MODEL BASED DEPLOYMENT OF WEB SERVICES TO STANDARDS-COMPLIANT RELIABLE MIDDLEWARE

László Gönczy
Budapest University of Technology and Economics
Magyar Tudosok krt.2. I.B.414., Budapest, Hungary, H-1117
gonczy@mit.bme.hu

János Ávéd
Budapest University of Technology and Economics
Magyar Tudosok krt.2. I.B.414., Budapest, Hungary, H-1117
avedjami@freemail.hu

Dániel Varró
Budapest University of Technology and Economics
Magyar Tudosok krt.2. I.B.414., Budapest, Hungary, H-1117
varro@mit.bme.hu

ABSTRACT

Nowadays, due to the rapid increase in the number of available web services, more and more emphasis is put on their reliability, availability, security, etc. In order to meet such non-functional requirements, a service needs to be designed for reliability by making design decisions on a high, architectural level. Fortunately, web service related standards also include how to provide reliable messaging between services (e.g. in WS-ReliableMessaging and WS-Reliability). Several frameworks (like RAMP, RM4GS) implement the standard to provide a reliability middleware. However, these standards typically offer very low, XML-level mechanisms for specifying the reliability parameters of web services when deploying them to a reliability middleware. For this purpose, we propose a model-driven approach for the deployment of web services to standards-compliant reliable middleware. The actual XML descriptors are generated automatically from platform-independent description of services enriched with reliability attributes by model transformations.

KEYWORDS

Model-based Development, Service Deployment, Dependability in SOA, Reliable Middleware.

1. INTRODUCTION

As the importance and incidence of Service Oriented System continuously grows, new means of automated integration of software components deserve an intensive investigation. Due to the rapid increase in the number of available web services, more and more emphasis is put on their reliability, availability, security, etc. In order to meet such non-functional requirements, a service needs to be designed for reliability by making design decisions on a high, architectural level. This results in the re-use of methods and techniques which were formerly used at a lower level of system design. Several standards and initiatives are available in the field of Web services to describe the non-functional properties of web services, such as WS-Reliability (OASIS, 2004), WS-RM (Bea Systems et al, 2002), WS-Security, WS-Policy, etc. However, these standards are mostly at the very low XML level, not supporting the visual design of standard compliant applications.

Therefore, there is a need for techniques which can process domain specific and easy-to-understand visual representations of a system, which are readable for an engineer and do not require the knowledge of the details of the actual platform, such as application server, description language, communication middleware, etc.

Hereby we present a methodology which follows the Model Driven Architecture approach (OMG, 2002) and uses graph transformation techniques to generate various types of code from high level models. Target platforms are RAMP (IBM, 2004) and RM4GS (Fujitsu, 2004) (implementation for the latter still in progress) for service deployment, WSDL (W3C, 2001) for service description and OWL (Martin, 2004) for service representation.

Our aim was to be as standard-compliant as possible to support the use of existing commercial applications servers (e.g. Apache Tomcat, IBM WebSphere, etc.) and Web service middleware, as well as a widely used reasoning tool like RACER (Haarsley et al, 2004).

The structure of the paper as follows. Section 2 discusses the concepts of model based development and deployment, also briefly summarizing the principles of graph transformations, Section 3 presents the target domain (reliable messaging) and the available middleware platforms, Section 4 describes the VIATRA2 model transformation framework (Balogh et al, 2005) and other parts of the tool chain we used, Section 5 discusses related work and the paper closes with some conclusions and brief description of ongoing and future work in Section 6.

2. MODEL BASED DEVELOPMENT FOR RELIABLE MESSAGING

As suggested by the MDA specification of the Object Management Group, we follow an approach which separates the concepts of a target domain from the technology-related details of implementation. First, a Platform Independent Model (PIM) is generated from the high level description of the system. This corresponds to the MDA approach which intends to separate low-level details from general concepts in order to make the transformations reusable and maintainable. Then Platform Specific Models are created from which the concrete code is generated. In the case of Web services with reliable messaging specifications, this step will mean a number of transformations which can be executed simultaneously.

2.1 General concepts

Fig. 1. shows our suggested development process. Transformations are denoted by arrows whilst the other elements show (intermediate) models and resulted code.

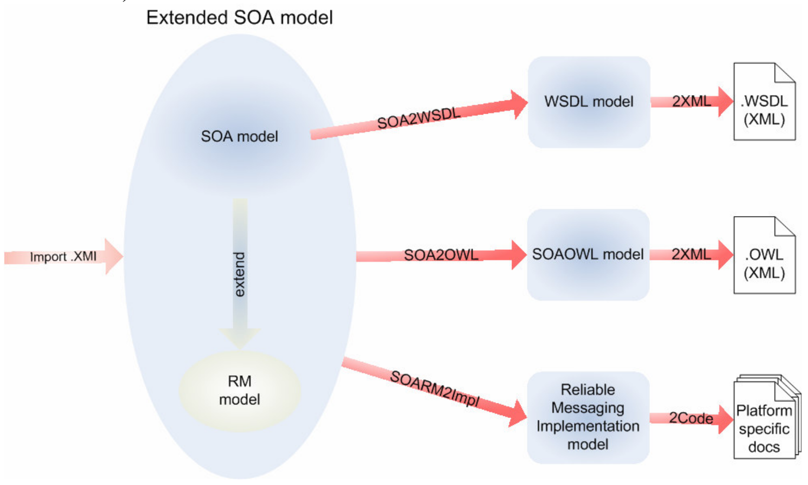

Figure 1. Model transformation-based deployment

As shown in Fig. 1., the MDA-conform deployment consist of the following steps:

- First, a system model is built, which should be compliant to the metamodel of SOA extended with reliable messaging properties. This is a high-level model, independent from the actual implementation. This model is then imported to a model transformation framework ("Import .XMI")
- Second, several transformations can be executed to generate domain-specific aspects of the complex system model. The results of this transformations are still high-level models (not in the

representation of a concrete language) but focusing only to some parts of the entire model. In our case, transformation “SOA2WSDL” creates a *service description* model (conform to the metamodel of WSDL), transformation “SOA2OWL” generates an *ontological model* (conform to the OWL metamodel) and a *reliable messaging implementation model* are generated, the latter being independent from the concrete standard.

- Finally, code (fragments) in the proper XML formats are generated. This includes WSDL description of the functionality of the services, OWL representation of the services and dependencies between them and deployment code for reliable messaging middleware. The XML generation is implemented in one common transformation, which takes an XML schema-conformant model and generates a concrete XML description. For the reliable messaging, as there are different target platforms (RAMP and RM4GS, described in the next section), first a platform-specific model is derived from the general reliable messaging description, then the XML generation is executed (both steps covered by “2Code” arrow).

2.2 Graph transformations

For the generation of configuration of services, we use model transformation techniques based on *graph transformation*. Here we only give a short introduction of these techniques, the interested reader finds more details and theoretical foundations for instance in (Ehrig et al, 1994). Generally speaking, graph transformations are pattern matching on typed graphs, where typed elements and their relations are described by a metamodel. Graph patterns can be considered as an UML class diagram while concrete graph instances are like UML object diagrams. The patterns are then used to modify parts of these graph instances. Hereby these instances are SOA models with reliable messaging properties. Usual application areas of graph transformations are model transformations and code generation (as in the case of the present work), model analysis, language engineering, reverse engineering, etc.

A graph transformation rule consists of a Left Hand Side (LHS), a Right Hand Side (RHS) and optionally a Negative Application Condition (NAC). The LHS is a graph pattern consisting of the mandatory elements which prescribes a precondition for the application of the rule. The RHS is a graph pattern containing all elements which should be present after the application of the rule. Elements in the RHS\LHS are left unchanged by the execution of the transformation, elements in LHS\RHS are deleted while elements in RHS\LHS are newly created by the rule. The fulfillment of the negative condition prevents the rule from being executed on the particular matching. Hereby we follow the Single Pushout Approach (SPO) approach with negative application conditions.

In our case, a graph pattern typically correspond to a part of the model (e.g. a port and the corresponding message specification), and the purpose of pattern matching is to generate instances of another metamodel or textual code. WSDL metamodel and concrete WSDL code are two examples for the target platform of transformations.

3. RELIABLE MESSAGING STANDARDS

At the moment, there are various industrial standards reflecting the emerging need for reliable Web services middleware. Here we concentrate on reliable messaging standards as running example. However, our approach can be adapted to support the generation of configurations of other non-functional Web services standards such as WS-Security. The main importance of reliable messaging standards are rooted in the fact that they can replace the existing several messaging middleware (such as Message Queuing servers or JMS) which are now used together with SOAP to provide a reliable asynchronous communication service.

Reliable messaging in the fields traditional distributed systems is closely related to the guaranteed *semantics* of message delivery. Usual delivery classes are the following:

- *At least once* delivery. In the case of normal operation, every message is transferred at least once, with the possibility of sending multiple instances of the same message. This can only be allowed in systems where this does not have an undesired side-effect.
- *At most once* delivery guarantees that no message will be sent multiple times to the receiver. However, the successful transmission of messages is not ensured.

- *Exactly once* delivery is the strongest delivery semantics, guaranteeing both the successful message delivery (usually acknowledgements are required for each message) and the filtering of duplicate messages.

Other delivery semantics, such as “x out of y” (meaning that from every y instances of the message at least x have to be transferred) exist in the literature. For practical reasons, the maximum number of retransmission is usually also bounded by an integer.

Currently, there are two existing SOA standards in the field of reliable messaging, both having a reference implementation publicly available. Web Service Reliable Messaging Protocol (WS-ReliableMessaging) is a specification of IBM, BEA, Microsoft and TIBCO, while WS-Reliability is an OASIS standard, submitted by Fujitsu, Novell, Oracle and Sun. Although they are different in some ways, they have a lot of common and a joint standard is likely to appear (see OASIS, 2005). The common points in these standards are that they both implement the above three delivery semantics, and both assume a *reliability middleware* layer under the layer of applications. The original applications are not aware of the message delivery semantics of the middleware which catches the outgoing and incoming SOAP messages of Web services and modifies them by inserting/removing reliability specific tags in the header of the SOAP envelope.

3.1 Metamodel of reliable messaging

A metamodel to describe both the static and the runtime elements of reliable messaging was created in (Gönczy and Varró, 2006). A subclass from SOA elements was derived in the reliable SOA metamodel, and then an association from the child class (e.g. RelMsgEnvelope) to the parent class (e.g. Message) was created in addition. The original messages are kept but wrapped into an envelope by introducing a new association, which corresponds to the actual implementations (in which the SOAP header is modified). The `RelMsgSpecification` class corresponds to the pre-defined requirements on the messages between two communicating parties (at the level of Ports, as in the standards) while the runtime properties of message instances are stored in attributes of instances of the `ReliabilityProperty` class.

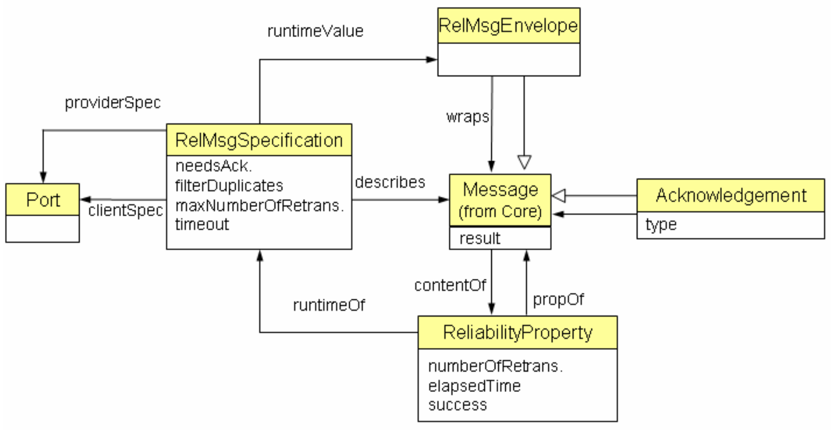

Fig. 2. Metamodel of reliable messaging

We implemented the metamodel shown in Fig.2. in the VIATRA2 framework and then later used this as a common metamodel for Platform Independent Model instances. This had the advantage that our transformations depend on the actual middleware platform and format only in the last phase of code generation (when a platform specific model is created and XML deployment descriptors are generated).

4. TOOL CHAIN

Here we present the tool chain we used, with an emphasis on the framework in which the model transformations were implemented.

4.1 The VIATRA2 model transformation framework

The VIATRA2 (Visualized Automated Transformations) framework is part of the Eclipse General Modeling Tools Project (Eclipse, 2005). It is a plugin-based, extensible Eclipse application with an own language for representation of models as typed graphs (VTML) and a language which supports the definition of transformation of these typed graphs. The VTCL language, first described in (Varró and Pataricza, 2003) allows for the definition of graph patterns (which can be considered as a declarative programming paradigm) and the control flow for model transformation by implementing Abstract State Machine (Börger et al, 1994) instructions, similar to those of traditional imperative programming languages.

To define transformation rules in VIATRA, one has to implement a metamodel of the source and (if different) the target language of the transformation. These in our case meant the creation of metamodels of reliable messaging concepts at different levels. To illustrate the modeling procedure, we present a small fragment of metamodel in Fig.2. in VTML.

```
entity(RM) {
            …
            entity(RelMsgEnvelope);
            subtypeOf(RelMsgEnvelope, SOA.messages.Message);
            relation(runtimeValue, RelMsgSpecification, RelMsgEnvelope);
            relation(wraps, RelMsgEnvelope, SOA.messages.Message);
            …
}
```

This code snippet corresponds to the definition of the ReliableMessageEnvelope in the metamodel. It refers to concepts in the core SOA metamodel and uses typed entities (graph nodes) and relations (directed graph edges). Metamodels of WSDL and OWL descriptions were also created.

After having the metamodels created, transformations between the metamodels can be implemented. To facilitate code generation (which is basically different from model transformations between different typed graphs, all represented within the framework), graph transformation rules in VIATRA may have an *action* part besides the left hand side, right hand side and NAC, which corresponds to the generation of some textual output.

4.2 Implementation details

Fig. 3 illustrates the tool chain we use to model and deploy web applications.

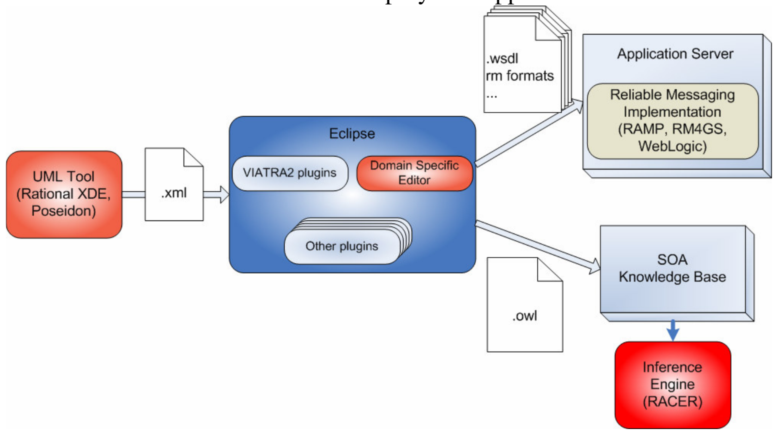

Figure 3. Tool chain to deploy SOA applications with reliable messaging

First, the engineering model is imported from a UML modeling tool, such as Rational XDE or Rational Software Architect. Note that the VIATRA2 can easily be extended by additional parsers to read the XMI output of other modeling tools. Once the model is parsed, we have a domain specific (but still platform-independent) model in the VCTL language of the transformation framework.

Alternatively, the domain specific model can be implemented within Eclipse, by describing it in a Domain Specific Editor. There is an ongoing research to implement such an editor using the Graphical Modeling Framework (GMF).

As a next step, the transformation steps discussed in Section 2.2 are executed within the VIATRA2 framework. These transformations create the platform-specific models of the system from the high-level description. Finally, the code generation transformations are executed.

The output code, generated by the transformations, are standard WSDL (which describes the interface of the services and can be used to generate clients for them), OWL representation which can be used to perform queries and other actions by a reasoning system as briefly described in the next section, and reliable messaging configurations for the IBM RAMP Toolkit, running on IBM WebSphere Application Server 6.0. Note that this configuration itself has to be added to the existing XML configuration descriptor of an existing application server instance. RM4GS for Apache is also under investigation. However, at the moment, we have not found any documentation on the format on the configuration descriptor (documentation of message formats was available) and were not able to generate a proper code. This is, nevertheless, a technical problem which does not affect the main deployment technology and needs only some further experience with the related product.

As an experience, we performed the transformation lifecycle on sample models, implemented in a standard UML modeling tool, using stereotypes which correspond to the SOA reliable messaging metamodel (from which a specific UML package was derived). Then the models were parsed by the VIATRA2, and by executing the transformations, WSDL descriptions of the services, OWL representation of the system and a reliable messaging configuration file for RAMP was created. The only additional information was the location of the services, which can also be included in the UML model but also can be considered as a parameter defined later.

4.3 Ontologies for modeling non-functional aspects of Web services

Ontologies are used in the filed of semantic web to represent knowledge about the world in a structured format. Once a metamodel is constructed to describe the target application domain (properties of reliable messaging), an ontological description can also be generated using the VIATRA2 framework. This allows us to use a reasoning system such as RACER to validate concrete system models against metamodel-level constraints. Complex queries related the semantic structure of the system (such as "Are there services with a port type X which only use services with given reliable communication characteristics?") can be executed. To fully automate the query generation, some additional transformations have to be implemented as discussed in the "Future research" section. Fig. 4. shows the possibilities of extending the ontology in a modular way to describe multiple aspects of system design, or more technically, multiple WS-* configurations.

Here we generate an .owl file which can be read by RACER. This spares the system engineer the direct creation of an ontology in some specific tool, and let him focus on the design of the system instead, still having the opportunity to use semantic verification and queries.

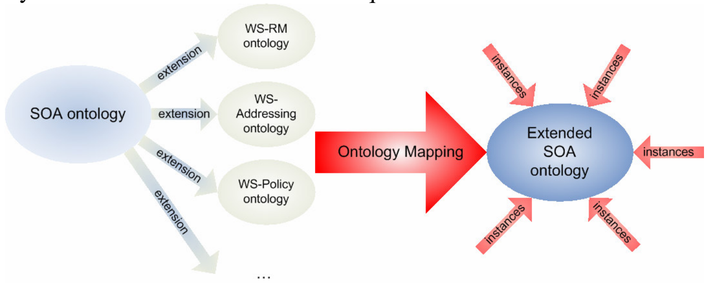

Figure 4. Extensions for the SOA ontology

5. RELATED WORK

(Rodrigues et al 2004) introduces a profile for reliability of J2EE applications. This offers a support for reliability engineering in the MDA framework in a different technical context. Non-functional aspects of e-business applications are discussed among others in (Pataricza et al, 2006), having some description of deployment optimization for J2EE applications, but without discussing details of model-based deployment.

A framework for automated WSDL generation from UML models is described in (Vara et al, 2005), using the UML extensions of MIDAS (Careces et al, 2003). (Gronmo et al, 2004) addresses a similar problem. For the reengineering of existing Web service applications, Federica framework was presented in (Fuentes et al, 2005). However, none of these works considers non-functional properties of Web services.

The work presented in (Algawait and Ghandeharizadeh, 2004) aims to develop a dependable web services framework, which relies on extended proxies. However, this needs a modification at the client side in order to handle exceptions and find new service instances. Moreover, the reconfiguration of client side proxies uses non-standard WSDL extensions while we concentrated on standards-compliant solutions.

Dependable Web services are investigated in (Tartanoglu et al, 2002) from the composition aspect and in (Looker et al, 2005) from the aspects of fault injection and quantitative analysis.

General Service Oriented Architecture elements were considered as typed graphs with graph transformation rules visually describing their behavior in (Baresi et al, 2004). Definition of the metamodel for reliable messaging was and capturing the dynamic behavior by graph transformation rules (Gönczy and Varró, 2006). Verification of the behavior of the system (i.e., checking the conformance to requirements on reliable messaging) was performed in (Gönczy et al, 2006), thus giving a formal semantics which can be checked by using some basic verification techniques and tools. Our work differs from these in that we focus on the practical aspects of deployment and implementation issues in this context.

6. CONCLUSIONS AND FUTURE RESEARCH

Our aim was to implement a complex system, based upon model transformation techniques, to generate deployment code and ontological description of Web services having specifications enriched with non-functional requirements. The proposed techniques use the VIATRA2 model transformation framework with a UML model or VIATRA representation as input. The target platforms of the transformation sequences were WSDL, OWL and WS-ReliableMessaging configuration descriptors for RAMP Toolkit of IBM.

Ongoing work includes the implementation of a GMF-based domain specific editor to enhance the definition of SOA models enriched with non-functional properties. The automated generation of the RM4GS configurations is also to be finished. However, as mentioned earlier, this needs only a minor effort and practically one simple transformation creating WS-Reliability specific elements from the platform independent RM description. The integration of RACER to the model transformation framework itself is also an interesting problem, raising the issue of generating ontology-level queries and constraints from high-level models. Finally, other aspects and platforms such as security and WS-Security can be included in our extensible approach, by means of new metamodels and transformations, but also using existing parsers and the XML generator component.

REFERENCES

Alwagait, E., and Ghandeharizadeh, 2004. S.DeW: A Dependable Web Services Framework. In *Journal RIDE*, IEEE Computer Society, Vol .1 pp. 111-118.

Balogh, A. et al, 2005. The VIATRA2 model transformation framework, Presented at *European Conference on Model Driven Architecture, Foundations and Applications (ECMDA-FA2005)– Tools track*, Nuernberg, Germany.

Baresi L. et al, 2006, Style-Based Modeling and Refinement of Service-Oriented Architectures. In *Journal of Software and Systems Modelling*.

BEA Systems et al, 2002. *Web Services Reliable Messaging Protocol (WS-ReliableMessaging) specification.*

Börger, E., and Stark, R., 2003. *Abstract State Machines. A method for High-Level System Design and Analysis.* Springer, Heidelberg, Germany.

Caceres, P. et al, 2003. A MDA-Based Approach for Web Information System Development. In *Proceedings of the Workshop in Software Model Engineering (WiSME@UML2003)*, San Francisco, USA.

Eclipse.org, 2005. *Generative Modeling Tools (GMT) project*, http://www.eclipse.org/gmt/.

Ehrig, H., et al, 1999. *Handbook on Graph Grammars and Computing by Graph Transformation, vol. 2: Applications, Languages and Tools*, World Scientific, Singapore.

Fuentes, J. M. et al, 2005. Semi-automatic Web service generation. In *Proceedings of the IADIS WWW/Internet Conference (ICWI 2005).* Lisbon, Portugal.

Fujitsu Limited et al, 2004. *RM4GS Reference Guide, Version 1.0.* http://xml.coverpages.org/rm4gs20041125-reference.pdf

Gönczy, L. and Varró, D., 2006a.: Modeling of Reliable Messaging in Service Oriented Architectures, In *Proceedings of the Inernational. Workshop on Web Service Modeling and Testing (WS-MATE 2006)*, Palermo, Italy, pp.35-50.

Gönczy, L. et al, 2006. Modeling and verification of reliable messaging by graph transformation systems. In *Proc. of the Workshop on Graph Transformation for Verification and Concurrency (GT-VC 2006)*, Bonn, Germany.

Gronmo, R., et al, 2004. Model-driven Web Services Development. Presented at the *2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*, Taipei, Taiwan.

Haarslev, V. et al, 2004. *RACER User's Guide and Reference Manual Version 1.7.19.*

IBM alphaworks, 2004. *Reliable Asynchronous Message Profile (RAMP) Toolkit*, In http://www.alphaworks.ibm.com/tech/ramptk

Kleppe, A., et al, 2003. *MDA Explained, The Model Driven Architecture: Practice and Promise*. Addison Wesley, Boston, USA.

Laprie, J. P. et al, 2004. Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, Vol.1, No.1. pp. 11-33.

Looker, N., et al, 2005. An Ontology-Based Approach for Determining the Dependability of Service-Oriented Architectures. In *Proceedings of the 10th IEEE International Workshop on Object-oriented Real-time Dependable Systems*, Sedona, USA.

Martin, D. et al, 2004. *OWL-S: Semantic markup for web services, v1.1.* In http://www.daml.org/services/owl-s/.

OASIS Open Consortium Web Services Reliable Messaging TC. 2004. *WS-Reliability 1.1 Standard.*

OASIS Open Consortium, 2005. *Call for Participation in the OASIS Web Services Reliable Exchange (WS-RX),* In http://xml.coverpages.org/ni2005-05-04-a.html

Object Management Group (OMG), 2002. *Model Driven Architecture.* http://www.omg.org/mda

Pataricza, A., et al, 2006. Chapter Verification and Validation of Nonfunctional Aspects in *Enterprise Modeling with UML*, Idea Group Publishing, Hershey, USA.

Rodrigues,G.N., et al, 2004. Reliability Support for the Model Driven Architecture. In *Proceedings of the Workshop on Software Architectures for Dependable Systems (WADS 2003),* Portland, USA, pp. 79-98.

Tartanoglu, F. et al, 2002. Dependability in the Web Service Architecture. In *Proceedings of the Workshop on Software Architectures for Dependable Systems (WADS 2002)* , Orlando, USA.

Vara, J.M. et al, 2005. WSDL Automatic Generation from UML Models in a MDA Framework. In *International Journal of Web Services Practices*, Vol.1, No.1-2, pp. 1-12.

Varró, D. and Pataricza, A., 2003. VPM: A visual, precise, and multilevel metamodeling framework for describing mathematical domains and UML. In *Journal of Software and Systems Modeling*, 2, 187-210.

W3C Consortium, 2001. *Web Service Description Language 1.1 Specification,* http://www.w3.org/TR/wsdl.