Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2006 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2006

Towards using Visual Process Models to Control Enterprise Systems Functionalities

Jens Weller
Technische Universität Dresden

Martin Juhrisch
Westfälische Wilhelms-Universität

Werner Esswein
Technische Universität Dresden

Towards using Visual Process Models to Control Enterprise Systems Functionalities

Jens Weller
Technische Universität Dresden
jens.weller@tu-dresden.de

Martin Juhrisch
Westfälische Wilhelms-Universität
juhrisch@uni-muenster.de

Werner Esswein
Technische Universität Dresden
werner.esswein@tu-dresden.de

ABSTRACT

In this paper, we propose a model-driven approach for web-service development with the intention of IT-Infrastructure integration in enterprise models. This implies the extension of existing conceptual modeling languages with language constructs that aim at model information transparency on a web-service basis. Therefore, we assess at the grammatical specification of conceptual modeling languages for business domains. Acting on language meta-level means in this context the introduction of web-service related constructs into widely used visual modeling grammars. Vision is the situation dependent adaptation of software functionality to the actual documented utilization. Economical relevance results from the reduction of significant discrepancies between enterprise systems functionalities and business requirements.

Keywords

Enterprise Architecture, visual modeling, web-service, enterprise models, model driven architecture, workflow.

INTRODUCTION

Because of public laws and increased competition today's companies have gained significant resources for quality management in the recent past. Simultaneously a strengthened process orientation is indicated within the whole commercial scope (Maier and Remus, 2001). Due to the intention of quality improvements but also to the increasing cost pressure through globalization, the focus of rationalization is more and more related to the efficiency and effectiveness of its business process structure (Kettinger, Teng and Guha, 1997). Prerequisites for an effective intervention into the process domain are in the first instance the entire analysis of the problem domain and its consistent documentation in the course of business process reengineering activities (Hammer and Champy, 1994). Within this scope of first stage analysis and documentation, visual models are established – so called enterprise models representing business requirements for the actual enterprise situation (Fettke and Loos, 2003). Corresponding modeling languages provide a graphical notation for high-level semantic description of real world phenomenon.

While enterprise models are an established technique for business analysis and development, the complexity of enterprises requires a more holistic view. Understanding enterprises as complex information systems, there are not only business aims and business processes but also software and data. For a successful development of an enterprise, it is necessary to integrate these different views. Enterprise architectures (EA) have been established to fulfill this task. The definition of EA varies (Beznosov, 2000) but is often understood as a framework that "…provides a way of viewing a system from many different perspectives and showing how they are all related." (Sowa and Zachman, 1992) Thereby, enterprise architecture should not been mixed up with software architectures, that are seen as the "…structure of the components of a program/system…" (Garlan and Perry, 1995).

While enterprise architectures can help understanding and managing the different perspectives, they generally do not solve problems arising when one view (or architectural layer) alters. In that case, the other views have to be adapted according to the dependencies existing between them. That means, when an enterprise model (e.g. process model) changes, the software supporting that process has to be tailored. That is usually carried out by functionality adaptation or by a complete new implementation. However, that will lead to exceptional effort, which is not possible to be justified in line with a reduced

budget caused by increasing cost pressure. Possible financial savings through convenient process improvements are thereby scattered due to the adaptation of associated application systems.

To avoid that manual software adaptation, we suggest an automated binding of functionality to the described enterprise models. Therefore, we propose a scenario, where business-supporting software directly accesses the enterprise models to adopt its behavior according to the described processes, organizational responsibility or whatever is stored within the enterprise models. We can identify two tasks for an appropriate realization of our vision:

1. Provide real-time access to relevant information of the enterprise models
2. Development of flexible business-supporting software accessing that information

To ensure real-time access to the enterprise models, the content of the models should be available electronically. Due to the raising usage of modeling tools, this is not a problem in modern enterprises. Thus, a connection between the business software and the modeling tool (or the modeling tool data) is required. For such a connection, the SOA paradigm (Erl, 2005; Stojanovic and Dahanayake, 2005) offers interesting methods of resolution. Due to their advantages and potentials in the present case, the focus lies on the W3C's web-service standards.

In this paper, we present a grammar to model such web-services. Thereby we do not concentrate on a general modeling approach, but focus on the enrichment of existing enterprise models with web-service elements. To realize a solution for enterprise models in different modeling languages, our approach aims at the meta-model level. In addition to the modeling approach, we present a framework that illustrates how such enriched enterprise models can be used for the scenario presented above.

The paper is structured as follows. In section 2, we give a short theoretical introduction into the specifics of enterprise models and web-services. Section 3 introduces a framework showing how enterprise and web-service models can control enterprise systems functionalities. In section 4, we introduce our web-service modeling approach embedded in the method engineering field. Section 5 summarizes consequences, recapitulates the proposed ideas and exposes open questions regarding the realization of the application integration.

BACKGROUND

Enterprise Modeling

To get an inside into specific problem domains the modeling approach is an important technique in system engineering. Essential benefit is attained through reduction of complexity by abstraction which facilitates analysis of complex systems (Balzert, 1994; Ferstl and Sinz, 1994). Models can be understand as the result of a construction "... done by a modeler, who examines the elements of a system for a specific purpose ..." (Schütte and Rotthowe, 1998). While conceptual models describe mental representations of real world phenomena with constructs representing both structural (e.g. things and their properties) as well as behavioral aspects (e.g. events and processes), design models represent software systems or parts of it (Evermann and Wand, 2005). Consequently a method to create conceptual models consists of a modeling grammar (language), providing a set of constructs and rules to combine the constructs and a procedure by which a grammar can be used (Wand and Weber, 2002).

For further scientific analysis, we will use enterprise models as synonyms to conceptual models. The construction of enterprise models often occurs within the requirements analysis phase during the information system development (Wand and Weber, 2002). Hence, the semantic mightiness of enterprise modeling language constructs has to cover non-formal aspects supporting a deep understanding of the business domain and of the potential of an information technology employment as well as formal aspects in order to support the system implementation (Frank, 1999). Thus, we need semi-formal languages to model problems which are not well-structured, highly subjective, individual and finally not objectively well formed (Harel and Rumpe, 2000).

However, most grammars do not possess a sufficient number of language constructs to model all phenomena in a domain (Agarwal, De and Sinha, 1999). Consequently, we might need a multitude of modeling languages to model the dynamic changing business domains (Wand and Weber, 2002). Therefore project or business situation dependent developed modeling methods have to be constructed adequately maybe only for one single project or only a short period of time (Brinkkemper, 1996). Much time and effort is spent on applying standard modeling methods effectively in such projects since they are often too general and include parts, which are not suitable for the requirements of a specific problem context (Brinkkemper, Saeki and Harmsen, 1998). The outcome of this is the need to adapt these methods or to construct them completely new.

With generic respectively situational method engineering, we mean the construction of project-specific methods from parts of existing method fragments (Brinkkemper 1996; Harmsen, Brinkkemper and Oei, 1994). Any model or combination of it can be used, their meta-level languages transformed into or compared with each other hence we distinguish between object and meta-level language (Frank, 1999). Software that supports generic method engineering is subsumed under the concept of meta-CASE tools like MetaEdit+ (Kelly, Rossi and Tolvanen, 2005) or cubetto toolset (Cubetto toolset, 2006). In the course of an increasing number of conceptual modeling grammars, meta-CASE tools are needed to support language independent modeling. Therefore, they require only the adaptation to a specific modeling grammar before the deployment. Hence, generic meta-CASE tools offer modeling support in various problem domains. As generic method engineering tools, they provide a meta-modeling language to extend or to simplify modeling languages as well as to create new ones.

Web Service Technology

The paradigm of Software Orientated Architecture (SOA) provides the basis for the present distributed application framework (W3C, 2003). Software components are provided in modular and reusable services. In this context we speak about loose coupling and place independency of participating services since there are no strong logical or physical dependencies between services and involved applications (Knuth, 2003).

This paper focuses on a web-service based realization of the SOA paradigm according to the recommendations of the W3C consortium (Erl, 2005). The intension of the web-service concept in the present paper derives from the following allocated criteria:

- An abstract interface for embedding the web-service in high level specifications like the Business Process Execution Language for Web Services (BPEL4WS; IBM, 2002), to define which messages and data types the web-service understands and in what sequence they will be processed.
- The connection to a concrete transmission protocol as the basis for communication.
- Finally the service itself with an address and the implementation of its functionality.

W3C's layered architecture for today's web-service technology focuses on three principle core elements described in detail by Muschamp (Muschamp, 2004). The Simple Object Access Protocol, the Web-Service Description Language (WSDL) (Gudgin, Lewis and Schlimmer, 2004) as well as the Universal Description, Discovery and Integration Language (UDDI; Walsh, 2002) have become de facto standards for XML messaging, web-service description and registration. In addition to these main protocols, web-service composition requires higher levels of description. In the course of this demand orchestration and choreography languages base upon several high level standards like business BPEL4WS, WSFL (IBM, 2001), Web for business process design (XLANG; Ardissono, Goy and Petrone, 2003) or Web Service choreography interface (WSCI; Arkin, Askary, Fordin, Jekeli, Kawaguchi, Orchard, Pogliani, Riemer, Struble, Takaci-Nagy, Trickovic and Zimek, 2002). Compared to the core protocols this high level languages take a step forward by integrating web-services in business process models. Essence is the integration of business processes across enterprise's boundaries by modeling web-services in directed graphs in the order of their chronological sequence. One of the fundamental characteristics of SOA is the separation between service interface and implementation (Fremantle, Weerawarna and Khalaf, 2002). As a result, the business logic of an application system can be separated from its implementation- and infrastructure details, too. Hence, the whole process within a web-service based architecture can be modeled as a process model with involved flows, states and activities.

Modeling languages in this area derive from techniques out of the business process modeling scope. Some patterns are reused for instance to map the process of sending and receiving messages in terms of activities in a business process flow diagram. Besides the interface behavior the composition of web-services is another modeling aspect concerning in which manner web-services complements one another with the objective to model service-based business processes whereas major process parts are encapsulated as web-services communicating over organization boundaries. Integration between web-services and modeling languages arises from the objective to choose utilized web-services dependent from the actual business situation. We can differentiate between two kinds of composition modeling. In this context, we speak about local or global view meaning to limit the description to one single web-service or rather to consider all web-services contained in the system.

The local view contains all behavioral information of one web-service including control, data flow and all changes in variable states. Behavior of partner applications and accordingly of other connected web-services is not considered. This view is only interested in which functionality is carried out for the analyzed service. Is a web-service described from local view it can be scanned regarding all relevant aspects. In contrast to the global the local view does not offer any clear link to conceptual modeling in our problem domain.

The global view involves all web-services contained in a system abstaining from all internal process logic. The emphasis lays on the modeling of interaction of all participating web-services, the localization of discrepancies in web-service behavior descriptions and on the analysis of substitution capabilities between web-services of one similar domain. Languages in this area are also called choreography languages (Busi, Gorrieri, Guidi, Lucchi and Zavattaro, 2005) offering constructs to describe the interaction and the process flow of two or more web-services aiming at one common goal. Regarding conceptual models, web-service modeling with these high-level standards takes place in process flow models mapping business processes including web-services. Language products are executable business process models in XML notation. An intersection between existing composition languages and the present scope could not be found as XML tags are not sufficient for a semantic description in a conceptual modeling language. Thus, ideas for a conceptualization of a web-service stay constricted to the core layers of W3C's web-service architecture.

INTEGRATION FRAMEWORK

To sum up the recent section we can say that the business orientation is the main difference to previous design- or integration approaches within the software engineering discipline (Busi et al., 2005). However, in contrast to the orchestration of web-services to executable business processes, the following approach aims at a model-driven control of a dynamic IT-Infrastructure.

As a result of application centered IT-architectures the functions of program systems are highly integrated and particular functionality cannot easily been taken out. Processes are normally supported within an application but are restricted to the application or end at system borders. Hence, business process alignment is currently reached with attached workflow management systems (WfMS) integrating applications via complex data interchanges. Alternatively a dynamic IT-Infrastructure can be realized by direct program to program interaction (SOA) exemplary illustrated in figure 1. The essence of our approach is now to reach Business Agility through integrating IT-Systems into enterprise models by using a generic web-service technology. Particularly, the generic of engaged web-services allows us to change the way of integration between infrastructure and enterprise models absolutely flexible. This requires additionally to the description of system requirements during the modeling of business applications, a model of web-services, describing the services within the web-service, that are in turn used by the applications.

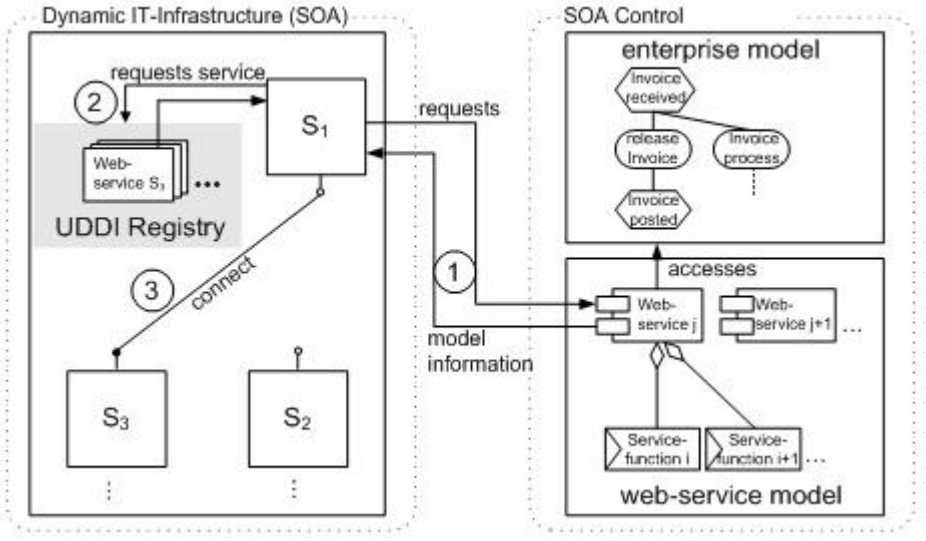

Figure 1. Framework for dynamic adaptation

A possible scenario could start in a first step (1) with a request of an autonomous web-service S_1 to receive enterprise model information from a modeling tool. For instance, S_1 is interested in the next process step within the enterprise model. Appropriate information would be the name of the activity "release invoice" in our EPK process model. As illustrated, the SOA infrastructure is organized over a central UDDI registry with references to all available web-services. Depending on the received model information from web-service S_1 the UDDI registry responds with reference information to web-service S_3. Having all interface information downloaded web-service S_1 can finally establish its communication with web-service S_3 (3).

In this paper, we concentrate on the model-driven way of developing generic web-services that make enterprise models transparent to infrastructure requests.

MODELING APPROACH

Overview

With regard to the integration problem at hand, existing approaches in the area of model-driven web-services are not sufficient. Their background is confined to the domain of web-service composition in business process models. To overcome these limitations, we require a more common, more specific description of web-services.

The following section extends an existing method of the method-engineering discipline (E³; Greiffenberg, 2004) to provide the necessary modeling grammar in order that both meta- and web-service model can be entirely described. The E³-Model is classified as meta-meta-model on M3-Level of the Meta-Object-Facility Architecture (Object Management Group, 2002). Our approach is based upon the extension of a meta-modeling method in order to develop generic web-services on meta-level. Thus, we introduce the E³+WS method that on the one hand allows to describe a web-service based on meta-model information and on the other hand prepares the implementation of its operations. In this paper, we focus on the interface modeling with E³+WS. We work on a model-driven realization of web-service functionality. The implementation of web-service operations requires a model-driven technique to combine the referenced API functions.

Matter of the present method engineering are web-services providing information in terms of modeled issues in a conceptual model. Therefore, web-service related constructs will be integrated into the meta-model. Hence, the objective of the method engineering is to develop another method-engineering method. Consequential the present domain can be illustrated as in figure 2.

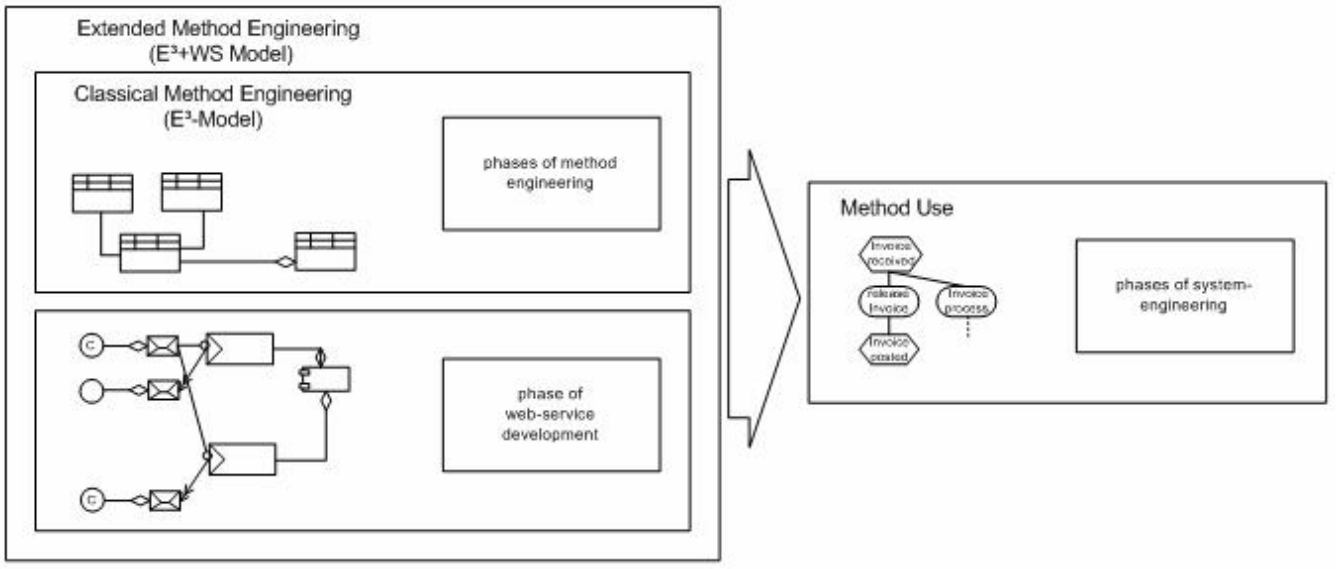

Figure 2. Method engineering with E³+WS

By separating existing methods into method fragments we can generically rebuild new ones (Harmsen et al., 1994). In the current case, we use as existing method-fragment the meta-modeling language E³. Similarly, while WSDL describes the input/output format of a web-service, we introduce a language dependent modeling of web-services. Language dependent stands for web-service development on meta-model layer since relevant model information are characterized by significant patterns in the meta-model. Every web-service described by the proposed technique has in this case unlimited functionality for every enterprise model that is formulated by the referenced meta-model.

Web-service modeling language

Abstracting from a determined notation, first step is the development of a language-based meta-model containing, in our case, all constructs to model a meta-model, a web-service interface and its implementation. In dividing the meta-model of E³+WS in separate views, we can simplify the process of construction. First, we turn our attention to the construction of the meta-model that we want to extend. We assume that all contained constructs and relationships between them are described over the E³-Model language conventions outlined in (Greiffenberg, 2004) within an eE³-View (extended E³-View).

Starting from a constructed meta-model we generate web-service descriptions within a separate web-service view (see again figure 2). Thus, we generate separately a modeling language that on the one hand is able to express WSDL according to the W3C standard and on the other hand can be understood and adopted by target group users of the E³+WS method. In the course of a necessary parallelism with the W3C's standards, our approach possesses a certain affinity to the vocabulary of WSDL.

Centric element is the web-service. Within a web-service-view, a web-service is constructed in two ways. The abstract definition takes place by adding its functionality and a unique name. Functionality is mapped through the aggregation of service-functions to web-services while the designator acts on the one hand as an ordinary identifier but also possesses a descriptive character by declaring the semantic of the web-service. A web-service element can be composed out of one or more service-functions by aggregation edges. Thus, web-services correspond to a number of similar service-functions. Furthermore, a service-function consists of an appropriate parameter assignment. Therefore, we define special message objects that act as containers for input respectively output parameter objects. Messages are connected with service-functions over so called messageEdge_in or messageEdge_out constructs depending on the role of the message regarding their related service-functions. Thereby a service-function can be connected to maximum one messageEdge_in and always exactly one messageEdge_out. Hence, it is assumed that the possible service-functions communicate either with a request-response or with a notification pattern.

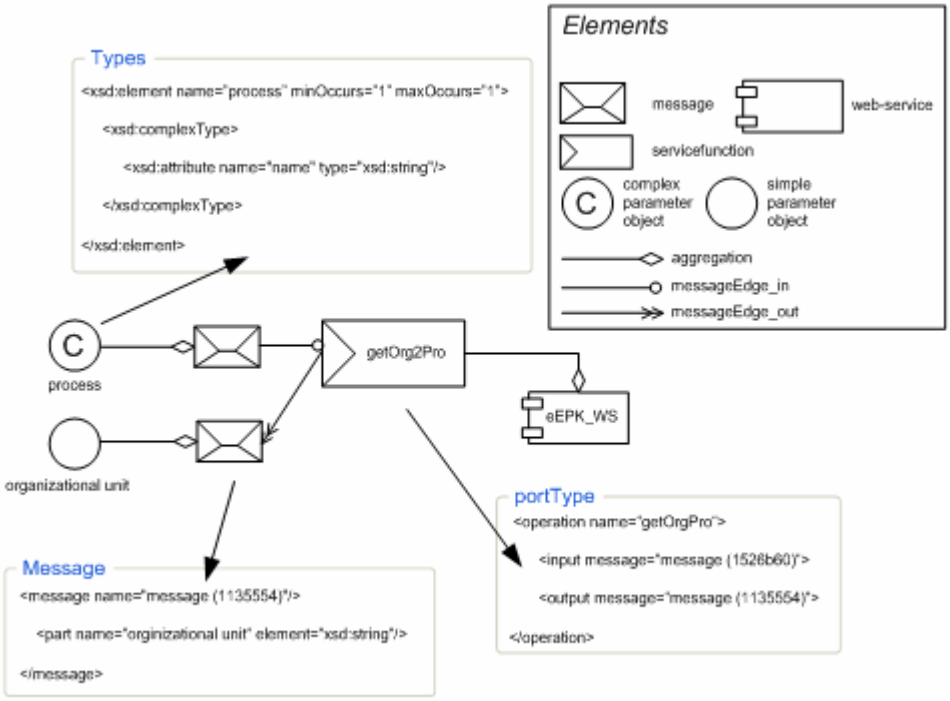

Figure 3. Web-service model describing interface information

To define a single message parameter, we introduce the construct parameter object dividable into simple or complex objects. Simple parameter objects are assigned to simple data types. These types comprises on the one hand predefined XML schema data types and on the other hand own data types declared within the WSDL types tag. With self declared data types we are able to define for example appropriate report formats as potential result layouts for a model migration operation. A complex

parameter object corresponds to design activities of a method engineer. Therefore, an E³-data type is introduced to establish ties between complex parameter objects and meta-model patterns. In the first move, we only tie E³ object types down to complex parameter objects. Message element and parameter object are analogical to service-function and web-service element connected over an aggregation edge. Figure 3 illustrates an exemplary web-service model including one service-function. The attached WSDL-Code shows related interface information generated out of the web-service model.

Different user perspectives

Although different conceptual languages are able to express requirements and behavioral models of target systems (Bézivin, Hammoudi, Lopes and Jouault, 2004; Patrascoiu, 2004; Thoene, Depke and Engels, 2002), no existing language supports integration and sharing of model information with IT applications through web-services of enterprise models. However, this is necessary to implement the possible adaptation processes already before the analysis phase of the system engineering. The property of web-services to be easily specifiable with formal design languages, leads to an evaluation of mapping possibilities from formal concepts of web-service design to the meta-level of conceptual modeling. In consideration of the outlined deficits we propose in this paper a semantically enriched meta-model for conceptual modeling. Figure 4 illustrates the participating roles in our modeling method.

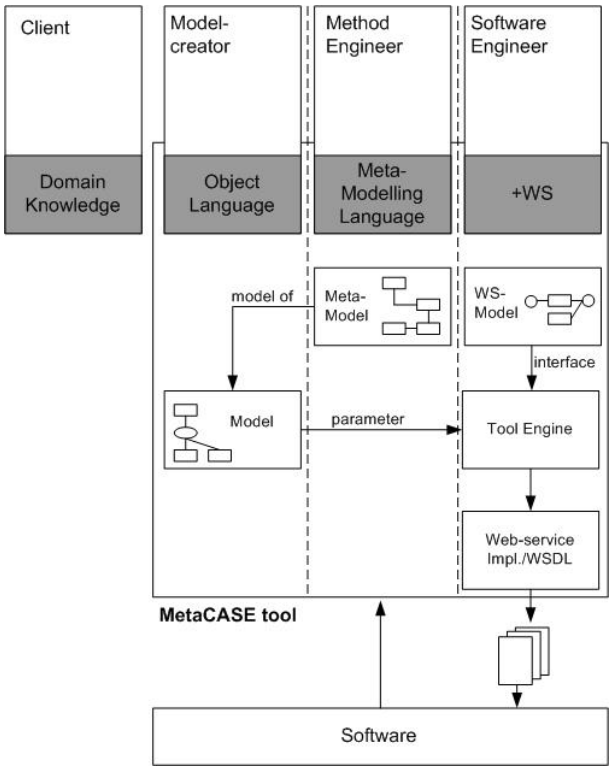

Figure 4. Role Model for software adaptation

The main requirement for this framework represents the avoidance of any restriction to the ordinary modeling task on object level. Both domain (clients) and modeling experts (model creator) should not notice that their input into an enterprise model is made visible. Usually the construction of enterprise models requires "a specific competence that is neither covered by software engineering nor by organization science" (Frank, 1999) so implementation related information have to be kept away. Additionally the integration of implementation aspects after the analysis phase of the system engineering would not facilitate the adaptation process of enterprise systems functionality. Furthermore, cost reduction potentials go astray.

We suggest, that only the software engineer deals with generic web-service modeling in relation to a meta-model as a result of a method engineering task. At the moment the proposed modeling grammar offers a sufficient set of constructs to model any generic web-service interface in relation to a given meta-model.

DISCUSSION

The question about why to develop web-services in visual models is equivalent to the question about the motive for introducing OMG's Model Driven Architecture (MDA; Object Management Group, 2003) in the earlier stages of software engineering in general. Visual modeling together with model transformations play the key roles in MDA which is an approach that's focuses on IT system specification separating system functionality specification from the specification of the implementation of that functionality.

Since we think that the MDA idea for web-services can partly be extended also for the requirements analysis, this paper introduced a grammar for model-driven web-service development. E³+WS describe the structure and functions of web-services while abstracting away technical details. Our method complies also with the requirement to integrate web-service concepts in conceptual models. Our approach is however, restricted to the development of web-services for model information transmission. Web-service description in E³+WS is limited to meta-models respectively meta-model elements.

Until now, a first implementation is available for the meta-CASE tool cubetto toolset. Thereby, we implemented the introduced modeling language, that enables the user to model web-services and connect their service parameter to conceptual modeling constructs. Additionally, it is already possible to create an xml file out of the web-service model containing the WSDL description. The implementation of a model-server for the automatic creation of associated web-services is in progress.

REFERENCES

1. Agarwal, R., De, P., Sinha, A. P. (1999): Comprehending object and process models: An empirical study, *IEEE Transactions on Software Engineering*, 25, 4, 541-556.
2. Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takaci-Nagy, P., Trickovic, I., Zimek, S. (2002): Web Service Choreography Interface (WSCI) 1.0. http://wwws.sun.com/software/xml/developers/wsci/.
3. Ardissono, L., Goy, A., Petrone, G. (2003): Enabling conversations with web services, in AAMAS'03, *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, Melbourne, Australia, ACM Press, 819-826.
4. Balzert, H. (1994): Development of Software-Systems: Principles, methods, languages, tools. BI, Mannheim.
5. Bézivin, J., Hammoudi, S., Lopes, D., Jouault, F. (2004): Applying MDA Approach for Web Service Platform, in EDOC'04: *8th IEEE International Enterprise Distributed Object Computing Conference*, ACM Press.
6. Beznosov, K. (2000): Information Enterprise Architectures: Problems and Perspectives. Technical report, School of Computer Science, Florida International University, Miami.
7. Brinkkemper, S., Saeki, M., Harmsen, F. (1998): Assembly Techniques for Method Engineering, in B. Pernici, C. Thanos (Eds.) *CAiSE'98*, LNCS 1413, 381-400.
8. Brinkkemper, S. (1996): Method Engineering: Engineering of Information Systems Development Methods and Tools, *Journal of Information and Software Technology*.
9. Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., Zavattaro, G. (2005): Choreography and orchestration: A synergic approach for system design, in ISSOC'05: *Lecture notes in computer science / service-oriented computing*, 228-240.
10. Cubetto toolset (2006): http://www.semture.de/cubetto.
11. Erl, T. (2005): Service-oriented architecture: concepts, technology, and design, Prentice Hall PTR.
12. Evermann, J., Wand, Y. (2005): Ontology based object-oriented domain modelling: fundamental concepts, *Requirements Engineering*, 10, 2, 146-160.
13. Ferstl, O. K., Sinz, E. J. (1994): Fundamentals of Information Systems, 1, R. Oldenbourg, Munich, Vienna.
14. Fettke, P., Loos, P. (2003): Classification of reference models: a methodology and its application, *Information Systems and e-Business Management*, 1, 1, 35-53.
15. Frank, U. (1999): Conceptual Modelling as the Core of the Information Systems Discipline – Perspectives and Epistemological Challenges, in D.W. Haseman, D. Nazareth, D. Goodhue (Ed.) *Proceedings of the Fifth America's Conference on Information Systems, AMCIS'99*, AIS, Milwaukee, 695-697.
16. Fremantle, P., Weerawarna, S., Khalaf, R. (2002): Enterprise Services – Examining the emerging field of Web Services and how it is integrated into existing enterprise infrastructures, *Communications of the ACM*, 45, 10, 77-82.

17. Garlan, D, Perry, D. (1995): Introduction to the Special Issue on Software Architecture. *IEEE Transactions on Software Engineering*, 21, 4, 269-274.
18. Greiffenberg, S. (2004): Method Engineering in Business and Government, Dr. Kovac, Hamburg.
19. Gudgin, M., Lewis, A., Schlimmer, J. (2004): Web Services Description Language (WSDL) Version 2.0 Part 2: Message Exchange Patterns. W3C Working Draft, World Wide Web Consortium, Boston.
20. Hammer, M., Champy, J. (1994): Reengineering the Corporation: A Manifesto for Business Revolution, HarperBusiness, New York.
21. Harel, D., Rumpe, B. (2000): Modeling languages: Syntax, semantics and all that stuff, part I: The basic stuff. http://wisdom.weizmann.ac.il/archive/00000071/01/00-16.ps, Technical Report msc00-16, Weizmann Institute Of Science.
22. Harmsen, F., Brinkkemper, S., Oei, H. (1994): Situational Method Engineering for Information System Projects, in T.W. Olle, A.A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle, *Proceedings of the IFIP WG8.1 Working Conference CRIS'94*, North-Holland, Amsterdam, 169-194.
23. IBM (2001): Web Service Flow Language, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.
24. IBM (2002): Business Process Execution Language for Web Services Version 1.1. http://www.ibm.com/developerworks/library /ws-bpel.
25. Kelly, S., Rossi, M., Tolvanen, J.P.(2005): What is Needed in a MetaCASE Environment?, in Ulrich Frank (Ed.) *Enterprise Modelling and Information Systems Architectures*, October, 22-35.
26. Kettinger, W., Teng, J., Guha, S. (1997) Business Process Change: A Study of Methodologies, Techniques, and Tools. *MIS Quarterly* 21, 1, 55-98.
27. Knuth, M. (2003): Web Services. Introduction and Overview, 2, Frankfurt:S&S.
28. Maier, R., Remus, U. (2001): Towards a Framework for Knowledge Management Strategies: Process Orientation as Strategic Starting Point, *Proceedings of the 34th Hawaii International Conference on System Sciences*.
29. Muschamp, P. (2004): An introduction to Web Services, *BT Technology Journal*, 22, 1, 9-18.
30. Object Management Group (2002): Meta Object Facility (MOF) Specification, version 1.4, April.
31. Object Management Group (2003): MDA Guide, version 1.0.1, June.
32. Patrascoiu, O. (2004): Mapping EDOC to Web Services using YATL, in *Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference*, ACM Press.
33. Schütte, R., Rotthowe, T. (1998): The guidelines of modeling: An approach to enhance the quality in information models, *Lecture Notes in Computer Science* 1507, 240-254.
34. Sowa, J., Zachman, J. (1992): Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31, 3, 590-616.
35. Stojanovic, Z., Dahanayake, A. (2005): Service-Oriented Software System Engineering: Challenges and Practices, Idea Group Inc.
36. Thoene, S., Depke, R., Engels, G. (2002): Process-Oriented Flexible Composition of Web Services with UML, in eCOMO'02: *Proceedings of the Joint Workshop on Conceptual Modeling Approaches for e-Business*.
37. W3C (2003): http://www.w3.org/TR/2003/WD-ws-arch-20030808/wsa.pdf.
38. Walsh, A.E. (2002): Uddi, Soap, and WSDL: The Web Services Specification Reference Book, Prentice Hall Professional Technical Reference, New Jersey.
39. Wand, Y., Weber, R. (2002): Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda, *Information System Research*, 13, 4, 363-376.