Notice of Violation of IEEE Publication Principles

"An MDA Based Approach for Facilitating Representation of Semantic Web Service Technology"
by Thida Win, Hninn Mar Aung, Ni Lar Thein
in the Proceedings of the 6th Asia-Pacific Symposium on Information and Telecommunication Technologies, 2005 (APSITT 2005), Nov. 2005, pp. 260-265

After careful and considered review of the content and authorship of this paper by a duly constituted expert committee, this paper has been found to be in violation of IEEE's Publication Principles.

This paper contains significant portions of original text from the paper cited below. The original text was copied without attribution (including appropriate references to the original author(s) and/or paper title) and without permission.

Due to the nature of this violation, reasonable effort should be made to remove all past references to this paper, and future references should be made to the following article:

"A Model-Driven Approach for Specifying Semantic Web Services"
by John T.E. Timm and Gerald C. Gannod
in the Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005). vol.1, July 2005, pp. 313- 320

"An MDA-based Approach for Facilitating Adoption of Semantic Web Service Technology"
by Gerald C. Gannod and John T.E. Timm
in the Proceedings of the IEEE EDOC Workshop on Model-Driven Semantic Web (MDSW04), Sept. 2004, pp. 654-675

An MDA based Approach for Facilitating Representation of Semantic Web Service Technology

Thida Win*, Hninn Mar Aung*, Ni Lar Thein**

University of Computer Studies, Yangon,Myanmar
*thida.win@gmail.com,** nilarthein@mptmail.net.mm

Abstract

A semantic web service extends the capabilities of a web service by associating a semantic description of the web service in order to enable better search, discovery, selection, composition and integration. Semantically rich language such as OWL-S has been created in order to provide a mechanism for describing the semantics of semantic web services. Unfortunately, for the common developer the learning curve for such languages can be steep, providing a barrier for adoption and widespread use. Model Driven Architecture (MDA) is an approach to software development that is centered on the creation of concerns between a specification and an implementation. We are developing an approach that allows architecture to focus on creation of composite web services by specifying a semantic web services using OWL-S specifications. These composite services are specified using standard UML model and generating specifications and applications using MDA concepts.

Keywords: Model Driven Architecture, UML, OWL-S, Semantic Web Services.

1. Introduction

A *web service* is a loosely coupled component that exposes functionality to a client over the Internet (or an intranet) using web standards such as HTTP, XML, SOAP, WSDL and UDDI. Of the many challenges of using web services are the problems of specification, search, discovery, selection, composition, and integration. The current state of practice in web services is dominated by the use of the Web Service Description Language (WSDL) [1] to specify access to services. This language lacks an ability to address the challenges due to a lack of semantic constructs.

A *semantic web service* extends the capabilities of a web service by associating semantic concepts to the web service in order to enable better search, discovery, selection, composition, and integration. Semantically-rich languages such as OWL-S [9] have been created on order to describe concepts and semantics related to a web service using ontologies. Unfortunately, for the common developer, the learning curve for such languages can be steep, providing a barrier to widespread adoption.

Model Driven Architecture (MDA) is an approach to software development that is centered on the creation of models rather than program code. The primary goals of MDA are portability, interpretability and reusability through an architectural separation of concerns between the specification and implementation of software. In MDA based approaches, the focus is upon creation of software via the development of UML models.

We are developing an approach that allows a developer to focus on creation of semantic web services and associated OWL-S specifications via the development of a standard UML model. By using MDA approach, the technique facilitates creation of descriptions of semantic concepts while hiding the syntactic details associated with creating OWL-S specifications. By using transformation from equivalent UML constructs, difficulties caused by a steep learning curve for OWL-S cab is mitigated with a language that has a wide user base, thus facilitating representation of semantic web approaches.

In this paper we describe our effort to develop a transformation approach for translating UML specification into equivalent OWL-S specifications. The approach relies upon the use of MDA concepts to translate XMI specifications into OWL-S via the use of XSLT transformations.

The remainder of this paper is organized as follows. Section 2 describes related work. The specifics of our approach and the detail of conversion process are presented in section 3. Section 4 presents the CongoBuy example and section 5 describes conclusion and future works.

2. Related Work

Currently, there are only a few tools that support visual development of OWL- S descriptions. These tools are outlined below. Protege is a software tool, which provides a visual environment for editing ontologies and knowledge bases [5]. The OWL plug-in, for Protege, allows for visual editing of OWL-based ontologies but does not specifically support the OWL-S ontology for services. Because the tool supports generic OWL ontologies, there is no mechanism to visualize OWL-S processes. Therefore, creating composite processes is difficult to achieve in a tool like Protege. Protégé works well for general ontology development but lacks features required to model specific OWL-S constructs and demands its own set of techniques. The learning curve for such a tool can be steep.

Elenius et al. have developed an OWL-S plug-in for Protégé [4]. The OWL-S plug-in allows for the modeling of semantic web services within the Protege environment. The plug-in will also transform a WSDL specification into a partial OWL-S specification which can then be manipulated in a visual environment. Our process focuses on the forward engineering of the semantic web service and then requires a mapping

process to ground the semantic web service to a concrete service realization. Furthermore, the goal of our research is to create tools that do not require any knowledge beyond UML to develop semantic web services and thus avoids use of environments like Protege.

The OWL-S Editor supports visual editing of OWL-S descriptions [7]. It provides a graphical user interface to create and modify an OWL-S description including all three parts: Service Profile, Service Model, and Service Grounding. The Visual Composer feature of the editor allows for the modeling of a composite process using a standard UML Activity Diagram. The OWL-S Editor also provides a mapping mechanism between WSDL and OWL-S. Our approach leverages existing skills in UML modeling which can greatly improve the efficiency of the semantic web service development workflow.

Jaeger, Engel and Geihs [6] have developed a three step process for generating OWL-S specifications by create a template using existing software artifacts (e.g. software models, WSDL), automating the identification of relevant ontologies, and performing a classification based on those ontologies. Their methodology differs from ours in that it is focused around the use of a matchmaking algorithm to identify relevant ontologies. Elements are then classified in a semantic description using those ontologies. Our work focuses on the creation of models, leveraging existing knowledge of UML, and generating a partial OWL-S specification from those models. The full specification is then created by mapping concepts in the partial description to those found in the WSDL of existing services.

3. Approach

This section describes our approach for using MDA techniques to synthesize OWL-S specifications.

3.1. Overview

In order for many new techniques and technologies to gain entry into a market, many factors must be considered. While standardization can often be a major reason why adoption of a new technology succeeds, another factor is *ease of use*. That is, if a technology adds little new overhead or requires little in the way of new training, then it is more likely to be adopted. For instance, the benefits of formal methods for software development have been described numerous times [3]. However, an adoption barrier to formal methods by much of the software development community remains in the form of a lack of education. From this standpoint, one of the overarching philosophies of the research described in this paper is the following: *Development of applications that are based on the use of semantic web services should not require knowledge beyond the use of the UML modeling language.* Specifically,it is our belief that in order for techniques that are based on the use of the semantic web and semantic web services, a bridge should be created using UML that facilitates adoption. In our work, this bridge is enabled via the use of MDA.

Web services have gained a great deal of attention as a way for organizations to use information as a commodity. While many web services currently exist, very few have semantic descriptions associated with them. Furthermore, from the viewpoint of an "end-user", a given web service may or may not provide exactly the data that is needed for a given task. However, it may be the case that a group of web services, when properly composed and integrated, will provide the desired outcomes. Based on the above, we have identified two primary requirements that are being used to drive our approach:

R1 The approach should be able to incorporate the use of both web services (e.g., services with WSDL specifications only) and semantic web services (e.g., services with semantic descriptions)

R2 The approach should facilitate composition of services to form applications or federations.

The intent of requirement *R1* is to leverage existing web services regardless of the state of their specification. Specifically, the requirement is recognition of the fact that adoption issues exist in the community and the expectation of widespread use of semantically-rich languages such as OWL-S is unreasonable. However, we do assume that at the very least, a WSDL specification does exist, which is reasonable given that frameworks such as the .NET platform provide tools that automatically generate WSDL.

Requirement *R2:* a technique for integrating services into composite services by generating an OWLS specification via the use of MDA.

The OWL-S language consists of three separate parts. The *Service Profile* is a description of what the service does. It is similar to, yet more powerful than WSDL. The *Service Model* is a description of how service works (e.g., the semantics of the service). Finally, the *Grounding Model* is a description of how to access the web service. From a WSDL description, an OWL-S description can be generated using XSLT. However, for semantic web service composition, other information is required to describe the operational behavior of the service (e.g., the service model). We are developing an approach for generating an OWL-S Service Model by taking an XML-based representation of a UML activity diagram and using XSLT to transform the diagram. In this approach, the software developer specifies the operation of a Web service as a composition of other services using UML. A transformation process then takes the UML description of the service and converts it into the OWL-S Service Model description.

3.2. Framework

Figure.1 shows the process workflow for our approach. The framework uses a model- driven development approach for specifying, mapping, and executing semantic Web services. In this regard, the framework not only allows the specification of semantic

web services but facilitates the mapping of constructs in those semantic descriptions to concrete service realizations. These concrete service realizations are transformed into executable specifications for infrastructures. In this framework, an architect is responsible for creating models using UML. The framework manages the transformation process to OWL-S with very little additional effort on behalf of the developer. This benefits the developer in two ways. First the developer is able to focus on creating models instead of writing code or in the case of semantic Web services creating a semantic specification. Secondly, the developer becomes more efficient because low-level details are abstracted away and the developers can focus more on the top-level structure and semantics. The developer can leverage existing skills in UML tools such as Poseidon [8].

The second stage of the framework involves the process of mapping concepts in the OWL-S description to concepts in the WSDL file of a concrete service realization. This process will use the profile and process model portions of the OWL-S description as well as a set of WSDL files for corresponding services and automatically generate the OWL-S grounding portion of the OWL-S description.

Once the grounding is created, the final step in the model driven development process involves execution of the OWL-S specification. In order to leverage existing technologies, a process will be created, as part of the general framework, to automatically generate an executable BPEL[10] specification. Using BPEL we can leverage existing execution engines. Semantic discovery and composition can take place utilizing the generated OWL-S specification. Once a service is composed, the BPEL specification can be used for execution. The workflow for the macro process can be seen in the diagram below: A semantic web service composition may have multiple concrete realizations. Completely separate executable specifications can be generated from these multiple groundings.

In the remainder of this paper we discuss the implementation of the first stage of this framework, a conversion process that synthesizes OWL-S specifications from UML models.

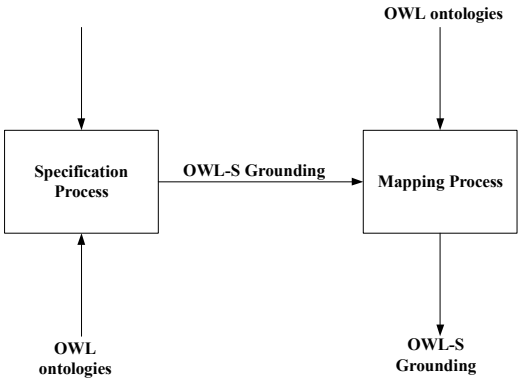

Figure 1. Framework

To facilitate the transformation process, we extended the standard UML static structure diagram by creating a UML profile in order to accommodate some of the constructs in the OWL-S specification. The UML profile includes stereotypes, custom data types and tagged values which each map to a particular construct in the OWL-S description as shown in Table 1. The leftmost column provides the abstract type represented by the constructs. The middle column in the table shows the UML constructs that are used to specify semantic services using class diagrams. Finally, the rightmost column names the corresponding target construct in OWL-S specifications. For instance, in a UML class diagram, a UML class combined with a << Service >> stereotype is used to model a service instance in an OWL-S specification, while a UML association with a << presents>> stereotype is used to identify a "presents" property in an OWL-S specification.

These UML extensions are used to facilitate the transformation process. The stereotypes help to identify which classes must be generated in the OWL-S description. Tagged values correspond primarily to property values in the output representation. The custom data types are necessary to represent the XML Schema data types. There are other ways that UML could have been utilized. For example, attributes with default values could have been used instead of tagged values (our current method); however, the latter method is conducive to the transformation process.

In general the transformation rules to map from the UML model to the OWL-S specification are straight forward. The main classes marked with the <<Service>>, <ServiceProfile>>, <<ServicModel>> and <<ServiceGrounding>> stereotypes each map into a separate output file. These constructs are linked semantically through the stereotyped associations in a diagram (e.g. <<presents>>, <<describedBy>>, and <<supports>>) and are used to generate the respective elements in output documents. Tagged values are used primarily to model properties in the OWL-S specification, especially in the ServiceProfile. Custom data types are used to model XML Schema data types that are used typically used in an OWL-S specification. At present, we allow only one service to be specified per diagram. In future work, we will expand the transformation process to allow for multiple services.

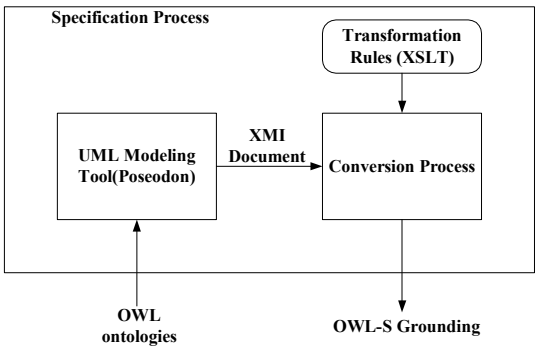

Figure 2. Basic Architecture

The basic architecture of the conversion process is shown in Figure 2. First, a UML Class Diagram is created in Poseidon [8] with the UML profile for OWL-S. The class diagram is exported in XMI format. The conversion tool is invoked from the command-line and runs multiple transformations on the input file to produce the corresponding ServiceProfile, ServiceModel Service and ServiceGrounding OWL-S documents. The transformations come in the form of XSLT transformations automatically performed on an XMI file. From the standpoint of MDA terminology, the process of creating a UML diagram can be considered equivalent to creating a platform independent model (PIM). The XSLT transformation correspond to other information necessary to generate a second level PIM in the form of an OWL-S specification. The act of specifying groundings, which is part of the framework we are developing, but outside the scope of the tool described here, corresponds to creating a platform specific model (PSM) in the sense that the mapping provides information specific to creating a specific executable implementation of a semantic service.

UML Type	UML Construct	OWL-S Construct
Class	<<Service>> stereotype	Service instance
Class	<<ServiceProfile>> stereotype	ServiceProfile instance
Class	<<ServiceModel>> stereotype	ServiceModel instance
Class	<<ServiceGrounding>> stereotype	ServiceGrounding instance
Association	<<presents>> stereotype	"presents" property of Service instance
Association	<<describedBy>> stereotype	"describedBy" property of Service instance
Association	<<supports>> stereotype	"supports" property of Service instance
Parameter	<<ConditionalOutput >> stereotype	ConditionalOutput in ServiceModel
Parameter	<<UnConditionalOutput>> stereotype	UnConditionalOutput in ServiceModel
Data Type	xsd:string	XML Schema data type
Data Type	xsd:integer	XML Schema data type
Data Type	xsd:float	XML Schema data type
Tagged Value	textDescription	textDescription property of ServiceProfile instance
Tagged Value	serviceName	serviceName property of ServiceProfile instance
Tagged Value	actor:name	name property of actor instance
Tagged Value	actor:title	title property of actor instance
Tagged Value	actor:phone	phone property of actor instance
Tagged Value	actor:fax	fax property of actor instance
Tagged Value	actor:email	email property of actor instance
Tagged Value	actor:physicalAddress	physicalAddress property of actor instance
Tagged Value	actor:webURL	webURL property of actor instance
Tagged Value	preCondition	preCondition property of AtomicProcess instance
Tagged Value	coCondition	coCondition property of ConditionalOutput instance
Tagged Value	owl:Class	Standard OWL class

Table1. UML to OWL Mapping

4. Example

The CongoBuy service is an example developed by the OWL-S working group to demonstrate the features of the OWL ontology for services. The example is a fictitious B2C company that offers a number of services including the ability for customers to buy books on the web.

4.1. Specification

Figure 3 shows a class diagram that models the CongoBuy service with the OWL-S stereotypes. Specifically, it shows the specification for an Express CongoBuyService that allows a user to directly purchase a book without using a browsing service. The class diagram represents the core structure of the OWL-S description as represented in the UML. Other classes, stereotypes and tagged values are also needed in order to generate the corresponding OWL-S documents.

The example contains seven classes, three of which are OWL classes specified with the << owl:Class>> UML stereotype. These classes are domain concepts that are relevant to the service being specified. Specifically, they are elements of ontologies pertinent to the book buying and shipping domains and act as "types" for the input and output parameters of the service. The remaining four classes specify the top level service (Express CongoBuy-Service) and its constituent profile, model, and grounding (Profile CongoBookBuying Service, ExpressCongoBuyProcessModel, and CongoBuyGrounding respectively).

The single operation in the ExpressCongoBuyProcess Model class represents an OWL-S AtomicProcess. Currently, only one Process instance is allowed per Service-Model instance. Parameters of this operation represent inputs and outputs. Those parameters marked with an "in" the UML model are inputs and those parameters marked with an "out" are outputs. Additionally, output parameters can have a Conditional-Output stereotype or Unconditional Output stereotype. A Parameter with a Conditional Output must also have a "coCondition" tagged value, which refers to a condition, which must be true for the output to be generated. Any output parameter that does not have a stereotype is considered to be an Unconditional Output. Parameters can either be XML Schema types or they can be standard OWL classes.

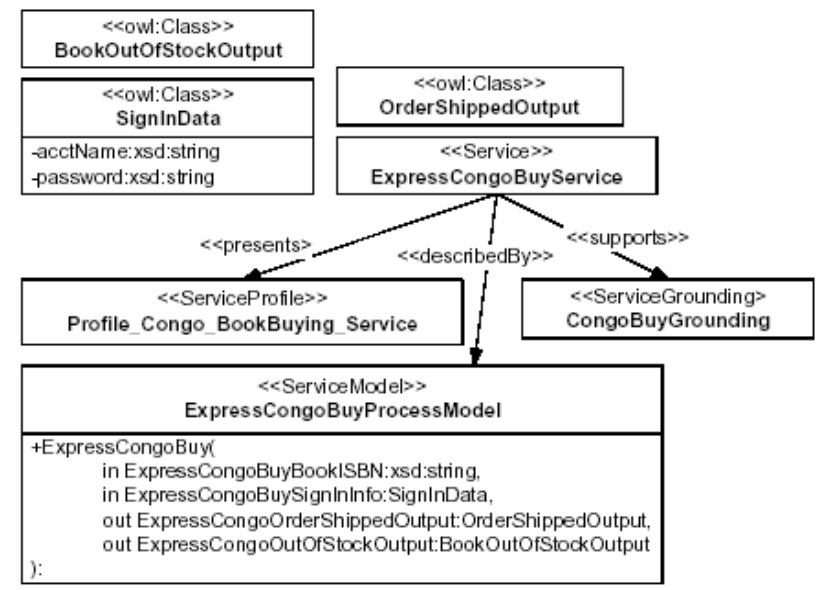

Figure 3. UML with OWL-S Stereotypes

Any parameters that are not simple data types (e.g., xsd:string, xsd:integer, xsd:float, etc.) must also have a corresponding class in the UML model or be accessible via a UML package namespace. In this example, we directly specify owl:Class stereotype instances. Any number of standard OWL classes can be created and used by the parameters in the AtomicProcess. Tagged values were used extensively to annotate the UML model for transformation. Specifically within the ServiceProfile, there were many tagged values used to store the contact information.

Not shown in the diagram are a number of other properties that are specified as part of properties of various classes, a capability provided by many UML based tools such as Poseidon. These properties are embedded in the resulting XMI representation of the classes. These properties include tagged values added to the various classes which make up the class diagram. The tag definitions and tag values come out in the exported XMI file but are not seen in the class diagram. Tagged values can be manipulated directly through the UML tool. For example, all of the information in the Profile that corresponds to <profile:contactInformation> element is implemented as a set of tagged values added to the class in the diagram with the <<ServiceProfile>> stereotype. Also, any preconditions in the process model are implemented as tagged values added to single method in the class marked with the <<ServiceModel>> stereotype. Finally, any conditional outputs in the process model are implemented as tagged values added to the corresponding method parameters of the single method in the class marked with the <<ServiceModel>> stereotype.

4.2. Transformations

Figure 4 is an excerpt of an XMI specification of a UML class that would be used as input to the conversion tool. This represents the OWL class"SignInData", a parameter type in an OWL-S Process Model presented later. Specifications such as these are transformed using XSLT transformation rules such as the ones shown in Figure 5.

```
<UML:Class xmi.id = '...'
           name = 'SignInData' visibility = 'public'
           isSpecification = 'false' isRoot = 'false'
           isLeaf = 'false' isAbstract = 'false'
           isActive = 'false'>
    <UML:ModelElement.stereotype>
        <UML:Stereotype xmi.idref = '...'/>
    </UML:ModelElement.stereotype>
    <UML:Classifier.feature>
        <UML:Attribute
            xmi.id = 'sm$1620d92:1008b11138c:-7f9d'
            name = 'acctName' visibility = 'private'
            isSpecification = 'false'
            ...>
            ...
        </UML:Attribute>
        <UML:Attribute
            xmi.id = 'sm$1620d92:1008b11138c:-7f9c'
            name = 'password' visibility = 'private'
            isSpecification = 'false'
            ...>
            ...
        </UML:Attribute>
    </UML:Classifier.feature>
</UML:Class>
```

Figure 4. XMI for SignInData class

Figure 5 contains an excerpt of the corresponding XSLT

transformation rule for OWL classes. In this case, the transformation rule looks for all classes with the owl:Class stereotype and processes them one by one. An excerpt of a synthesized OWL-S specification is shown in Figure 6. The excerpt shows part of the result of

```
<xsl:comment> OWL Classes </xsl:comment>
<xsl:for-each
   select ="//UML:Class[*//UML:Stereotype
            [@xmi.idref=$owlClassId]]">
<xsl:variable name="className"
   select="current()/@name"/>
<owl:Class rdf:ID="{$className}"/>
<xsl:if test="current()//UML:Attribute">
  <xsl:for-each select="current()//UML:Attribute">
    <xsl:if test="current()//UML:Class">
      <xsl:variable name="cId"
           select="current()//UML:Class/@xmi.idref"/>
      <xsl:variable name="cName"
          select="//UML:Class[@xmi.id=$cId]/@name"/>
       <owl:ObjectProperty rdf:ID="{@name}">
         <rdfs:domain rdf:resource="#{$className}"/>
         <rdfs:range rdf:resource="#{$cName}"/>
      </owl:ObjectProperty>
    </xsl:if>
   <xsl:if test="current()//UML:DataType">
      ...
    </xsl:choose>
   </xsl:variable>
     <owl:DatatypeProperty rdf:ID="{@name}">
          <rdfs:domain rdf:resource="#{$className}"/>
          <rdfs:range rdf:resource="{$dataType}"/>
    </owl:DatatypeProperty>
  </xsl:if>
 </xsl:for-each>
</xsl:if>
</xsl:for-each>
```

Figure 5. Excerpt of XSLT Rule

applying the transformation rules in the conversion process. The OWL class SignInData refers to a semantic concept that allows for an abstract notion of account name and password to be associated with the concrete structures of account name and password. From the perspective of the user architect, the entire transformation process is hidden behind the tools, allowing the user-architect to focus on graphical model construction.

```
<owl:Class rdf:ID="SignInData"/>
<owl:DatatypeProperty rdf:ID="acctName">
     <rdfs:domain rdf:resource="#SignInData"/>
     <rdfs:range
       rdf:resource="http://.../XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="password">
     <rdfs:domain rdf:resource="#SignInData"/>
     <rdfs:range
       rdf:resource="http://.../XMLSchema#string"/>
</owl:DatatypeProperty>
```

Figure 6. Excerpt of Synthesized OWL-S

5. Conclusions and Future Works

OWL-S provides ontology for web services, which can be used to describe the semantics of a web service. Unfortunately, adopting something like OWL-S can be difficult because of the learning curve and current state of tool support. We have created a UML profile for OWL-S in order to allow software developers to model OWL-S descriptions with a standard UML modeling tool. The conversion process described in this paper was created to take an XML representation of the UML model and convert it into an OWL-S description. Some issues that will be resolved in future work include: modeling effects for atomic processes, modeling composite processes using a behavioral diagram and for mapping process between a WSDL file and concepts in an OWL-S service model as a means for specifying an OWL-S service grounding**.**

References:

[1] E. Christensen,, F. Curbera, G. Meredith and S.Weerawarana, "Web service description language 1.1", W3C Note [Online] http:// www. w3c.org/ TR/wsdl.

[2] J. Clark, "XSL Transformations v1.0 ",W3C Recommendation.[Online]http://www.w3c.org/TR / xslt.

[3] E. W. Clarke, J. M. Wing, " Formal Methods: State of the Art and Future Directions" Technical Report CMU-CS-96-178, Carnegie Mellon University, August 1996.

[4] D. Elenius, G. Denker, D.Martin, F. Gilham, J. Khouri and S. Saddati," The OWL-S editor: a development tol for semantic web services", In Proceedings of the Second Europen Semantic Web Conference, May 2005.

[5] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy and S. W. Tu,"The evolution of Protégé: An environment for knowledge based systems development" July 2002.

[6] Michael C. Jaeger, Lars Engel and Kurt Geiths," A methodology for developing OWL-S descriptions", In First International Conference on Interoperability of Enterprise Software and Applications Workshop on Web services and Interoperability, February 2005.

[7] J. Scicluna, C. Abela and M. Montebello," Visual Modeling of OWL-S services", In Proceedings of the IADIS International Conference WWW/Internet, October 2004.

[8] Gentleware, "Poseidon for UML", [Online] Available http://www.gentleware.com/

[9] OWL services Coalition, "OWL-S: semantic markup for web services", [Online] Available http://www.daml.org/services/owls/1.0/owl-s.html.

[10] S.Weerawarana et.al, "Business Process Execution Language for web services vol1.1", [Online] http:// www-128.ibm.com/developerworks/library/bpel.