from big company called Metsänhoitajat Oy last week in sauna. They have agreed to meet in more formal way as soon as they clarify how much work would be needed to integrate their information and ordering systems. Timo has called to the SOFIA maintenance center and has received a surprisingly good answer. With the minimum cost a new customer's ordering system can be connected to Timo's platform with no need to stop it. Luckily, Metsänhoitajat Oy company has already worked with other contractor, who is using SOFIA platform, therefore the system adapter is already available, it only needs to be configured for Timo's platform. Even, though, the adapter wasn't ready, it would not take longer than one month to plug a new ordering system to SOFIA.

In the evening, after a successful meeting with Metsänhoitajat Oy where a new contract was signed, Timo has made an order to SOFIA maintenance center for a platform configuration. Next day Timo could already see a new partner in his system and in short time, new orders have started to come.

In the following section we present a middleware platform called UBIWARE – a convenient tool for the implementation of SOFIA platform.

3 UBIWARE PLATFORM

UBIWARE is a generic domain independent middleware platform (Katasonov et al., 2008) that is meant to provide support for integration, interoperability, adaptation, communication, proactivity, self-awareness and planning for different kinds of resources, systems and components (e.g. data information and knowledge, software and services, humans, hardware and processes). The UBIWARE platform is developed inline with the fundamental vision towards GUN - Global Understanding Environment (Terziyan, 2003, 2005; Kaykova *et al.*, 2005). In GUN various resources can be linked to the Semantic Web-based environment via adapters (or interfaces), which include (if necessary) sensors with digital output, data structuring (e.g. XML) and semantic adapter components (XML to Semantic Web). Software agents are to be assigned to each resource and are assumed to be able to monitor data coming from the adapter about the state of the resource, make decisions on behalf of the resource, and to discover, request and utilize external help if needed. Agent technologies within GUN allow mobility of service components between various platforms, decentralized service discovery, utilization of FIPA communication protocols, and multi-agent integration/composition of services.

When applying the GUN vision, each traditional system component becomes an agent-driven "smart resource", i.e. proactive and self-managing. This can also be recursive. For example, an interface of a system component can become a smart resource itself, i.e. it can have its own responsible agent, semantically adapted sensors and actuators, history, commitments with other resources, and self-monitoring, self-diagnostics and self-maintenance activities.

At the same time, UBIWARE naturally integrates such domains as Semantic Web, Proactive Computing, Ubiquitous Computing, Autonomous Computing, Human-Centric Computing, Distributed AI, Service-Oriented Architecture and Enterprise Application Integration.

3.1 UBIWARE Platform Architecture

UBIWARE has two main elements: an agent engine, and S-APL – a Semantic Agent Programming Language (Katasonov and Terziyan, 2008) for programming of software agents within the platform.

The architecture of UBIWARE agent (Figure 2) consists of a *Live* behavior engine implemented in Java, a declarative middle layer, and a set of Java components – *Reusable Atomic Behaviors* (RABs).

RABs can be considered as sensors and actuators, i.e. components sensing or affecting the agent's environment, but are not restricted to these. A RAB can also 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.

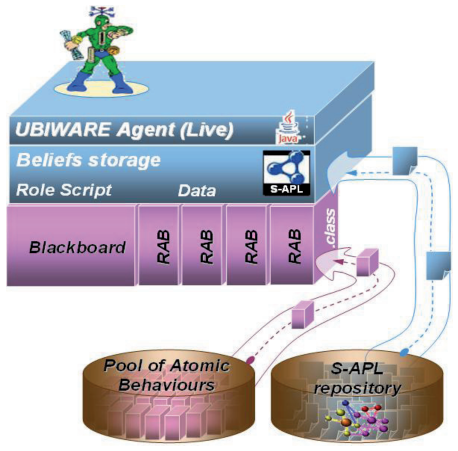

Figure 2: UBIWARE Agent.

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, logically the UBIWARE platform, consists 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.

As Figure 2 shows, an S-APL agent can obtain the needed data and rules not only from local or network documents, but also through querying S-APL repositories. Such a repository, for example, can be maintained by some organization and include prescriptions (lists of duties) corresponding to the organizational roles that the agents are supposed to play.

Technically, the implementation is built on top of the JADE – Java Agent Development Framework (Bellifemine et al. 2007), which is a Java implementation of IEEE FIPA specifications.

4 UBIWARE MEETS FORESTRY

The business models that were thoroughly studied in the recent research (Lappalainen, 2009), stumbled in the technological challenges that are being tackled in the UBIWARE. At the same time, UBIWARE platform should be tailored to the industrial domains, in order to attract businesses. Therefore a SOFIA platform (SOFIA stands for Seamless Operation of Forest Industry Applications) described in Section 2.3, when developed on top of the UBIWARE, will benefit from inherent flexibility and extensibility and ensure sustainable ICT infrastructure for logging and transportation SMEs.

4.1 Tailoring UBIWARE to Forestry

The logistics optimization drives the main direction of platform development. However, in order to solve optimization tasks, the platform requires adaptation and connectivity problems to be resolved first. Those, in turn, call for a unified domain model (domain ontology).

As soon as SOFIA platform will serve as an integrator of information systems provided from different order makers (wood buyers and forest owner associations), the orders coming from different systems will be gathered to one integrated view allowing the contractor to apply logistics optimization tool and decrease useless overheads in operation (see Figure 3).

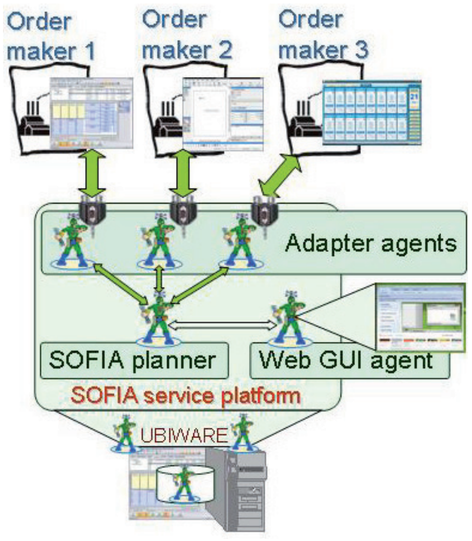

Figure 3: SOFIA platform.

To perform integration, we define a set of target information systems (based on the case studies) and make deep analysis of the connectivity options and internal data models used. The integration will result in construction of adapters to the respective systems. The configuration on the order maker site may still be required to redirect data flow from order maker to SOFIA platform.

The order makers, however, may not be flexible in changing some settings or opening access to their data and systems. Therefore, we have a requirement to minimize changes on the order maker side if not to avoid at all.

4.2 Handling Connectivity Challenges

The main implementation challenge of the platform is connectivity. Figure 4 shows a generic data exchange scenario between order maker and contractor.

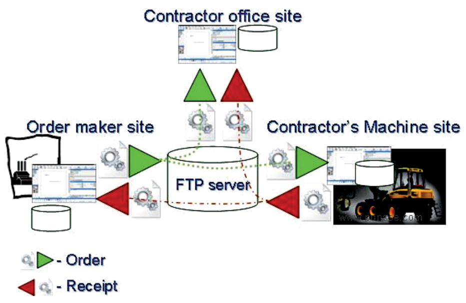

Figure 4: Data flow between order maker and contractor.

The flow is mostly organized via FTP server, which is checked by the client software installed on the contractor's machine site and, sometimes, at the office. The order flow goes directly from the order maker to the machine. In such situation a contractor is unable to decide, which order goes to which machine, i.e. the machine is rather directly controlled by the order maker (preceded by a generic contract of course). The software for data exchange is proprietary and, therefore, does not provide any API. The intermediate files, though, appear on the FTP-site as well as in temporary folders on local machines.

In such scenario even data collection may put platform development to the tight corner. We propose virtualization approach (see next subsections) to handle the issue and introduce two possible workaround scenarios.

4.2.1 Contractor Site as a Firewall

In case when an FTP server can be accessed by our platform software (only username and password are required and known), we can fully emulate the behaviour of a harvester or a truck. No changes on the order maker site are needed.

In case, when an FTP server can only be accessed by the proprietary client software (the password and username may be hardcoded and not known to the contractor), we can move the client from the machine to the contractor site and access FTP server from it. Temporary files, stored by the client, then can be sensed by the platform and passed to the responsible agent.

4.2.2 Machine Agent

In neither case, when no FTP access can be arranged, nor client software can be moved from the machine site (e.g. proprietary restrictions), we can establish our listener software on the machine site to catch the data from the client and send the data to the contractor site. In this case, we have extra delay time because of additional data transfer step from the machine to the contractor site and back. This approach may change the architecture of the whole system drastically and may require a lot more efforts. Nevertheless it is still possible.

4.3 Virtualization of Forestry Market

The requirement to preserve systems of order makers untouched can be accomplished by introducing the know-how of the SOFIA platform – a concept of virtual machine applied to harvesters and timber trucks. The contractor creates an interface-like view to his/her harvesting and transportation facilities by means of virtual machines (see Figure 5).

From the service order maker point of view the contractor looks the same, however, the real equipment of the contractor is hidden. The orders are made seamlessly, but the assembly chosen for the execution, is virtual. The service provider then has the opportunity to build an optimized operations plan and after that assign tasks to the real units.

The virtualization may go beyond the SME boundaries. Several contractors may establish virtual enterprise that works as a proxy for order makers. Such enterprise would have better optimization capacity because of wider order and equipment base. The business model, that clearly explains and guarantees the benefit to stakeholders, yet to be elaborated. The model should take into account region-specific circumstances and context.

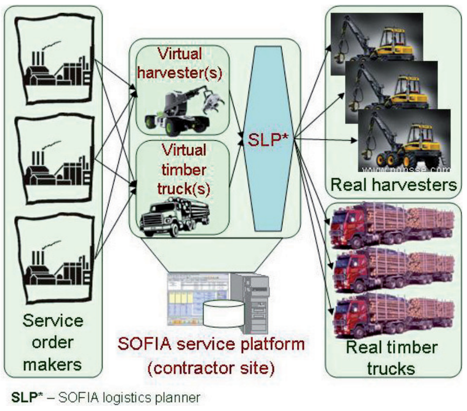

Figure 5: A virtual machine concept.

Although, physical resources virtualization (harvesters and trucks) is attractive to contractors, it may also lead to complexities in resource planning in global scale. In general, if the same resource is present in two independent planning systems (e.g. two virtual contractor SMEs have signed the contract with the same harvester owner), then both systems may build long-term plans, expecting the resource to be available. The problem may show up, only when a detailed short-term contract has to be signed and both virtual harvesting contractors are pretending to employ the same harvester. The business model should *exclude ambiguity* and guarantee the availability of the resources at the execution time. We can compare the problem to the car rental process, where we see the capacity (cars available) and know the car class (e.g. Ford Focus or analogous), but we do not know the exact car license plate id, before we come to the office and get the keys. In the simplest case, the rental company is the owner of the car, but in harvesting we may have a situation, when a harvester owner has signed contracts with two or more harvesting SMEs.

4.4 SOFIA beyond National Boundaries

The platform and the model described above fit well forestry market in global scale. We expect that a globally present enterprise can sell platform services worldwide. Although, the localization requires quite significant effort, which is not in the nature of the global service, still it is compensated by limited number of harvester manufacturers (only three key manufacturing enterprises). The small amount of manufacturers means significant reduction of software adaptation efforts. Running one nationally wide service platform would already include adapters to the systems of the key manufacturers. Next, the middleware platform we use, possesses the features for easy tailoring to local region-specific requirements. The platform architecture employs semantic technology, which can be considered as most expressive one for domain modelling. The semantic nature combined with the agent technology brings other benefits as well – adaptivity and configurability that allow the platform users to get new business models up and running with small effort. Configurability also makes the maintenance of the platform less resource-consuming for the customer.

The web-based solution on the global scale, if it takes place, can utilize cloud computing (Hayes, 2008) to ease scalability and optimize expenditures for the hardware and software infrastructure. The components of the platform (see Figure 6), when run in the cloud can be updated or configured on-the-fly.

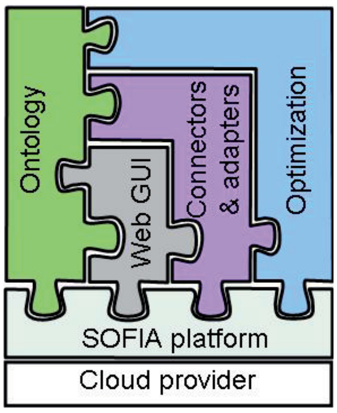

Figure 6: SOFIA component view.

The platform behaviour is specified in the ontology – a backbone of both the data and the business logic. We can consider ontology as a rich configuration file that describes structure of the software components being run as well as the data. The Web GUI component may undertake minor localization changes, whereas Connectors & Adapters will differ significantly from region to region, due to a variety of local information systems at wood buyer sites that have to be connected. The optimization algorithms and methods may require region-specific settings for better efficiency, but otherwise remain untouched.

In this Section we have presented the analysis of the domain-specific features that have to be implemented on top of the existing middleware platform. The analysis shows that technical implementation is feasible in spite of the state of the market and relationships amongst market players.

5 RELATED WORK

Latest industrial ICT trends define inter-component and system interoperability as a key direction.

The interoperability is also known as one of the major future challenges in ICT. Current industrial standardization efforts aim at resolving this challenge by creating a unified vocabulary of communication, or, in other words, a standard. Forest industry is not an exception. Standardization and interoperability form a basis for competitive market, and hence, for cost-efficient production. In recent years forest industry have run a set of standardization projects, e.g. papiNet (www.papinet.org) and its initiative - WoodX, Edifact (www.unece.org/trade/untdid/welcome.htm) and StanForD (http://www.skogforsk.se/upload/6867/StanForD_MainDoc_070327.pdf).

The standards mentioned above simplify the implementation of SOFIA in order of magnitude and, therefore, we have to make thorough analysis of standards already adopted as well as those being developed. SOFIA's ontology should be built standard-compliant and allow easy standard-based document and interface generation.

The improvement of the wood production chain is also a subject of integrated EU FP7 project called "IndisputableKey" (www.indisputablekey.com). The project aims at a new methodology and advanced technologies that improve the use of wood and optimize the forest production. The project is targeting the supply chain improvement as a whole, whereas SOFIA is a contractor SME-oriented at the same time applying and developing ICT technologies far beyond those currently available.

In (Frayret et al., 2007; Forget et al., 2008) the authors state that forestry companies are facing the need to re-engineer their organizational processes and business practices taking into account other companies in forest industry. An agent-based approach is proposed to tackle the problem of dynamic planning in the supply chain. SOFIA rather approaches the same domain from the software architecture viewpoint and introduces innovative software platform model for forestry contractors.

6 CONCLUSIONS

A unified platform solution for forest industry faces the ICT challenges that were foreseen in GUN activities early in years 2002-2003. The UBIWARE platform being designed to resolve such challenges, still remains domain independent, and, therefore, has to be tailored and extended to meet domain-specific needs. Such platform customization is a first step towards GERI (Global Enterprise Resource Integration) – where various industrial domains will be taken into account. At the moment, UBIWARE-based industrial applications are naturally needed for proper platform evolution as a whole. SOFIA platform will have an extended tool set (RABs and S-APL models) on top of UBIWARE to solve forest industry sector tasks. We believe that success of SOFIA forest industry platform can bring a new breath both to the forestry and to the ICT worlds.

The results of the research published in (Lappalainen, 2009), state that utilization of business models with more than one customer for the harvesting, transportation and chipping contractors, can save approximately 50 million euro annually in the forest biomass supply chain in Finland only. A simulation study conducted within the same project has shown that annual cost savings in raw wood harvesting only would account for 21 million euro at least if business models and ICT-tools would support it (Väätäinen et al., 2008).

In addition to the cost savings mentioned above, SOFIA can be used to provide higher level harvesting and transportation services for customers at the same time serving simultaneously many of them. SOFIA would enable more efficient order handling and organization of right-time deliveries while minimizing the risk of human failures.

We have presented the results of the preparatory project. This work has been inspired by thorough analysis of business opportunities that led us to look for technological implementation challenges and workarounds. At the moment we are considering both business-oriented as well as science-oriented directions for further development of SOFIA.

REFERENCES

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing Multi-Agent Systems with JADE*. Wiley.

Frayret, J.-M., D'Amours, S., Rousseau, A., Harvey, S., Gaudreault, J. 2007. Agent-Based Supply Chain Planning in the Forest Products Industry. *International Journal of Flexible Manufacturing Systems, 19(4),* p. 358-391.

Forget, P., D'Amours, S., Frayret, J-M., Multi-behavior agent model for planning in supply chains: An application to the lumber industry, Robotics and Computer-Integrated Manufacturing, Volume 24, Issue 5, October 2008, Pages 664-679, ISSN 0736-5845, DOI: 10.1016/j.rcim.2007.09.004.

Hayes, B. 2008. Cloud computing. *Commun. ACM* 51, 7 (Jul. 2008), 9-11. DOI= http://doi.acm.org/10.1145/1364782.1364786

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., and Terziyan, V., 2008. Semantic agent programming language (S-APL): A middleware platform for the Semantic web. In Proc. 2nd IEEE International Conference on Semantic Computing, pp. 504–511, 2008.

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

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

Penttinen, M. & Mikkola, J. & Rummukainen, A., 2009. Profitability of wood harvesting enterprises. Working Papers of the Finnish Forest Research Institute, No. 126.

Terziyan, V., 2003. 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, pp.279-291.

Terziyan, V., 2005. 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.

Vesterinen, M., 2005. 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.

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.

VI

MASTERING INTELLIGENT CLOUDS: ENGINEERING INTELLIGENT DATA PROCESSING SERVICES IN THE CLOUD

by

Sergiy Nikitin, Vagan Terziyan and Michal Nagy 2010

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.

MASTERING INTELLIGENT CLOUDS
Engineering Intelligent Data Processing Services in the Cloud

Sergiy Nikitin, Vagan Terziyan and Michal Nagy
Industrial Ontologies Group, University of Jyväskylä, Mattilanniemi 1, Jyväskylä, Finland
{sergiy.nikitin, vagan.terziyan, minagy}@jyu.fi

Keywords: Agent Technology, Cloud Computing, Semantic Web, Cloud Services, Ubiware.

Abstract: Current Cloud Computing stack mainly targets three architectural layers: Infrastructure, Platform and Software. These can be considered as services for the respective layers above. The infrastructure layer is provided as a service for the platform layer and the platform layer is, in turn, a service for the Software layer. Agent platforms fit the "Platform as a service" layer within this stack. At the same time, innovative agent-oriented approaches to programming, open new possibilities for software design in the cloud. We introduce main characteristics of our pilot agent platform called UBIWARE and offer flexible servicing architecture within the cloud platform, where various components and systems can configure, run and reuse intelligent cloud services to provide higher degree of flexibility and interoperability for their applications.

1 INTRODUCTION

Fast development of network technologies has recently brought back to life business models with the "thin client" architecture. Powerful data centers connected to the internet via broadband networks can minimize IT-infrastructure of any company to a set of simple terminals with less demanding system requirements. All the software and data can reside on the data center side, making user access easy and location independent. The providers offer different payment schemes as "pay-per-use" or subscription-based, that seems to be advantageous compared to standard IT-infrastructure expenses. The approach has got a set of new features and a new branding name: "Cloud Computing" (Hayes, 2008; Foster, 2008).

Cloud management platforms provide API for management either through command line or a remote method calls. The API, however, is used mainly by system administrators, who take care of proper functioning of services within the cloud. Management of the cloud platform is considered as something that a system administrator should arrange and do. Usually administrators use batch files for managing routine tasks and resolving exceptional situations.

At the same time, more and more software architecturing paradigms call for new approaches to software design and development, where software components get a certain degree of self-awareness, when a component can sense its own state and act based on the state changes. The vision of Autonomic Computing (Kephart, 2003) proposes to handle the complexity of information systems by introducing self-manageable components, able to "run themselves." The authors state, that self-aware components would decrease the overall complexity of large systems. The development of those may become a "nightmare of ubiquitous computing" due to a drastic growth of data volumes in information systems as well as heterogeneity of ubiquitous components, standards, data formats, etc. The Cloud Computing and Autonomic Computing paradigms will become complementary parts of global-scale information systems in the nearest future. Such a fusion sets the highest demands to the software architects because cloud platforms will have to provide self-management infrastructure for a variety of complex systems residing in the same cloud, separated virtually, but run physically on the same hardware. At the same time, the cloud platform itself may possess features of self-aware complex system. A variety of self-aware components of different nature (i.e. end-user oriented, infrastructure-oriented, etc.) will need a common mechanism for interoperability, as far as they may provide services to each other.

The vision of GUN – Global Understanding Environment (Terziyan, 2003, 2005; Kaykova et al.,

2005) has introduced a concept of “Smart Resource” and a notion of an environment where all resources can communicate and interact regardless of their nature. In GUN various resources can be linked to the Semantic Web-based environment via adapters (or interfaces), which include (if necessary) sensors with digital output, data structuring (e.g. XML) and semantic adapter components (e.g. XML to Semantic Web). Software agents are assigned to each resource and are assumed to be able to monitor data coming from the adapter about the state of the resource, make decisions on behalf of the resource, and to discover, request and utilize external help if needed. Agent technologies within GUN allow mobility of service components between various platforms, decentralized service discovery, utilization of FIPA communication protocols, and multi-agent integration/composition of services.

When applying the GUN vision, each traditional system component becomes an agent-driven “smart resource”, i.e. proactive and self-managing. This can also be recursive. For example, an interface of a system component can become a smart resource itself, i.e. it can have its own responsible agent, semantically adapted sensors and actuators, history, commitments with other resources, and self-monitoring, self-diagnostics and self-maintenance activities.

In this paper we introduce a flexible servicing architecture within the cloud platform, where various components and systems can configure, run and reuse intelligent cloud services to provide higher degree of flexibility and interoperability for their applications. We use our pilot agent platform developed in accordance with GUN vision called UBIWARE to show how cloud computing can expand platform functionality and at the same time how an agent platform can become a high-level service provisioning instrument in the cloud.

The paper is organized as follows: In the next Section we discuss architectures of state-of-the-art cloud computing platforms and explore the possibilities for the interoperability mechanisms. Section 3 presents the architecture of the semantic middleware agent platform and explores possible options of connectivity with the cloud. Section 4 describes the scenarios and the architecture of the agent-driven intelligent servicing platform for a cloud. In Section 5 we discuss related work and conclude in Section 6.

2 STATE OF THE ART IN CLOUD INTELLIGENCE PLATFORMS

Architecture of current Cloud Computing stack mainly targets three layers: Infrastructure, Platform and Software. These layers can be considered as services for the respective layers above. The infrastructure as a service (IaaS) is provided to the platform layer and the platform becomes a service (PaaS) for the software layer. And finally, the software as a service layer (SaaS) brings the topmost end-user web services to clients (see Figure 1).

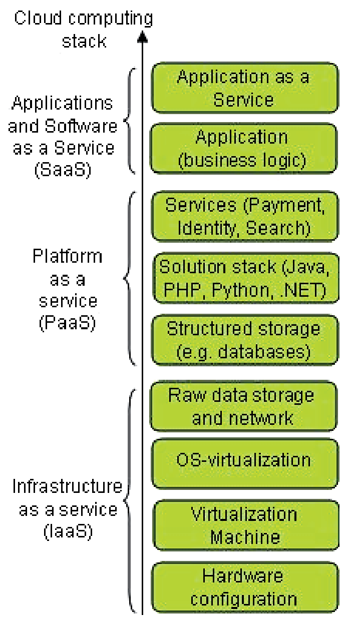
Cloud computing stack
Applications and Software as a Service (SaaS)
Platform as a service (PaaS)
Infrastructure as a service (IaaS)
Application as a Service
Application (business logic)
Services (Payment, Identity, Search)
Solution stack (Java, PHP, Python, .NET)
Structured storage (e.g. databases)
Raw data storage and network
OS-virtualization
Virtualization Machine
Hardware configuration

Figure 1: Cloud computing stack.

Cloud providers market niche is already a competitive field. Several big players are currently active in the market, e.g. SalesForce.com (SFDC), NetSuite, Oracle, IBM, Microsoft, Amazon EC2, Google Apps, etc. For a comprehensive survey of cloud computing systems see (Rimal et al., 2009). The services of the platform layer are in the scope of this work. In the next section we present a middleware platform and later introduce a new servicing approach in the cloud stack.

3 UBIWARE PLATFORM

UBIWARE has two main elements: an agent engine, and S-APL – a Semantic Agent Programming Language (Katasonov and Terziyan, 2008) for programming of software agents within the platform.

The architecture of UBIWARE agent (Figure 2) consists of a *Live* behavior engine implemented in Java, a declarative middle layer, and a set of Java components – *Reusable Atomic Behaviors* (RABs).

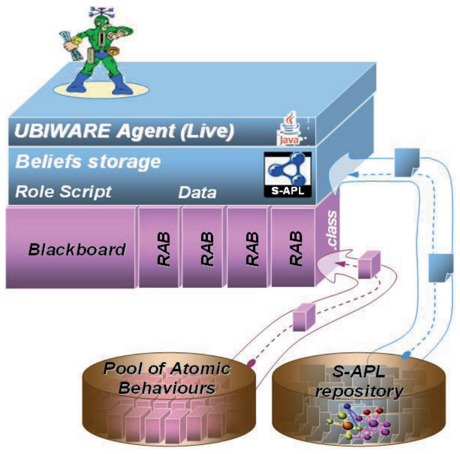

Figure 2: UBIWARE Agent.

RABs can be considered as sensors and actuators, i.e. components sensing or affecting the agent's environment, but are not restricted to these. A RAB can also 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. 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, logically the UBIWARE platform, consists 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.

As Figure 2 shows, an S-APL agent can obtain the needed data and rules not only from local or network documents, but also through querying S-APL repositories. Such a repository, for example, can be maintained by some organization and include prescriptions (lists of duties) corresponding to the organizational roles that the agents are supposed to play.

Technically, the implementation is built on top of the JADE – Java Agent Development Framework (Bellifemine et al. 2007), which is a Java implementation of IEEE FIPA specifications.

4 MASTERING INTELLIGENT CLOUD PLATFORM

Cloud computing providers offer various stack configurations with different sets of software and services inside. Theoretically, one can buy any configuration from the cloud provider; however this configuration will have nothing to do with the already running business logic of the customer. The application scenarios a customer wants to run will have to be adjusted. For example, consider a case, when a customer buys a virtual server with the MySQL database installed and a Java solution stack available. On top of this stack customer runs a workflow engine, e.g. BPEL-based. The user will have to install the engine, and then adjust local data storage settings (passwords, tables, queries). Then the process descriptions (BPEL files) should be adjusted to work with local settings. In some cases this process may be avoided if the cloud stack is identical to the customer's environment, and if the all code was developed as portable. But what if the cloud stack slightly differs, but the prices are very attractive? Then customers may need to spend resources for solution code porting.

We propose architecture for a generic stack extension that allows users and platform providers to:

- Smoothly integrate with the infrastructure
- Build stack-independent solutions
- Automate reconfiguration of the solutions

The architecture is based on the UBIWARE platform architecture and extends cloud platform services with the standardized configurable intelligent models.

4.1 Agent-driven Servicing in the Cloud

Interoperability is stated as one of the challenges of the cloud computing paradigm. We believe that adoption of the existing interoperability tools and solutions will become one of the major cloud computing research directions. Dummy platform API extension will just put the interoperability problem from the cloud provider to the client side. At the same time, the competitiveness of the cloud providers may depend on the simplicity of the integration with the client solutions and systems. Therefore, we foresee that embedded services offered by the cloud providers should be flexible and smart enough to handle client-specific model adjustments and configurations. We expand the understanding of the platform service to the smart proactive agent driven service. Such a service should not only be flexible and configurable in accordance with the customers' needs, it should also be prepared to resolve data- and API-level interoperability issues while being integrated with the client software.

Figure 3 shows the placement of the agent-driven extension in the cloud computing stack. From the user perspective the extension is still a service API but it offers an advanced functionality.

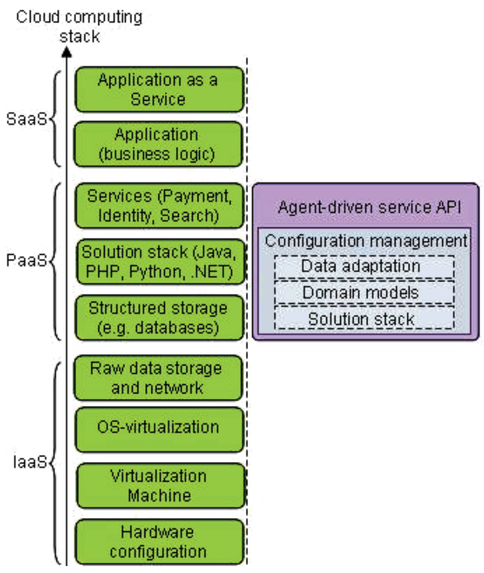

Figure 3: Placement of the agent-driven API.

The API shown above is a standalone middleware platform running either as a cloud facility, or embedded into the virtual machine instances as a platform extension. The detailed API content is shown of Figure 4.

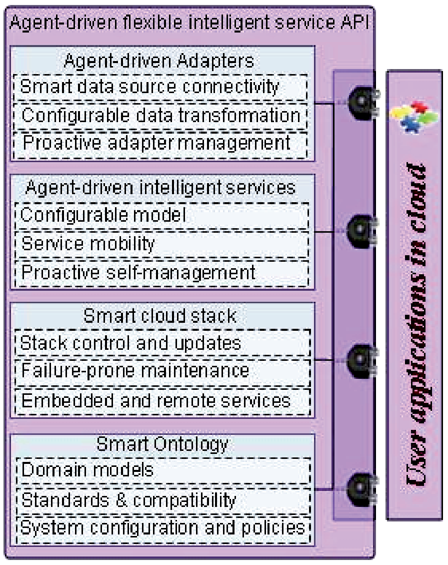
Agent-driven flexible intelligent service API
Agent-driven Adapters
Smart data source connectivity
Configurable data transformation
Proactive adapter management
Agent-driven intelligent services
Configurable model
Service mobility
Proactive self-management
Smart cloud stack
Stack control and updates
Failure-prone maintenance
Embedded and remote services
Smart Ontology
Domain models
Standards & compatibility
System configuration and policies
User applications in cloud

Figure 4: Agent-driven flexible platform service extension API.

The agent-driven adapters are software entities that facilitate data sources management. Adapters provide advanced data source connectivity functions (e.g. simplified database connectivity, file formats parsing, sensor data acquisition, etc.). Next, adapters handle the connectivity problem by providing the components for data transformation with configurable mapping functionality. The adapter becomes a proactive entity, i.e. it observes its state and takes actions based on the state and environment changes. The actions of the adapter may vary from simple fault messaging up to self-reconfiguration when an exceptional or fault situation occurs.

The services' API allows the user to run declarative models as services. The API provides a "model player API" for a particular domain-specific model definition language (the example of the API as well as the language will be discussed in the next Section). The model of the service being played is at the same time controlled by the dedicated agent that takes care of proper model functioning (e.g. load balancing and failures in the operation). Service agent may temporarily relocate the service executable code to another virtual machine instance to improve the performance in critical cases, thus the service becomes remote for its own original virtual machine instance. We also consider service API that has a local representative agent on each virtual machine, but the service execution is handled by the cloud provider (see Figure 5).

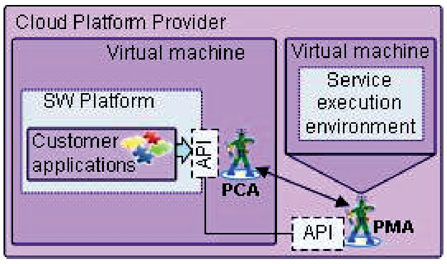

Figure 5: Service execution in cloud infrastructure.

In the Figure above the PCA stands for the Personal Customer Agent and PMA is a Platform Management Agent. The PCA may request the PMA agent to host a service execution (time period is a subject of contracting details) on a separate virtual machine to obtain e.g. higher performance, or for any other reason. At the same time the local API within the user's virtual machine and/or platform will stay the same. The PCA agent will wrap and forward local API calls to the PMA agent. The difference of the architecture proposed from the standard remote method invocation is a control channel between agents that allows the service management layer to stay separated from the service consumption (service calls).

In the next Section we discuss how the web services from the data mining domain can be integrated into the infrastructure described above. Data mining services can be embedded as platform services into the cloud stack for particular domain-specific cloud configurations at the same time preserving features of configurability, mobility and self-awareness.

4.2 Mastering Data Mining Services into the Cloud

To model the data mining services we have to define a corresponding data mining domain ontology. The ontology will cover data mining methods as well as requirements for method inputs and respective outputs. The inputs and outputs should, in turn, refer to the data types. The data mining domain can not include all possible applications of its methods; therefore we should keep the granularity of the conceptualization and distinguish the data mining models and their application scenarios. For the purpose of this scenario we take two data mining techniques: cluster analysis and k-NN method (classification).

The efforts towards standardization of data mining techniques, methods and formats have been a matter of discussion for the last ten years. One of the notable efforts is PMML language (PMML, 2009; Guazzelli et al., 2009). The language is a standard for XML documents which express instances of analytical models. In our work we take PMML as a reference model for the Data Mining Ontology and enhance both the model as well as the data with the semantic descriptions required to automate data mining methods application to the domain data. In this work we do not take into account the stage of information collection, preparation, etc. We assume that data is ready for data mining algorithm application. The PMML structure for model definition is composed of a set of elements that describe input data; model and outputs (Figure 6).

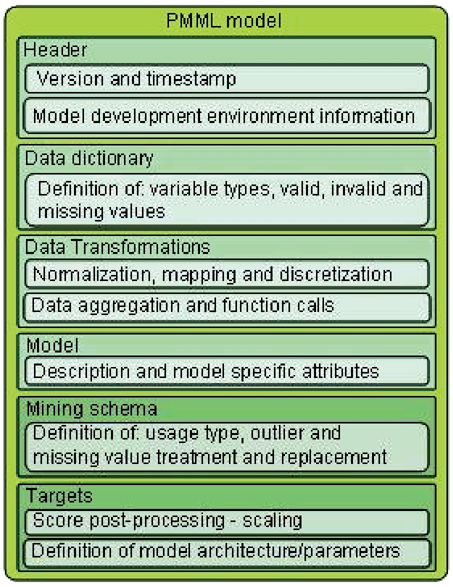
PMML model
Header
Version and timestamp
Model development environment information
Data dictionary
Definition of: variable types, valid, invalid and missing values
Data Transformations
Normalization, mapping and discretization
Data aggregation and function calls
Model
Description and model specific attributes
Mining schema
Definition of: usage type, outlier and missing value treatment and replacement
Targets
Score post-processing - scaling
Definition of model architecture/parameters

Figure 6: PMML model structure.

The PMML specification ver. 4.0 provides means for exhaustive model description, thus the model can be fully exported or imported without losses. Such model transportability gives huge opportunities for service orientation of the data mining methods. We also expect the PMML models reuse in the cloud computing domain in the nearest future. The specification of a software-independent descriptive data mining standard implies that Infrastructure and Platform layers of Cloud Computing stack are fully transparent for the data mining methods, i.e. the functional characteristics of the method-based services will be same for any stack configuration. The QoS, however, may vary depending on the performance of the hardware and efficiency of platform software, therefore the additional control channel over the service configuration may be needed.

We have identified three main types of data mining services regardless of their application domain and have introduced a classification of them (Figure 7).

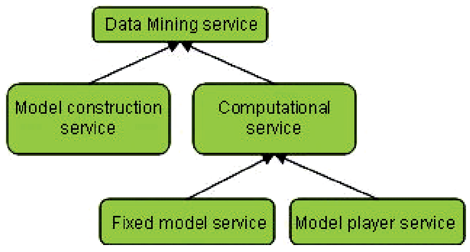

Figure 7: Upper ontology of data mining services.

We consider two major categories of data mining services:

- model construction services
- computational services

The model construction services produce a model (a semantic description) from the set of learning samples. In other words, input of such a service is learning data and conditions (for the neural network depending on its mode it can be a set of vectors plus e.g. initial network parameters). The output of the model construction service is a model with the parameters assigned (e.g. a neural network model, see Figure 8).

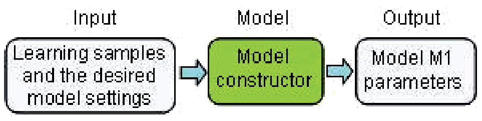

Figure 8: Model construction service.

The group of computational services can also be divided into two major groups:

- services with fixed model
- model player services

The services with fixed model define the format of the input and output as well as provide reference model description and the parameters that determine how the model is configured. For example, Figure 9 shows the definition of the neural network-based alarm classifier service for a paper machine.

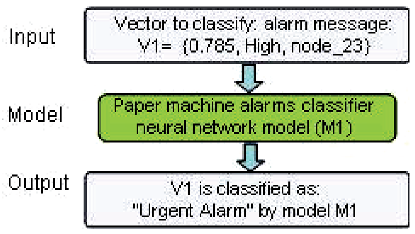
Input
Vector to classify: alarm message: V1= {0.785, High, node_23}
Model
Paper machine alarms classifier neural network model (M1)
Output
V1 is classified as: "Urgent Alarm" by model M1

Figure 9: Neural network model in a classification service.

Usage of model player services has two stages: in the first stage the service accepts the service model as an input, and in the second stage, it can serve as a fixed model service (see Figure 10).

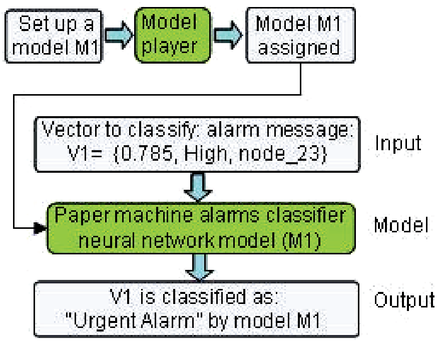
Set up a model M1
Model player
Model M1 assigned
Vector to classify: alarm message: V1= {0.785, High, node_23}
Input
Paper machine alarms classifier neural network model (M1)
Model
V1 is classified as: "Urgent Alarm" by model M1
Output

Figure 10: Model player service.

The true power of data mining services can be demonstrated in combination with the distributed querying (i.e. collecting learning or classification data), data mining model construction and further classification. The generic use case of such combination is shown on Figure 11.

The automated data collection process (first step in the use case) uses the Ontonuts technology and approach researched in the (Nikitin et al., 2009). The approach allows dynamic distributed query planning and execution, which we apply in this work to collect learning set data. The sources of the data may vary from databases, to csv-files and reside physically on different platforms and sites. The data collection and, hence, the querying implemented as a sequence of semantic data service calls orchestrated by a workflow management agent. Service orientation of data sources makes distributed querying a homogeneous part of other service-based workflow scenarios. The data collected (usually in form of a table of query results), may undergo preparatory steps to become a learning dataset. In this work we omit the procedure of normalization, or other data transformations, however, they will be necessary and obligatory. We assume that all operations with the data are also wrapped as semantic services.

As soon as the learning set is ready, a desired model constructor should be chosen (step 2). The model constructor may require specific data preparation, therefore it is good to combine data preparation step with the model constructor service.

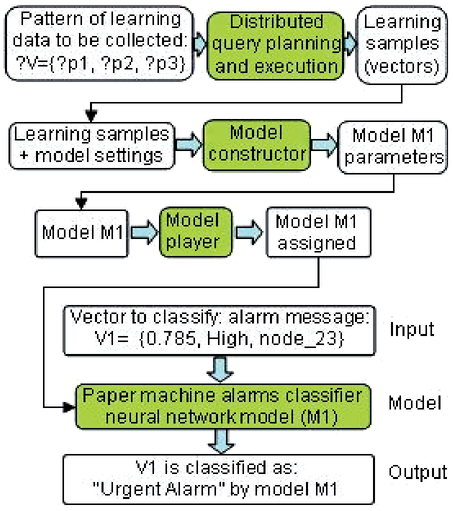
Pattern of learning data to be collected: ?V={?p1, ?p2, ?p3}
Distributed query planning and execution
Learning samples (vectors)
Learning samples + model settings
Model constructor
Model M1 parameters
Model M1
Model player
Model M1 assigned
Vector to classify: alarm message: V1= {0.785, High, node_23}
Input
Paper machine alarms classifier neural network model (M1)
Model
V1 is classified as: "Urgent Alarm" by model M1
Output

Figure 11: A use case scenario.

As an input, model constructor may require additional input parameters for model building. Those may be set as default, or, if other parameters were prepared, they should be supplied in the proper form. When a model is ready, we feed it to the model player service which is a platform service in terms of cloud computing, because it provides an infrastructure and software platform for service execution. As soon as our newly built model is deployed as a service, we can start classifying the data vectors, e.g. alarms coming from the paper machine.

The scenario described above may be dynamically reconfigured by the infrastructure agents. Some steps of the case (e.g. learning) may be temporarily moved to the separate execution environment (separate platform or virtual machine) to perform computationally expensive tasks.

The overall infrastructure of services should be highly proactive and responsive to the customer calls. Agents may monitor the execution and be ready to reconfigure their services in accordance with the current customer context.

5 RELATED WORK

The cloud platform solutions for business intelligence are gaining popularity. For example, SalesForce.com claims about 2 million success stories of its customers. The platform provides wide range of products (from infrastructure as a service, up to tailored web-based solutions for health care, retail and sales). The business intelligence tools are offered too. Nevertheless, the user has to adjust or prepare her/his software and data for the tools provided by the cloud. The advantage of the approach we offer is to empower any cloud platform with the intelligent adaptation mechanisms that would allow seamless data connectivity and integration. The architecture we offer is an extension to the cloud platform, not the platform itself. The data mining services with highly configurable parameters driven by the intelligent agents would simplify business intelligence integration, hence making adoption of cloud architecture easier for clients.

6 CONCLUSIONS

The research presented above describes specific application domain of intelligent services. We foresee that model player services will be a successful business case for the emerging paradigm of cloud computing. Pay-per-use principles combined with high computational capacities of cloud and standardized DM-models will be definitely an alternative to expensive business intelligence and statistics toolkits.

Another niche of data mining services in cloud computing can be model construction services. Such systems will drive innovations in data mining methods as well as applied data mining in certain domains. Such services will compete by introducing know-how and innovative tools and algorithms that bring add-values in e.g. predictive diagnostics or computational error estimation. This direction will lead to so-called “web intelligence” (Cercone et al.).

The role of UBIWARE platform in cloud computing emerges as a cross-cutting management and configuration glue for interoperability of future intelligent cloud services and client applications.

The main burden of UBIWARE will be management of consistency across different domain conceptualizations (Ontologies) and cross-domain middleware components. Fine-grained ontology modeling is still a challenge for research community and we predict that in the nearest future the domain modeling will be task-driven, i.e. the domain model engineers may incorporate some standardized and accepted conceptualizations, whereas the whole ontology for solution will be tailor made. Tailored ontologies will require subsequent mapping mechanisms and additional efforts.

The advanced data integration mechanisms embedded into the cloud platform as services is also an interesting concept that may become an add value for competing cloud platforms. The easiness of integration into the cloud infrastructure should not be underestimated especially by enterprise-sized companies, where business processes and component interdependencies have reached an unprecedented level of complexity. We believe that autonomy and self-awareness of building blocks will be a key to the future design of information systems and cloud platforms.

ACKNOWLEDGEMENTS

This research has been supported by the UBIWARE project, funded by TEKES, and the industrial consortium of Metso Automation, Fingrid and Inno-W. The preparation of this paper was partially funded by the COMAS graduate school.

REFERENCES

Bellifemine, F. L., Caire, G., and Greenwood, D., 2007. *Developing Multi-Agent Systems with JADE*. Wiley.

Cercone, N.; Lijun Hou; Keselj, V.; Aijun An; Naruedomkul, K.; Xiaohua Hu, 2002. "From computational intelligence to Web intelligence," Computer , vol.35, no.11, pp. 72-76, Nov 2002.

Foster, I.; Yong Zhao; Raicu, I.; Lu, S., 2008 "Cloud Computing and Grid Computing 360-Degree Compared," Grid Computing Environments Workshop GCE '08, vol., no., pp.1-10, 12-16 Nov. 2008.

Guazzelli, A., Zeller, M., Lin, W. and Williams, G., 2009. PMML: An Open Standard for Sharing Models. The R Journal, Volume 1/1, May 2009.

Hayes, B. 2008. Cloud computing. *Commun. ACM* 51, 7 (Jul. 2008), 9-11. DOI= http://doi.acm.org/10.1145/1364782.1364786

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

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

Nikitin S., Katasonov A., Terziyan V., 2009. Ontonuts: Reusable Semantic Components for Multi-Agent Systems, In: 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.

PMML, 2009. Data Mining Group. PMML version 4.0. WWW, URL http://www.dmg.org/pmml-v4-0.html

Rimal, B-P; Choi, E; Lumb, I, 2009, "A Taxonomy and Survey of Cloud Computing Systems," INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on , pp.44-51, 25-27 Aug. 2009.

Terziyan, V., 2003. 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, pp.279-291.

Terziyan, V., 2005. 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.

J Y V Ä S K Y L Ä   S T U D I E S   I N   C O M P U T I N G

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.