Automated Generation of Metamodels for Web service Languages

Behzad Bordbar and Athanasios Staikopoulos

School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK
B.Bordbar@cs.bham.ac.uk, A.Staikopoulos@cs.bham.ac.uk

Abstract: Recently, the application of the MDA to Web services has received considerable attention. In the MDA, models are instances of the MOF based metamodels. Model Transformation, which is a key feature of the MDA, can carried out via defining Transformation Rules between two MOF compliant metamodels. As a result, finding MOF compliant metamodels for languages is an essential prerequisite for model transformation.
This paper presents a semi-automated, tool-based method for the generation of MOF compliant metamodels for languages, which are specified via XML Schema Descriptions (XSD). We demonstrate that our approach can easily be implemented using existing XML Schema integration tool and UML CASE tool. To explain the approach, the paper sketches the stages involved in the generation of a metamodel for Web Service Description Language (WSDL) and compares the resulting metamodel with an existing metamodel for WSDL.

1. Introduction

Web services are Web- based enterprise application that use XML [19] based standards and transport protocols to communicate with each other in a platform and a programming-language independent manner. Applying Model Driven Architecture (MDA) [6][8][13] to Web services design has recently received considerable attention [1][8][3][4]. In particular, [1][8] study the Model Transformation for Web services and present a set of case studies involving the transformation of Web services models to various implementation platforms such as Java, Web Services Description Language (WSDL) [18] and EDOC [12].

Currently, there are a number of specifications and vocabularies defined and expressed in terms of the Extended Markup Language (XML) such as the Web Services Description Language (WSDL) [18] for Web Services. Such languages are XML extensions and are defined accordingly to a well-formed structure, the XML Schema. Therefore, an XML schema defines the language in the same respect where MOF is used to define the UML language. Considering the similarity it would be very beneficial within the domain of transformations to represent the XML family of languages such as Web Services in a MOF compliant metamodel.

In the MDA, each model is based on a specific *metamodel*, which defines the language that the model is created in. All metamodels within MDA, are based on a unique metamodel called Meta Object Facility (MOF)[14]. As a result, Model Transformations can be carried via defining Transformation Rules between two MOF compliant metamodels [1][3][6]. Consequently, there are two stages involved in any Model Transformation

- introducing MOF compliant metamodels for source and destination languages
- specifying Transformation Rules between metamodels.

This paper, which only deals with the first bullet point, aims to present a semi-automated, tool-based method for the generation of MOF compliant metamodels for languages, which are based on XML Schema Descriptions (XSD) [22] specification. In particular, Web Service languages such as WSDL [18], UDDI [11], SOAP[20], WSCI [21] and BPEL4WS [10] are examples of such languages. In general introducing a metamodel for each of the above languages involves identifying the concepts involved in the language and their relationship. Often, the starting point is reading and understanding the specification of such languages,

which are published by organizations such as W3C [17] and OASIS [9]. The next step is to create a conceptual model involving the model element of the language and their relationship. However, specification of all above languages includes an XML Schema Description (XSD), which is a meta-language representing various features for constructing and formalising the vocabulary and grammar of the XML model of the language. The current paper explores the idea of using the XSD representation of the language and generating MOF compliant metamodel for the language. The paper sketches an implementation of our method via *hyper*Model [6], an XML schema design tool, and Poseidon for UML [16]. We shall also apply our method to create a metamodel for WSDL and compare the result with a WSDL metamodel presented in [1].

The paper is organised as follows. The next section is a brief review of concepts used in the paper. Section 3 present the core of our approach and sketches the implementation via *hyper*Model and Poseidon UML tool. Section 4 is a case study involving the creation of a metamodel for WSDL. Section 5 sketches the future wrok. Finally, section 6 presents a conclusion.

2 Preliminaries

Kurtev and van den Berg [7] identify four MDA Model Transformation scenarios. Three of the scenarios studied in [7] make direct use of the definition of the Transformation Rules between metamodels. In particular, in the context of Web services, model transformations can be carried out via defining Transformation Rules between two MOF compliant metamodels [1][3][6]. Figure 1, depicts an example of the use of Transformation Rules for model transformation [1].

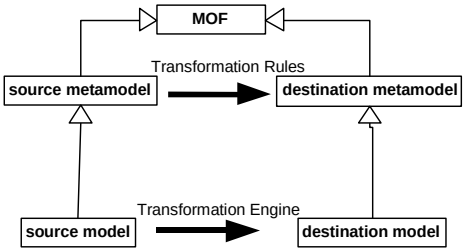

Figure 1: Using Transformation Rules in the MDA

As a result, defining a metamodel is one of the main steps in the process of the Model Transformation. In this paper, we are dealing with the creation of metamodel for languages for which the XML Schema Description (XSD) is available. This section presents a brief introduction on various concepts involved in the Model Transformation for XML based languages.

2.1 XML, XMI and XSD

The Extended Markup Language (XML) [19] is a cross-platform, text based W3C [17] standard for interchanging, structuring and representing data. One of the main characteristics of the XML is its extensibility mechanism and its flexibility to define complicated tree hierarchical structured data. In addition, XML can be used as a meta-language, allowing the generation of a whole family of XML languages. Such languages may be specialised in specific domains such as Web Services with WSDL [18], UDDI [11], BPEL4WS [10], Ontology with RDF and model interchange formats with XMI [15].

The XML Schema Definition (XSD) [22], which is also a W3C standard, is an XML language for describing XML documents. It offers a set of features both for specifying and formalising the vocabulary and the grammar of XML documents, and to impose various

constraints on their content. In this way, XSD provides a validating mechanism, allowing computer programs to validate and check the XML document for well-formedness.

The XML has also been used to create a common interchange format between UML tools for interchanging models and metadata. The XML Metadata Interchange (XMI) [15] is a format introduced by the OMG, combining the rigor of the MOF models with the XML definition semantics.

2.2 Transformations between XML and UML

The XML Metadata Interchange (XMI) is designed to facilitate the interchange of data and metadata expressed via the MOF. As a consequence, the XMI specification defines a number of mapping rules that specify how to generate XML Document Type Definition (DTD) and XSD schema from class diagrams. The XMI also specifies methods of producing MOF models from such input formats. The automatically generated DTDs and XML Schemas are based on the MOF defined rules and allow the MOF-based models to be serialized validated and interchanged among different tools without controversies. This makes XMI a necessary intermediate medium standing between MOF models and XML representations. Therefore any transformations from XML to MOF/UML need to be based or extend XMI. The transformation from an XML Schema or DTD to an XMI format can be performed using the Extensible Stylesheet Language (XSLT).

One of the key feature of the XMI is that it provides *parameterised mapping*, i.e. by choosing different mapping parameters, it is possible to define different mappings from a UML model to its schema representation. For example, it is possible to choose between mapping a class attribute to an XML attribute or a an XML element.

3 A tool-based approach to metamodel generation

A language metamodel defines the model elements of the language, specifies the semantics of language and relationship between various model elements. As a result, the modeller often starts by understanding the language description by studying its specification and creating a conceptual model involving the entities of the language and their relationship. Currently, there is no systematic way of creating such conceptual models. Figure 2 depicts the outline of our approach, which aims to address this issue. To create a MOF metamodel, we shall start from the XSD Schema representing the language. The XSD documents, for most Web service languages are included and published in their specifications, available from W3C www.w3.org or OASIS www.oasis-open.org web pages. As depicted in Figure 2, an *XML transformation tool* can be used to covert the XSD document into the XMI format, which can in turn be imported by a *UML tool* as a class diagram. As a result, the transformation from an XML Schema to a UML Model is a fully automated process, which is carried out via CASE tools. The UML model presents a clear, high-level view of the involving concepts and their relationship. At this point, the *Modeller* begins refining the UML Model by consulting the *Language Description*. However, unlike the ad hoc approach, the created UML Model can guide the refinement of the model by pointing out the existing model elements that the modeler needs to inquire about.

3.2 Implementation

*hyper*Model [6] is an XML schema design and integration tool, offering various UML modeling capabilities. *hyper*Model is offered as a free plug-in to Eclipse workbench [2][1] allowing the transformation of XML vocabularies and schema into XMI 1.0 format. To implement our method, we start by opening the XSD document of the language in *hyper*Model. In *hyper*Model creating an XMI document from an XSD document is at a click

of a mouse. It is possible to view the XMI model as a UML class diagram in *hyper*Model/Eclipse. However, in order to have greater flexibility in editing and refining of the model, we import the XMI document into a separate UML tool, for example Poseidon for UML [16]. In the next section, we shall apply our method to generate a metamodel for WSDL. We shall also compare our metamodel with the WSDL metamodel presented in [1]

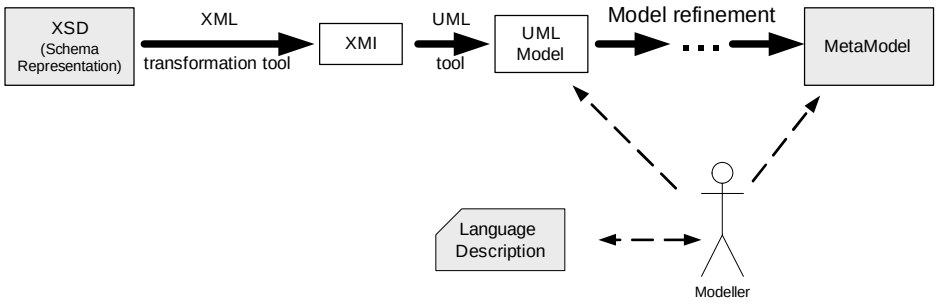

Figure 2 : Generating metamodels from XSD

4 Case study: a metamodel for WSDL

The Web Service Description Language (WSDL) [18] describes the syntax and semantics necessary to call up services. The language specification [18] contains the XSD for the WSDL. Figure 3 depicts a part of the first version of the WSDL metamodel created by *hyper*Model. Figure 4 depicts the refined version of the metamodel on which the following changes are made.

The XML provides an extensive mechanism for documenting and extensibility features. Some of the elements in the metamodel of Figure 3 are specific to XML and have no equivalent in MOF. For example, `tExtensibleAttributesDocumented` allows future extensions of the WSDL by adding new attributes from other XML namespaces. To create a MOF compliant metamodel, in the refined version, all such elements are deleted. Similarly, there are various stereotypes, for example `<<XSDattribute>>`, which are created from an XML tag representing *XSD attributes* , which are also deleted.

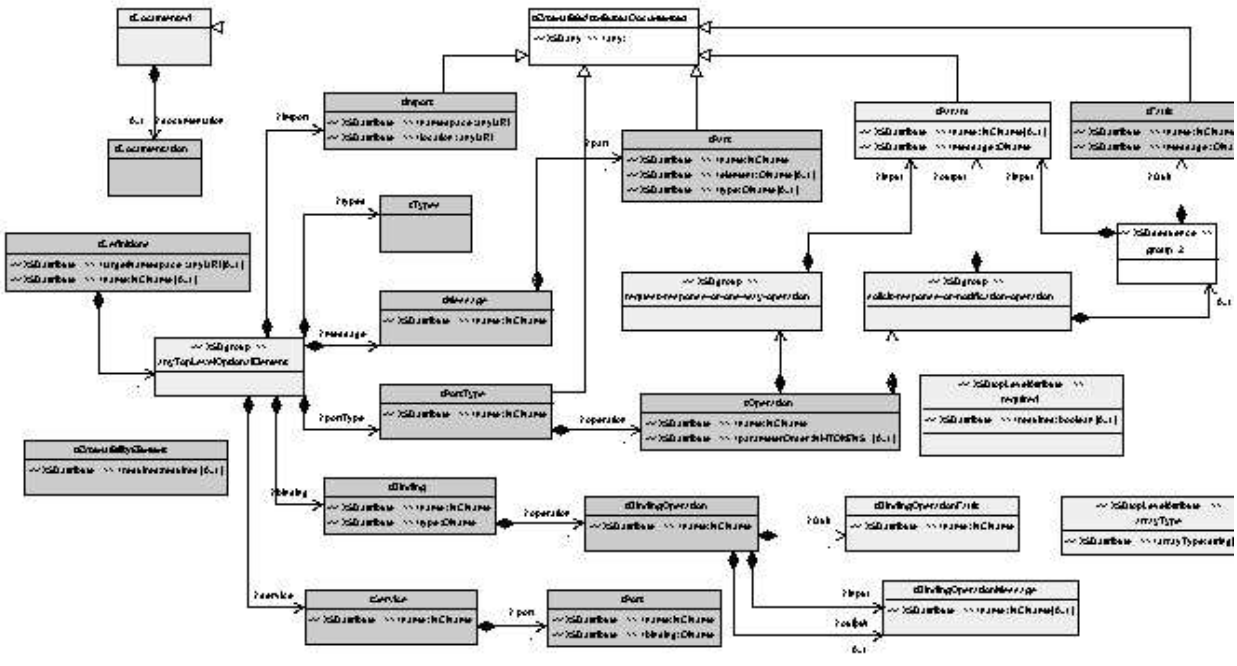

Figure 3: Initial WSDL Metamodel, version 1

The metamodel of Figure 4 contains the model element *group_2*, see the top-right corner of the picture, which is the translation of the following piece of XSD code.

```
- <xsd:group name="solicit-response-or-notification-operation">
- <xsd:sequence>
  <xsd:element name="output" type="wsdl:tParam" />
- <xsd:sequence minOccurs="0">
  <xsd:element name="input" type="wsdl:tParam" />
  <xsd:element name="fault" type="wsdl:tFault" minOccurs="0"…/>
  </xsd:sequence>
```

Creation of this metamodel element is a direct result of the XSD tag `</xsd:sequence>`, which means the elements within its scope must appear as a sequence, see [19]. This is a feature exclusive to XML. Eliminating such model element requires refactoring of the diagram, which can be easily done by redirecting each association of the model element `group_2`, to its source, *solicit-response-or-notification-operation*. This results in the metamodel of Figure 5. For the rest of the section, we shall compare the metamodel of Figure 5 created via our method and the WSDL metamodel presented in [1], depicted in Figure 6.

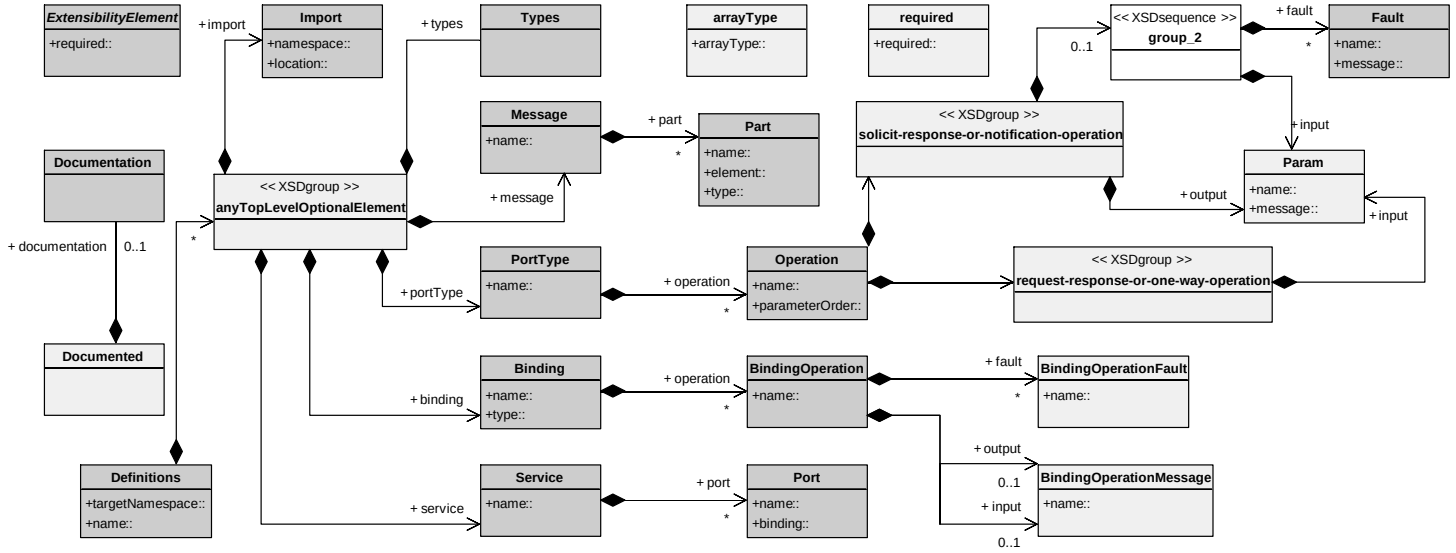

Figure 4 : Refined WSDL metamodel, version 2

There are clear similarities between the two metamodels. The gray shaded metamodel elements in Figure 5 are directly appearing in the other model. Figure 5 is more detailed and contains more elements. However, the authors of [1] clarify that the paper presents only a simplified version of their metamodel.

There are also a number of elements in Figure 6 which are not in our metamodel. Most notably, `input` and `output` are modeled as separate WSDL types in Figure 6, where in our case, they are modeled as metamodel *attribute ends*, which are of type parameters (`Param`). This correspond to the following line in the XSD document for the WSDL

```
<xs:element name="input" type="wsdl:tParam" />
```

In fact, we noticed that the XSD description of the WSDL does not define the types `inuput` or `output`. However, WSDL documentation [18] mentions phrases "output element" and "input elements" in numerous occasions. As a result, it is very natural that the authors [1] included `input` and `output` as model elements.

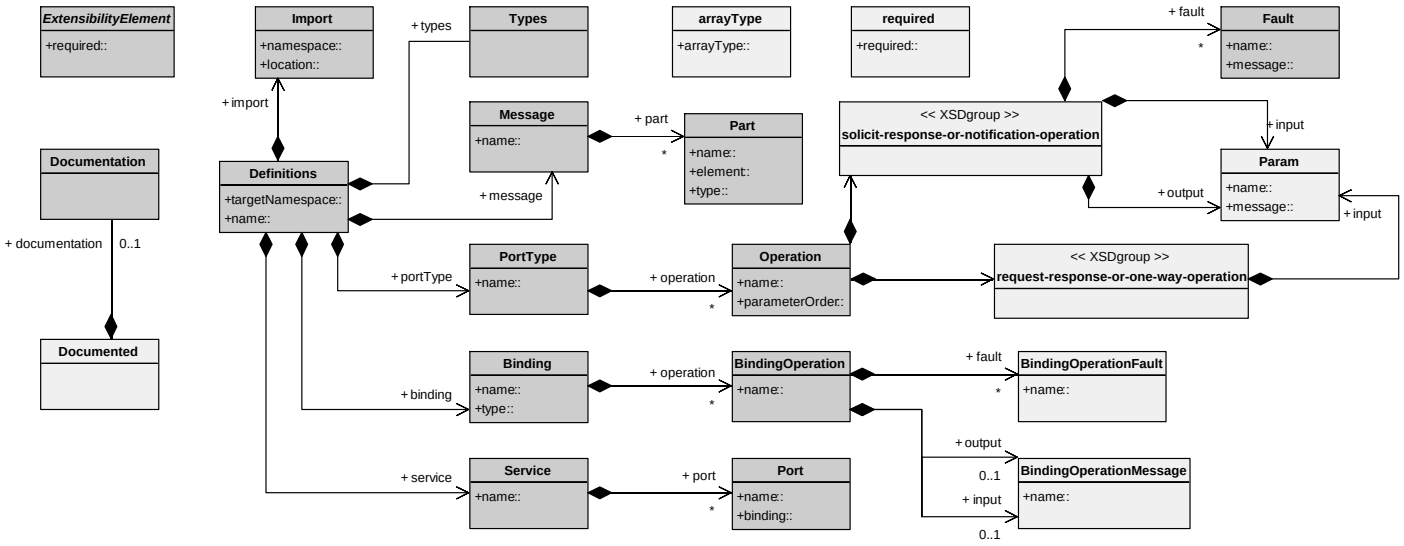

Figure 5: WSDL metamodel, final version

From the conceptual point of view, there is hardly any difference between the two metamodels[1]. From the model transformation point of view, the advantage of choosing one metamodel over another is not clear to us and remains a subject for future research.

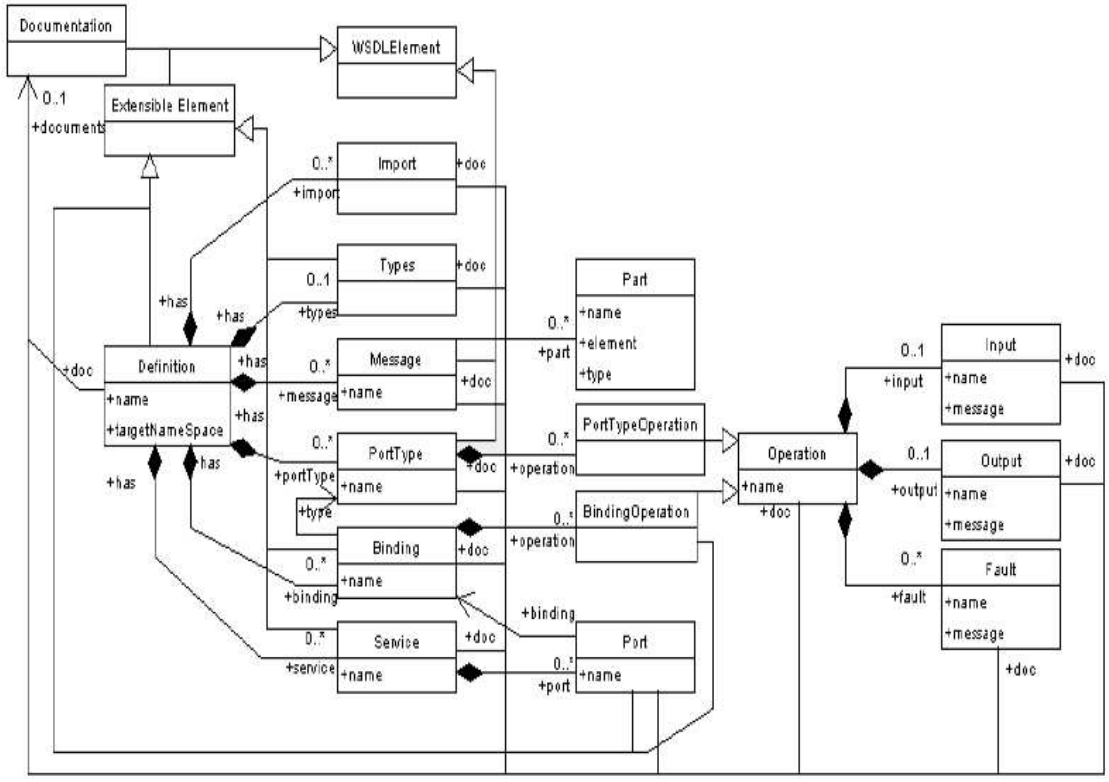

Figure 6 : WSDL metamodel, copied from[1]

[1] This is subject to including parameters (`Param`) in the metamodel of Figure 6.

5 Future works

*hyper*Model is a powerful tool for Web service integration and XML Schema design. However, the transformation from XSD to XMI is carried out in rigid form. It is important to make use of the *parameterized mapping* facilities of the XMI and be able to choose parameters to alter the transformation map. Moreover, the UML model created from the schema in *hyper*Model/Eclipse is only partially editable, which forces us to use another UML tool to edit and refine the model.

We have applied our method to generate metamodels for a number of Web service languages. Currently, the refactoring part of the process, which is at the heart of our approach, is performed manually. There is a clear scope for research into the automation of such refactoring activities. We are currently implementing the above method as an integrated UML tool, which particularly aims at the following

- providing greater flexibility in the transformation from XSD to XML, by allowing the modeller to choose the mapping of model elements
- producing better edit and viewing facilities to assist the modeller
- automating the refactoring of the model

6 conclusion

This paper presents a semi-automated method of generating metamodels for languages, which are specified via XML Schema Description (XSD). The method presented starts by creating an XMI document from the XSD specification of the language. The XMI model, which can be imported as class diagram in a UML tool, provides a high level view of the concepts involved in the language and their relationship. Such model is subsequently refined to create a metamodel for the language. The process of refinement may require refactoring of the model to eliminate some elements, which exclusively correspond to XML model elements and have no equivalent in MOF. Our method is particularly suitable for Web service languages and the paper sketches the generation of a metamodel for Web Service Description Language (WSDL).

References

[1] J. Bezivin, S. Hammoudi, D. Lopes, F. Jouault, *An Experiment in Mapping Web Services to Implementation Platforms*, Atlas Group, INRIA and LINA University of Nantes, Research Report, March 2004

[2] Eclipse project, www.eclipse.org

[3] D. S. Frankel, Model Driven Architecture, *Model Driven Architecture: Applying MDA to Enterprise Computing*, OMG Press, ISBN: 0471319201, January 2003

[4] D. S. Frankel, *White Paper: Using Model Driven Architecture to Develop Web Services*, IONA Technologies PLC, Second Edition, April 2002

[5] *hyper*Model, www.xmlmodeling.com/hyperModel/index.html

[6] A. Kleppe, J. Warmer, W. Bast , *MDA Explained. The Model Driven Architecture: Practice and Promise*, Addison-Wesley, ISBN: 321-19442-X, April 2003

[7] I Kurtev and K. van den Berg, *Unifying Approach for Model Transformations in the MOF Metamodeling Architecture*, Proceedings of the 1st European MDA Workshop, MDA-IA, University of Twente, the Nederlands, March 2004

[8] D. Lopes, S. Hammoudi, *Web Services in the Context of MDA*, University of Nantes, France, 2003

[9] OASIS, available from http://www.oasis-open.org/

[10] OASIS, Business Process Execution Language for Web Services (BPEL4WS), available from OASIS site
[11] OASIS, *Universal Description Discovery & Integration (UDDI)*, Version 3, available from OASIS site
[12] OMG, *Enterprise Collaboration Architecture (ECA) Specification*, Object Management Group, Version 1.0, February 2004
[13] OMG, *Object Management Group*, Available from http://www.omg.com
[14] OMG, *Meta Object Facility (MOF) Specification*, Object Management Group, Version 1.4, April 2002, available from OMG site
[15] OMG, XML Metadata Interchange (XMI), available from OMG site
[16] Poseidon for UML, www.gentleware.com/
[17] W3C, World Wide Web Consortium, www.w3.org
[18] W3C, *Web Services Description language (WSDL) Version 2.0*, W3C Working Draft, November 2003
[19] W3C, *Extensible Markup Language (XML) 1.0*, Third Edition, W3C Recommendation, Available from http://www.w3.org/TR/2004/REC-xml-20040204, February 2004
[20] W3C, *Simple Object Access Protocol (SOAP)*, Version 1.2, W3C Recommendation, Available from http://www.w3.org/TR/soap12-part1, June 2003
[21] W3C, *Web Service Choreography Interface (WSCI) 1.0*, W3C Note, Available from http://www.w3.org/TR/wsci, August 2002
[22] W3C, XML Schema Primer