Modeling Semantic Web Services using UML 2

Florian Lautenbacher and Bernhard Bauer

Programming Distributed Systems Lab
Institute of Computer Science, University of Augsburg, Germany

Abstract: The development of web services and especially semantic web services is a tremendous task for people who are not familiar with all language constructs. Especially with the growing mass of several semantic web service approaches a language-independent service development is needed. This gap can be filled by using model-driven software development. Therefore, in this paper we introduce a meta-model for semantic web services and show the benefits of automatic code generation on the basis of a small example.

1 Introduction and Motivation

Globalization and the new possibilities the internet provides have resulted in the need to define more loosely coupled components in distributed heterogeneous environments. Service Oriented Architecture (SOA) represents an approach that facilitates this loose coupling while at the same time providing sufficient qualities of services which are necessary for acceptable solutions. Therefore, many companies and research institutes have standardization efforts for web services (which are one way to realize an SOA). On the other side this results in a non-manageable amount of standards where each standard covers different parts than the others (see e.g. [1] for a short overview). But the large quantity of different 'standards' and the heterogeneity of platforms intensify the problem of interoperability. Annotating web services with semantic information, e.g. from an ontology, can help to resolve these shortcomings.

Future protection of investment makes companies unwilling to change from one standard to another from time to time. They need the possibility to easily adapt their companies to upcoming standards. This already begins with the design of company goals and business processes. Companies more and more demand a stable IT environment from business processes to executable services in order to improve the interoperability with other enterprises.

The Object Management Group (OMG) therefore promotes the model-driven architecture (MDA, [2]) approach towards the analysis, design and implementation of systems. The primary goals of MDA are portability, interoperability and reusability through an architectural separation of concerns between the specification and implementation of software. One of the key aspects is the usage of the Unified Modeling Language (UML, [3]) to model all kind of aspects. UML2 supports with its well-defined meta-model (building on the Meta Object Facility – MOF, [4]) model transformation and code generation. Using the MDA one can e.g. model (semantically enhanced) business processes and transform them to an SOA-profile for semantic web services which enables the code generation of executable web services.

Therefore we recognized the need to develop a UML-profile and meta-model for semantic web services (called MM4SWS in the future) and to define transformation rules to automatically generate code in languages like OWL-S, WSMO, WSDL-S, SWSF or SAWSDL. The advantage of using a UML profile is clear: a service can be modeled and code be generated by using one of the existing standards and if it is foreseeable that another standard becomes accepted the code can simply be generated by using the other language, too. Thus, the increasing investment risk can be prevented and a persistent IT environment from (semantic) business process models over semantic web services models to executable code is ensured.

This paper is structured as follows: In the next section we introduce our meta-model MM4SWS and describe the concepts based on the existing semantic web service approaches. In section 3 we present an example how our meta-model and UML-profile can be used. An overview about existing approaches for modeling semantic web services will be shown in section 4, before we conclude.

2 A Meta-Model for Semantic Web Services

Recognizing the need for an independent way of modeling semantic web services, we designed a meta-model for semantic web services (MM4SWS) which can be transformed into the current W3C submissions and working drafts. Therefore, we analysed the current approaches, evaluated their similarities and differences and developed a meta-model which covers all details needed in the different standards. Thus, we considered ODM [5] for the modeling of the underlying ontologies and built on current W3C semantic web service submissions named OWL-S [6], WSDL-S [7], WSMO [8] and SWSF [9] and the candidate recommendation SAWSDL [10]. A detailed description of these standards is out of scope of this paper and can for example be found on [11]. MM4SWS integrates all parts of these approaches (many concepts are similar or equal in different standards besides their names) and one can generate code for all of them using our meta-model.

In [12] five key workflow aspects have been identified, namely the functional, behavioural, informational, organisational and operational perspective. Since semantic web services are often used to execute a specific workflow we categorized MM4SWS similarly: our profile consists of five packages which interact with each other. The Ontology-package contains all concepts (organisational aspects) that are needed to model an ontology. The Interfaces-package provides all elements to model a WSDL service and to describe it with semantics (like in WSDL-S). The ServiceProvider-package includes all aspects to model one or more semantic web services with non-functional descriptions and using the elements of the ProcessFlow-package the behavioural elements and composite services can be modeled. Every action can be annotated with functional descriptions as described in the Functional-package.

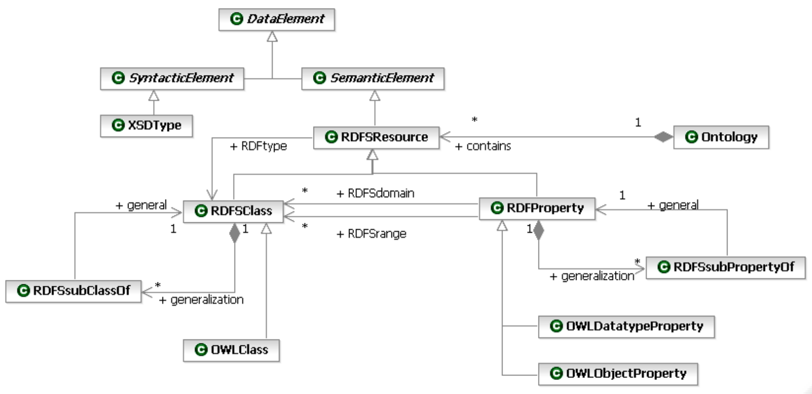


Fig. 1. SWS meta-model for the ontology.

At the bottom of MM4SWS are elements to model the constructs in an ontology[1]. To model all kind of syntactic and semantic data, the top-level element of our ontology-package (compare Fig. 1) is a *DataElement*. A *DataElement* can be a *SyntacticElement* (e.g. an element specified in an XSD-file: *XSDType*) or it can be a *SemanticElement*. This might be (referring to ODM) an *RDFSResource* which is part of an ontology. A resource can either be a class (or concept as it is called in WSML) or a property. *RDFSClass*es can be connected to other classes via *RDFProperties* which contain the domain and range of the property. Every class (resp. property) can be generalized and there exist specializations for OWL classes which are the mostly used presentation form of concepts in current ontologies.

The *DataElement* and *XSDType* are necessities from OWL-S where it should be possible to reference not only semantic elements as input or output, but also elements which are not semantically described. The other classes are directly adapted from ODM, where some of the classes in ODM have been neglected for the sake of simplicity.

Each web service (as defined in WSDL) has an *Interface* (cf. Fig. 2) which includes a number of operations. An *Operation* is callable from other web services. Every operation can have one input and output *Message* and zero or more fault messages. These messages are exchanged through channels between at least two services. Each operation might be described with a semantic element of the ontology. A message contains one or more *Document*s which can also be described with semantic elements and which can be structured with several *Attribute*s which might themselves be documents again or simple data types (like String, int, etc.). *Interfaces*, *Operations*, *Messages*, *Documents* and *Attributes* are needed to generate WSDL code, the dependencies to the *SemanticElements* have their origin in the WSDL-S and SAWSDL approach.

With these basic constructs for describing the interfaces, operations, messages and the underlying ontology we can now start the modeling of one or more semantic web services.

[1] whereby it doesn't matter whether this ontology should be in OWL or in WSML since both can be generated

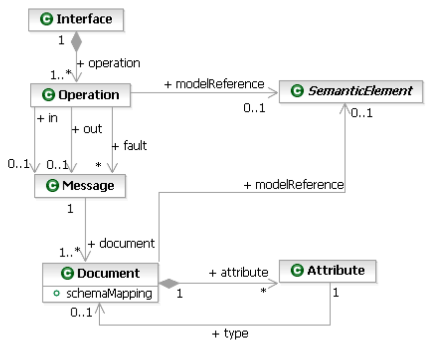


Fig. 2. SWS meta-model for interfaces, operations and messages.

A web service is a *Process* with a specified behavior. A *ServiceProvider* (e.g. an institution, person or organization) offers a number of processes which can be executed. Every process can be categorized (according to OWL-S and SWSF) with a *categoryName*, a path to a *taxonomy*, a specific *value* in the taxonomy and the *code* associated to a taxonomy. Each processs can be additionally described with a *ServiceDescriptor* which includes a serviceName (name), textDescription (description) and contactInformation (contact_info) as needed in OWL-S and additional information which is needed in SWSF such as the author of the service, the version, releasedate, etc.

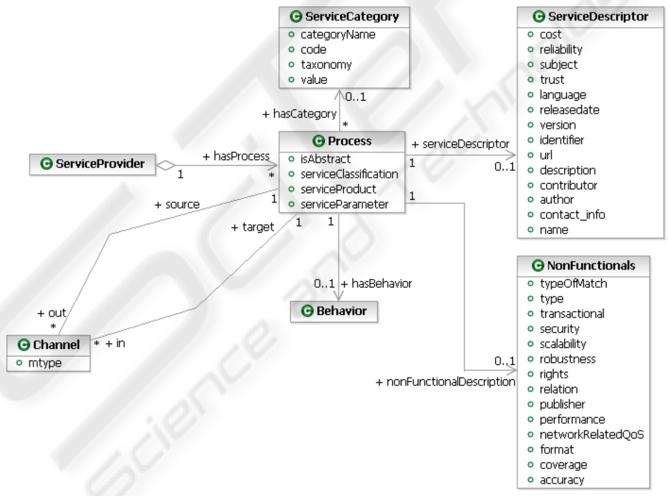


Fig. 3. SWS meta-model for a service provider.

Some more non-functional descriptions are defined in the WSMO-approach which are covered in the properties of the class NonFunctionals. The whole class and properties of *NonFunctionals* can be neglected, if the user models a semantic web service and is sure that he does not want to generate WSMO code in the future. Each process can communicate with other processes via *Channel*s (a requirement of SWSF) where the above introduced messages are exchanged.

Each process has an internal behavior which can be described using a *Process-Flow*. A process flow consists of *Node*s and *Connection*s between these nodes. A node might either be a control node like the ones known in UML activity diagrams (InitialNode, ForkNode, JoinNode, DecisionNode, MergeNode, etc.) or an *Action* or a *CompositeProcess* which consists of several other nodes. A *CompositeProcess* describes a composite service whereas a ProcessFlow, with only one node, specifies an atomic process. An *Action* can be either an *AtomicProcess* which calls an operation or a *CallCompositeProcess* which can start a new behavior or a *CompositeProcess*.

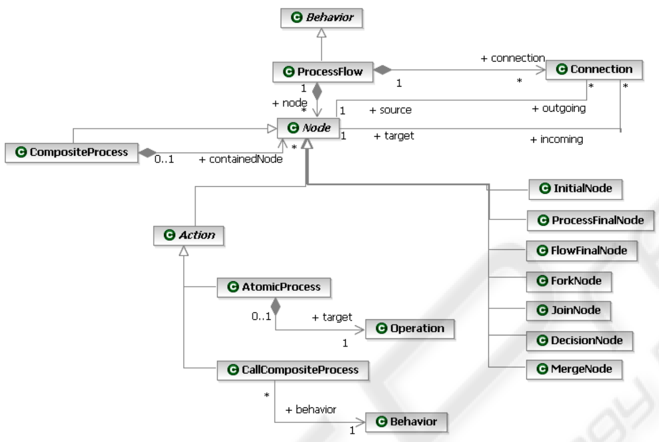


Fig. 4. SWS meta-model for the process flow.

Each action might require *Input*s and produces *Output*s (see Fig. 5) which can be described more abstract as *DataElement*s as introduced above. Every action can also be described with its *Precondition*s and *Effect*s, which themselves can be described with a class or concept of the ontology. The *Preconditions* describe the necessary information state and state of the world before an execution of the action is possible, the *Effects* show how the state of the world and the information state have changed after executing the action.

A more detailed description of MM4SWS, the developed UML-profile including transformation rules and examples can be found in [11].

It is difficult to integrate MM4SWS into the layers of MDA: in principal it is platform independent, but also includes constructs for each specific platform and SWS language and code can directly be generated from MM4SWS. But, this matches to our own experiences that business users prefer a single model, avoid model transformations where possible and use different views on a model instead.

3 Overall Example

As an overall example we shortly introduce a well-known example that has been created within the OWL-S specification.

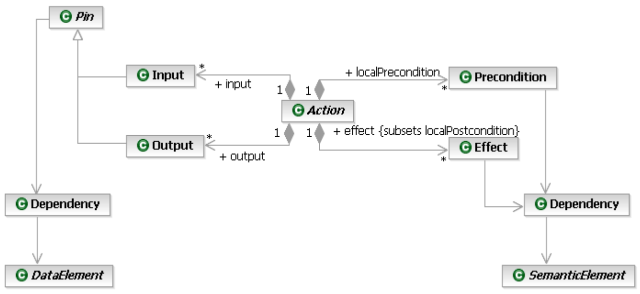
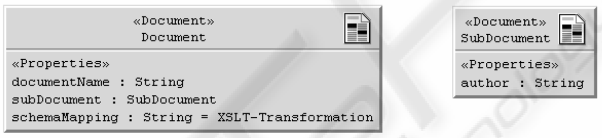


Fig. 5. SWS meta-model for the functional description of an action.

The example is called CongoBuy and is a B2C bookbuying example showing the OWL-S usage, illustrating a simple use of the process model. The service described is a fictional book-buying-service from www.congo.com. The example is divided in two parts: ExpressCongoBuy as a very small example with only one web service invocation (atomic action) and FullCongoBuy as a composite web service. Our UML-profile has been implemented in a UML tool-suite and we specified informal transformation rules to generate code from the meta-model and implemented these rules using the openArchitectureWare-language XPand [13].

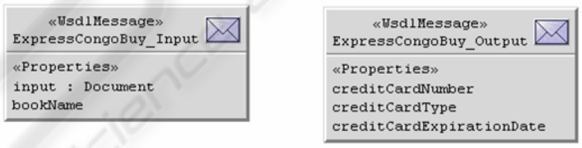


```
«Document»
Document
«Properties»
documentName : String
subDocument : SubDocument
schemaMapping : String = XSLT-Transformation

«Document»
SubDocument
«Properties»
author : String
```

Fig. 6. Example: Documents.

After the modeling of the ontology (similar to ODM) all documents which are needed to exchange data between processes need to be created. In the CongoBuy-example no documents have been specified, therefore we only show the usage of two simple documents which have been nested.



```
«WsdlMessage»
ExpressCongoBuy_Input
«Properties»
input : Document
bookName

«WsdlMessage»
ExpressCongoBuy_Output
«Properties»
creditCardNumber
creditCardType
creditCardExpirationDate
```

Fig. 7. Example: Messages.

The messages which are exchanged between two services need to be modeled next. The ExpressCongoBuy-example has one input and one output message with two inputs and three outputs (creditCardNumber, creditCardType and creditCardExpirationDate).

The above modeled messages can then be used to model the interfaces and their operations. The ExpressCongoBuy-example has only one interface, the FullCongoBuy-example has much more operations which were not modeled in fully detail here.

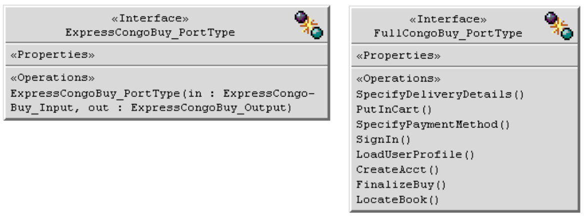


Fig. 8. Example: Interfaces.

Additional data types which are needed to model the inputs and outputs of the process are modeled in an own diagram. These elements can be imported from existing XSD-files or modeled from scratch.

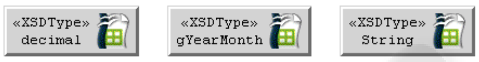


Fig. 9. Example: XSD-Types.

The next diagram specifies the details of each web service (Process). A service provider (here: Congo) can have multiple processes (ExpressCongoBuy and FullCongoBuy) which themselves can be described with additional information.

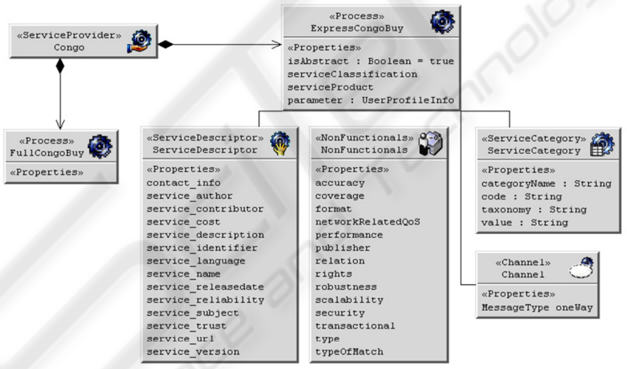


Fig. 10. Example: ServiceProvider with processes.

They can be categorized and described with functional and non-functional information (e.g. cost or trust). Similar to the attribute “isAbstract” which has a concrete type and a standard value, each of the attributes can be given a value which has been suppressed here for the sake of simplicity.

Each process has a behavior and in the example of ExpressCongoBuy this behavior consists of one simple action. This action has several inputs and one output and can be described more detailed with preconditions and effects. These can be specified in OCL, KIF or SWRL, where the former is a language used only in UML diagrams which needs to be transformed into a semantic web language and the latter two are used in several semantic web service languages.

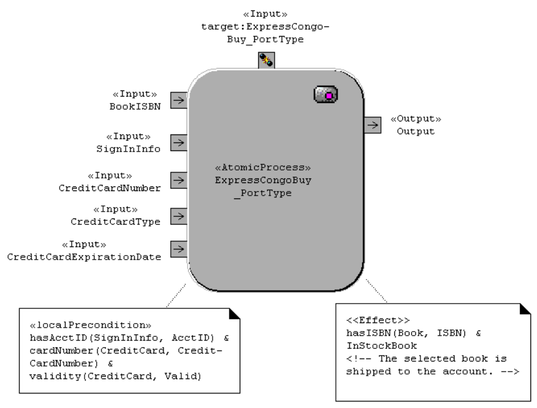


Fig. 11. Example: ExpressCongoBuy atomic process.

The FullCongoBuy-example does not only consist of one action, but of multiple actions (CompositeProcess) which are coordinated using the control nodes introduced above. These actions can be combined to composite processes again. To keep it simple we removed the more detailed functional description of each action (meaning inputs, outputs, preconditions and effects) in Fig. 12.

4 Related Work

There are several efforts to create a UML profile for semantic web services. However, to our knowledge none of the existing approaches tries to consider every existing W3C submission of semantic web services (meaning OWL-S, WSMO, WSDL-S, SWSF and SAWSDL).

In [14, 15] Grønmo et al. define transformations between UML and OWL-S and a web service composition based on this information. The developed profile uses the UML Ontology Profile (defined by Duric et al. for UML 1.5 class diagrams) to model the concepts of the ontology. They use a UML activity to describe a web service and attaching inputs and outputs which makes it difficult to use control nodes for the composition of several web services later. Their profile supports the generation of OWL-S and WSMO code, but is not designed to generate SWSF, WSDL-S or SAWSDL-code.

In [16] a model-driven approach for specifying semantic web services has been developed. However, the UML-profile only considered AtomicProcesses in OWL-S, not including the collaboration of several processes. It is only applicable to OWL-S and misses transformation rules for other approaches. [17] describes how semantic web services and policies can be modeled using the REWERSE Rule Markup Language (R2ML). [18] develops an MDD annotation methodology for semantic enhanced SOAs, but does not develop a UML profile for semantic web services in greater detail. [19] describes a case study with a methodological framework for the development of semantic web information systems (MIDAS-S) building on WSMO. In [20] (and other talks) the author promotes the integration of OWL-S and SWSF within the ODM. We completely agree and support this initiative, if the meta-model considers the other approaches of semantic web services, too.

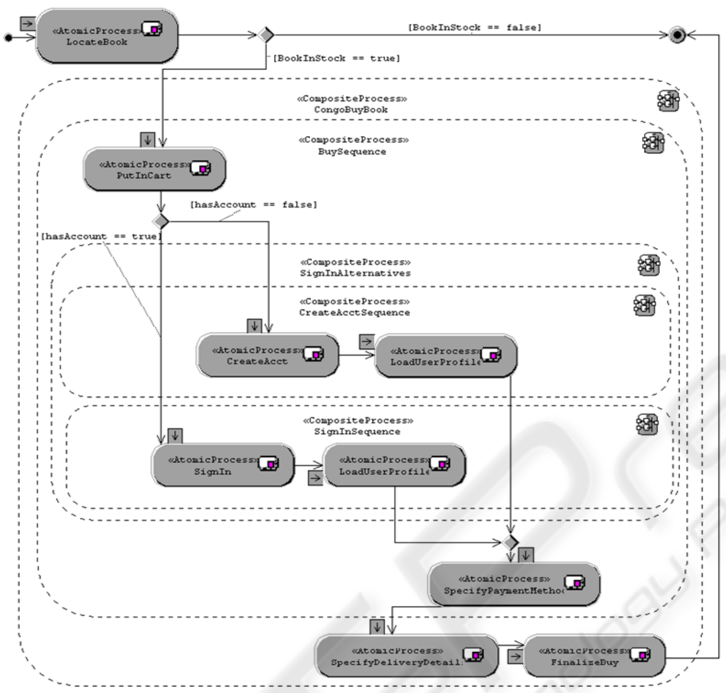

«AtomicProcess» LocateBook
[BookInStock == false]
[BookInStock == true]
«CompositeProcess» CongoBuyBook
«CompositeProcess» BuySequence
«AtomicProcess» PutInCart
[hasAccount == false]
[hasAccount == true]
«CompositeProcess» SignInAlternatives
«CompositeProcess» CreateAcctSequence
«AtomicProcess» CreateAcct
«AtomicProcess» LoadUserProfile
«CompositeProcess» SignInSequence
«AtomicProcess» SignIn
«AtomicProcess» LoadUserProfile
«AtomicProcess» SpecifyPaymentMethod
«AtomicProcess» SpecifyDeliveryDetails
«AtomicProcess» FinalizeBuy

Fig. 12. Example: FullCongoBuy composite process flow.

5 Conclusions and Future Investigations

We presented a meta-model and UML-profile which can be used to simply model a semantic web service and then generate code in one of the currently proposed SWS-languages. MM4SWS provides independence from the languages of all SWS approaches and provides the possibility to generate code for all of them. We integrated our profile into a UML CASE-tool and modeled a well-known example.

In order to make modeling of preconditions and effects easier we will develop a way to include rules in upcoming versions of our profile to support users in this still difficult task. The modeling of preconditions and effects currently needs to be done manually in UML2 constraints, so the user needs to know the expression from SWRL or KIF (or applies an automatic mapping from OCL to SWRL as described in [21]). Another issue is the difference between first-order and description logic that needs to be considered during the transformation process.

In the future we will also focus on a continuous approach from enterprise models to executable semantic web service code using this semantic web service model and work on an automatic discovery and integration of existing services into the model.

References

1. innoQ: *An Overview of the Web Services Standards landscape*, available online at http://www.innoq.com/soa/ws-standards/poster/, last called 2006-11-28
2. Object Management Group (OMG): *MDA Guide Version 1.0.1*, June 2003, available online at http://www.omg.org/docs/omg/03-06-01.pdf
3. Object Management Group (OMG): *Unified Modeling Language (UML) Specification: Superstructure*, Version 2.0, Final Adopted Specification, July 2005.
4. Object Management Group (OMG): *Meta Object Facility (MOF) Core Specification*, Version 2.0, January 2006.
5. Object Management Group (OMG): *Ontology Definition Metamodel (ODM)*, Sixth Revised Submission to OMG, ad/2006-05-01, June 2006.
6. Martin, D. et al: *OWL-S: Semantic Markup for Web Services*, November 2004, W3C Member Submission, available online at http://www.w3.org/Submission/OWL-S/
7. Akkiraju, R. et al.: *Web Service Semantics – WSDL-S*, November 2005, W3C Member Submission, available online at http://www.w3.org/Submission/WSDL-S/
8. Lausen, H., Polleres, A. and Roman, D. (Eds.): *Web Service Modeling Ontology (WSMO)*, June 2005, W3C Member Submission.
9. Battle, S. et al.: *Semantic Web Services Framework (SWSF) Overview*, September 2005, W3C Member Submission.
10. Farrell, J. and Lausen, H.: *Semantic Annotations for WSDL*, W3C Working Draft, available online at http://www.w3.org/TR/sawsdl/
11. Lautenbacher, F. *A UML profile and transformation rules for Semantic Web Services*. Technical Report 2006-20, Augsburg, Germany, available online at http://www.ds-lab.org/publications/reports/2006-20.html
12. Jablonski, S. and Bussler, C.: *Workflow Management: Modeling Concepts, Architecture and Implementation*. Int. Thomson Computer Press, 1996
13. Efftinge, S. and Kadura, C., 2006: *OpenArchitectureWare 4.1 Xpand Language Reference*, available online at http://www.eclipse.org/gmt/oaw/doc/4.1/.
14. Skogan, D.; Gronmo, R. and Solheim, I.: *Web Service Composition in UML*, presented at the International Enterprise Distributed Object Computing Conference (EDOC), Monterey, September 2004
15. Gronmo, R.; Jaeger, M. and Hoff, H.: *Transformations between UML and OWL-S*, presented at the European Conference on Model Driven Architecture – Foundations and Applications (ECMDA-FA), Nuremberg, November 2005
16. Timm, J. and Gannod, G.: *A Model-Driven Approach for Specifying Semantic Web Services*, presented at the International Conference on Web Services (ICWS 2005), July 2005
17. Kaviani, N.; Gašević, D.; Hatala, M.; Wagner, G. *Towards Unifying Rules and Policies for Semantic Web Services*. In: Annual LORNET Conference on Intelligent, Interactive, Learning Object Repositories Network, Montreal, QC, Canada, 2006.
18. Pondrelli L.: *An MDD annotation methodology for Semantic Enhanced Service Oriented Architectures*, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies, Porto, June 2005.
19. Acuna, C. and Marcos, E.: *Modeling Semantic Web Services – a case study*, Proceedings of the International conference on web engineering (ICWE06), Palo Alto, CA, USA, 2006.
20. Kendall, E.: *MDA and Semantic Web Services: Integrating OWL-S & SWSF with the Ontology Definition Metamodel (ODM)*, SOA, MDA and Web Services Workshop, OMG, March 2006.
21. Milanović, M.; Gašević, D.; Guirca, A.; Wagner, G.; Devedžić, V. *On Interchanging between OWL/SWRL and UML/OCL*. In: Workshop OCLApps at International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genoa, Italy, 2006.