Model driven approach for specifying WSMO ontology

Djamel Amar Bensaber [1], Mimoun Malki [1]
[1] EEDIS laboratory, University of Sidi Bel Abbes , Algeria
{amarbensaber@yahoo.com, mymalki@gmail.com}

Abstract. The semantic web promises to bring automation to the areas of web service discovery, composition and invocation. In order to realize these benefits, rich semantic descriptions of web services must be created by the software developer. A steep learning curve and lack of tool support for developing such descriptions thus far have created significant adoption barriers for semantic web service technologies. In this paper, we present a model-driven architecture approach for specifying semantic web service through the use of a UML profile that extends class diagrams. In this paper we describe our efforts to develop a transformation approach based MDA to translate XMI specifications (e.g., XML encodings of UML) into equivalent WSMO specifications via the use of ATL transformations.

Keywords: Model driven Architecture (MDA), WSMO, ATL, Metamodel.

1 Introduction

The potential to achieve dynamic, scalable and cost-effective infrastructure for electronic transactions in business and public administration has driven recent research efforts towards so-called Semantic Web services, that is enriching Web services with machine-processable semantics. As a matter of fact, describing Web services through aforementioned submissions are not easy for service developers to write. Although, several tools and editors such as OWL-S Editors, WSMO studio [1], and WSMOViz [2] have been proposed to facilitate writing Semantic description, developers still need to know the concepts and syntaxes of the Semantic Web service languages. This lack of knowledge and also the complexity of these languages cause the adoption of Semantic Web services slow down [3].
In order to tackle this problem, several approaches have been proposed based on Model driven Architecture (MDA) [4] for automatically generating semantic web service descriptions from a set of graphical models. MDA is an approach presented by OMG for developing application system in the way of creating model rather than code. The portability, interoperability, and reusability are primary goals of MDA, which are acquired via separation of concerns between the implementation and specification. In most of MDA-based approaches, Unified Modeling Language (UML) [5] is used as modeling language due to its widespread adaption among software developers [6].

In this context, we are developing an approach that allows a developer to focus on creation of semantic web services and associated WSMO [7] specifications via the development of a standard UML model. We describe our efforts to develop a transformation model for translating UML specifications into equivalent WSMO specifications. The approach relies upon the use of MDA concepts by developing two metamodels (source and target one) and a transformation model to translate XMI specifications (e.g., XML encodings of UML) into WSMO via the use of ATL transformations [8]. By using transformations from equivalent UML constructs, difficulties caused by a steep learning curve for WSMO can be mitigated with a language that has a wide user base, thus facilitating adoption of semantic web approaches.

The remainder of this paper is organized as follows. Section 2 describes the related works for semantic web services approaches, a WSMO overview is presented in section 3. The specifics of our approach, details and the main parts of our solution are presented in Section 4. Sections 5 and 6 discuss implementation and conclusions, respectively.

2 Related works

In this section we present briefly various approaches which allow to use UML for the creation of ontologies. Gasevic [9] suggests using an UML profile for ontology as well as the standards of the OMG concerning the approach MDA. By this method he wishes to insure the generation automatic of complete ontologies (in OWL [10]) by using transformations of models. The approach of Gasevic and his colleagues relays on the principles of MDA and transformation of models. For it they defined an UML profile named OUP (Ontology UML Profile) which takes back the concepts of ontologies such as they are defined in OWL. Their second contribution is to supply bidirectional transformations between it profile UML and the ODM (Ontology Definition Metamodel) metamodel, proposed by the OMG. Their last contribution holds in transformations between ODM and the languages of ontology such OWL.
Brambila et al in [11] present a model-driven methodology to design and develop WSMO-based Semantic Web services using Business Process Model and Notation (BPMN) [12] in conjunction with Web Modeling Language (WebML) [13].

MIDAS-S [14] is based on the expansion of MIDAS [15] which is a model driven methodology to develop Web Information System (WIS). This approach present a methodology to develop semantic Web service based on WSMO. The four top-level elements such as ontologies, goals, mediators, and Web services are formed in the PSM level.

3 WSMO Overview

The WSMO initiative aims at providing an overarching framework for handling Semantic Web services (SWSs). WSMO identifies four main top-level elements:

1. Ontologies that provide the terminology used by other elements;
2. Goals that state the intentions that should be solved by Web Services;
3. Web Services descriptions which describe various aspects of a service;

4. Mediators: to resolve interoperability problems.

Each of these WSMO Top Level Elements can be described with non-functional properties like creator, creation date, format, language, owner, rights, source, type; etc. WSMO comprises the WSMO conceptual model, as an upper level ontology for SWS, the WSML[16] language and the WSMX [17] execution environment.

The Web Service Modeling Language (WSML) is a formalization of the WSMO ontology, providing a language within which the properties of Semantic Web Services can be described.

WSMX provides an architecture including discovery, mediation, selection, and invocation and has been designed including all required supporting components enabling an exchange of messages between requesters and the providers of services.

4 Our approach

The approach relies upon the use of MDA concepts by developing two metamodels (source and target one) and a transformation model to translate XMI specifications (e.g., XML encodings of UML) into WSMO.

Figure 1 shows the overview of our approach. The model transformation is based on ATL language, and its relates two metamodels (source:UML and target: WSMO). A transformation engine takes a source model as input, and it executes the transformation program to transform this source model into the target model. The business model is created by any UML tool, consistent with the UML metamodel (UML profile). The obtained WSML document will be exported and validated by WSMO studio Tool.

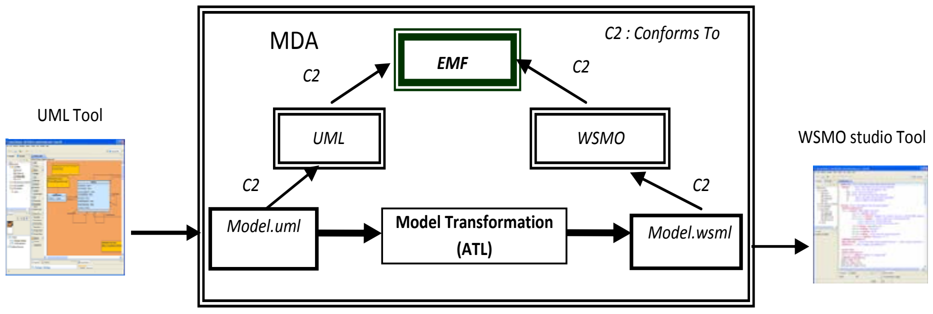


Fig.1. Architecture of our approach

4.1 The source metamodel

A UML profile [18] is a collection of stereotypes, tagged values and custom data types used to extend the capabilities of the UML modeling language. We use a UML profile to model various WSMO constructs in conjunction with the UML static structure diagram. In terms of MDA, the stereotypes, tagged values, and data types serve to mark-up the platform-independent model, or PIM, in order to facilitate transformation to WSMO specification. Stereotypes work well to distinguish different types of classes and create a meta-language on top of the standard UML class modelling constructs. Tagged values allow the developer to attach a set of name/value pairs to the model. Figure 2 shows a metamodel of our profile UML, where a group of extensions UML is introduced. The source metamodel consists of:

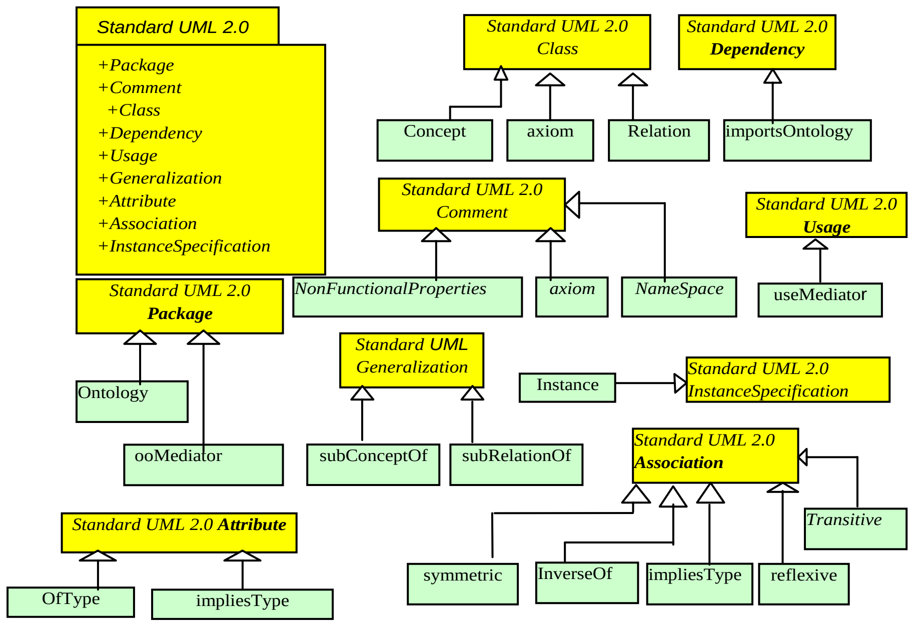


Fig.2. The source Metamodel : UML profile for WSMO

- The standard elements of UML: which are represented in the figure 2 by yellow color, we used: package, comment, Class, Dependency, Usage, Generalization, Attribute, Association, and InstanceSpecification. All these elements can be used in the class diagram for modeling the business model.
- Stereotypes: represented in the figure by the green color, they are introduced to allow the modeling of the diverse WSMO's constructs. The WSMO's constructs that we used are:

  - "Concept", " axiom ", "relation" which extend the " Class " element.
  - "Ontology", "ooMediator " which extend the "Package" element.
  - "NonFunctionalProperties","axiom" and "NameSpace" which extend the "comment" element.
  - " ImportsOntology " which extends the " Dependency " element.
  - "Instance" which extends the " InstanceSpecification " element.
  - « subConceptOf» and « subRelationOf » which extend « Generalization » element.
  - « OfType » and « impliesType » which extend « Attribute » element.

- « symmetric », « InverseOf », « impliesType », « reflexive » and « Transitive » which extend « Association » element.

4.2 The WSMO Ontology Target metamodel

This metamodel [19] is used by our transformation to generate the WSMO ontology . It consists of Ontology composed of : Concept, Relation, Axiom, Instance.
The figure 3 shows a fragment of WSMO metamodel.

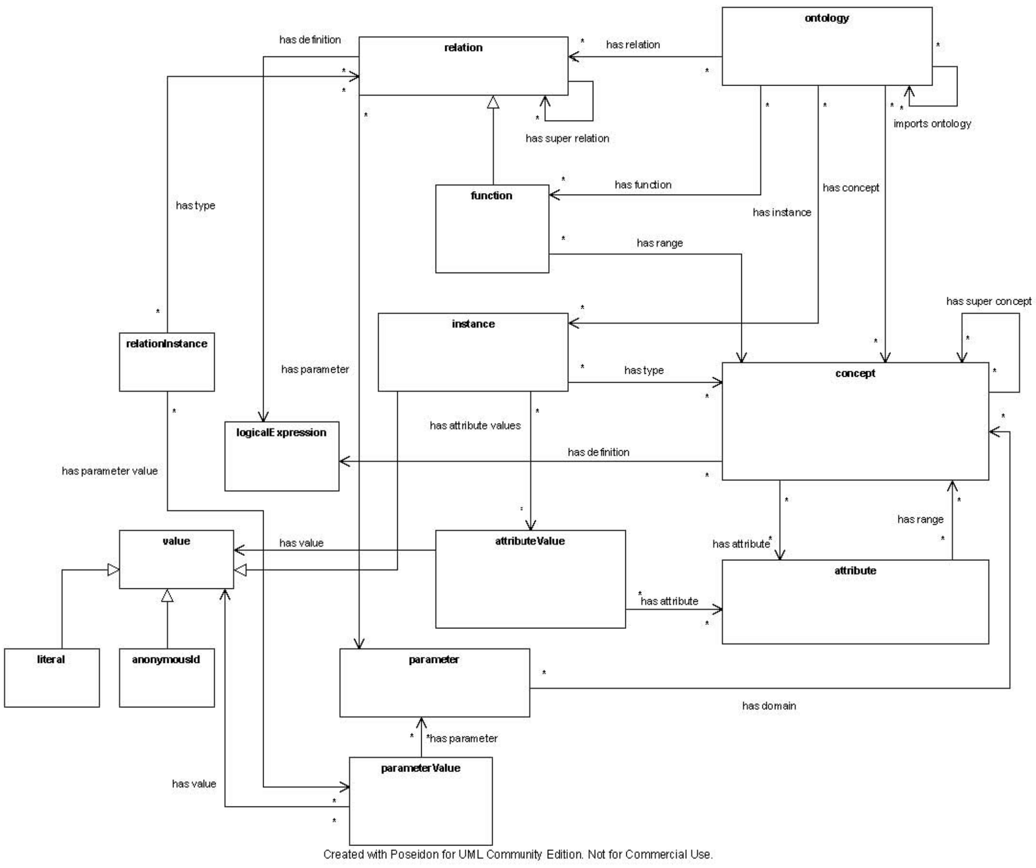


Fig.3. Fragment of WSMO Ontology metamodel

4.3 UML to WSMO transformations

4.3.1 Principle

The overview of the transformation is detailed in Figure 4. The set is split between two areas of modeling: MDE for space engineering models in which is defined the different metamodels described above and the transformation between UML and

WSMO, and WSMO space that defines the WSMO ontologies. In M3 layer we find ECORE language of metamodelisation. The both metamodels (UML and WSMO) and ATL metamodel located in M2 layer are based on ECORE.

At M1 layer we found the source model expressed in UML 2.0 conforms to our metamodel WSMO UML Profile, the model transformation UML2WSMO implemented in ATL language, WSMO ontology model resulting from the transformation process which is conforms to WSMO target metamodel in M2 layer, and a WSMO / WSML projector. This projector is a particular transformation that allows to switch from one model space to another. In our case it is used to transform the WSMO ontology in WSML document. Now we will explain in detail the transformation rules between our UML profile and WSMO ontology.

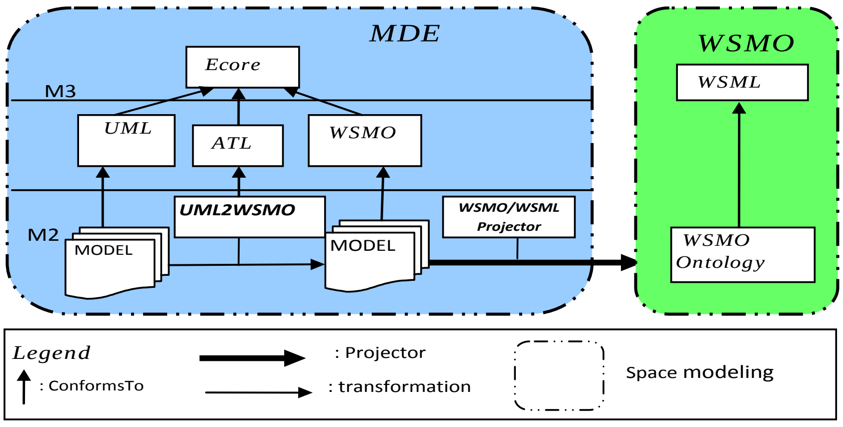


Fig. 4. UML to WSMO transformations

4.3.2 Transformation UML into WSMO

In our approach, a transformation definition is implemented in ATL language based on a mapping specification; we use the term mapping as a synonym for correspondance between the elements of two metamodels, while a transformation is the activity of transforming a source model into a target model in conformity with the transformation definition.

The source metamodel UML profile includes stereotypes, tagged values and constraints, each of which map to a particular construct in target metamodel WSMO description, as shown in table1. The left hand column provides the abstract type represented by the constructs. The middle column shows the UML constructs used to specify semantic services. Finally, the right hand two colomns name the corresponding target construct in WSMO specification and present the target elements which are defined in this transformation.

Once mappings are specified between the two metamodels (e.g. UML and WSMO), transformation definitions are implemented using transformation languages such as Atlas Transformation Language (ATL). An extract of these rules is illustrated below.

Table 1. UML to WSMO Mapping

UML TYPE	*UML Construct*	*WSMO Construct*	*Elements defined in target WSMO construct*
Package	« ontology » stereotype	**Wsmo** Ontology	URI,nonfocntionnal properties,imports ontolo gy,usesmediators,concept,relation,axiom,instances
Class	« Concept » stereotype	WSMO Concept	Concept name,subconceptof, nonfonctionnal properties, atribute
Class	« Relation » stereotype	Wsmo relation	Relationname, nonfonctionnalproperties, para meters,subrelationof
Class	« axiom » stereotype	Wsmo axiom	Name, Type,className,attributeName
Attribute	« oftype » attribute	Wsmo attribute	Name, Type,className,attributeName
Association	« implies type » stereotype	Wsmo attribute	Name, Type,className,attributeName
Association	« transitive_impliesType » stere.	WSMO attribute	Name, Type,className,attributeName
Association	« symmetric_impliesType »stere.	Wsmo attribute	Name, Type,className,attributeName
Dependency	« importsOntology» stereotype	WSMO import ontology	supplierName, clientName
Depenency	« usemediator» stereotype	Wsmo usemediator	supplierName, clientName
Comment	« nonfonctionalProperties» stereo.	Wsmo nonfonctional Properties	Name, body
Comment	« namespace » stereotype	Wsmo namespace	Name, body
Comment	« axiom » stereotype	Wsmo axiom	Name, expression_definition
Instance Specification	« instance Speciication» stere.	Wsmo Instance	Instance Name, memberOf, attributeValue
Instance Specification	« instanceproperty » stereotype	Wsmo attribute value	Instance Name,Instance, attributeNname

- **Rule : UMLClass2WSMOConcept**

This rule allows to create a concept WSMO from a class UML stereotyped "Concept". Any class UML stereotyped "Concept" is transformed into WSMO concept. This one is defined by the concepts from which it inherits “subConceptOf” , by a Name, NonFunctionalProperties and by these attributes.

```
rule UMLClass2WSMOconcept {
    from   s : UML!"uml::Class" (s.hasStereotype('WSMO_Profil::Concept'))
    to    t : b_Ontology_WSMO!Concept (
                Nom_concept <- s.name->debug('Cette Class Est Un Concept '),
                SubConceptOf <- if s.general.oclIsUndefined()then ''
                     else s.general->collect(a|a.name).first() endif,
                  NonFunctionalProperties<-
b_Ontology_WSMO!NonFunctionalProperties.allInstances() ->select(b|b.Nom_De = s.name)->collect(b1|b1.Corps).first(),
                  Attribute  <-  b_Ontologie_WSMO!Attribute.allInstances() ->
                    select(b|b.Nom_De_Class = s.name)->collect(b1|b1.Nom_Attribute) )  }
```

The helper " hasStereotype " receives a string and returns a boolean. It is used to know if the current UML element is stereotyped as the string taken in parameter.

```
helper context UML!"uml::Element" def: hasStereotype(name : String) : Boolean =
        self.getAppliedStereotype(name)->oclIsKindOf(UML!Stereotype);
```

- **Rule : Property2Attribute**

This rule allows to create attributes WSMO from UML properties stereotyped " OfType ". Any property UML stereotyped "OfType" is transformed into WSMO attribute. This one is defined by a Name, Type, class names and Attribute name.

```
rule Property2Attribut
{  from     P : UML!Property ( P.hasStereotype('WSMO_Profil::ofType'))
   to       A : b_Ontology_WSMO!Attribute (
                 Nom_Attribute <- P.name + ' ofType' + ' ' +
               P.type.toString().substring(4,P.type.toString().size()),
               Type_Attribute <- P.type.toString().substring(4,P.type.toString().size()),
             Nom_De_Class <- P.class.name  ),
             At : b_Ontologie_WSMO!AttRelation (
            Nom_Attribute <- ' ofType ' +P.type.toString().substring(4,P.type.toString().size()),
             --Type_Attribute <- P.type,
            Nom_De_Class <- P.class.name )   }
```

- **Rule : InstanceSpecification2WSMOInstance**

This rule transforms UML instance into WSMO instance. This one is defined by the classes that are " MemberOf", InstanceName and AttributeValues.

```
rule UMLInstance2WSMOInstance
{     from       I : UML!InstanceSpecification
    to     Ins : b_Ontology_WSMO!Instance  (   Nom_Instance <- I.name,
        MemberOf <- I.classifier->collect(a|a.name),
        AttributeValues  <-  b_Ontology_WSMO!AttributeValue.allInstances( )
      ->select(b|b.name = I.name)->collect(a|a.Nom_Att_Ins)  )   }
```

4.3.3 WSMO2WSML Projector

A projector consists of one or several transformations allowing realizing the projection of an artefact belonging to a technological space towards another. In our case, the artefact is a model in compliance with the WSMO metamodel, belonging to the technical space of MDE. We aim to project this artefact towards the WSMO space in WSML syntax. A model in the MDE space is serialized in the XMI format, whereas a document in the WSMO space is in WSML format. The projector implemented here includes a single ATL query file allowing the transformation of WSMO model into a WSML document. Among the main rules which compose the transformation file named Ecore2wsmo, we quote the first rule:

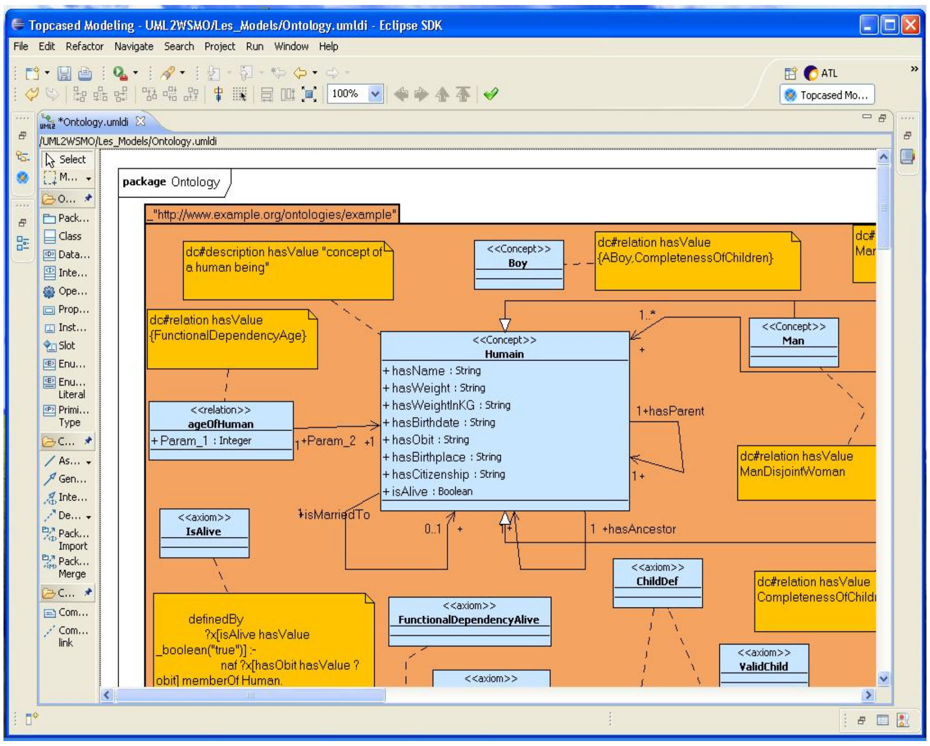
- **Rule 1** : This rule allows to translate the "concept" element of WSMO modeled in ECORE Format into WSML syntax.

```
helper context UML!Concept def : toString_b() : String =
        '\n'+' concept ' + self.Nom_concept + if self.SubConceptOf->iterate(e; acc : String =
'' |acc + e.toString()+ '') = 'OclUndefined'   then '\n'
```

```
                                        else
' subConceptOf '+self.SubConceptOf->iterate(e; acc : String =' ' |acc + e.toString() +' ') + '\n'
                                        endif
     + if self.NonFunctionalProperties.oclIsUndefined()    then ''
                 else ' nonFunctionalProperties '+ '\n' + self.NonFunctionalProperties + '
endNonFunctionalProperties ' + '\n'      endif
     + if self.Attribute.size() <> 0
      then self.Attribute->iterate(e; acc : String = '' |acc + e.toString() + '\n') + '\n'
      else ''      endif;
```

5 Implementation

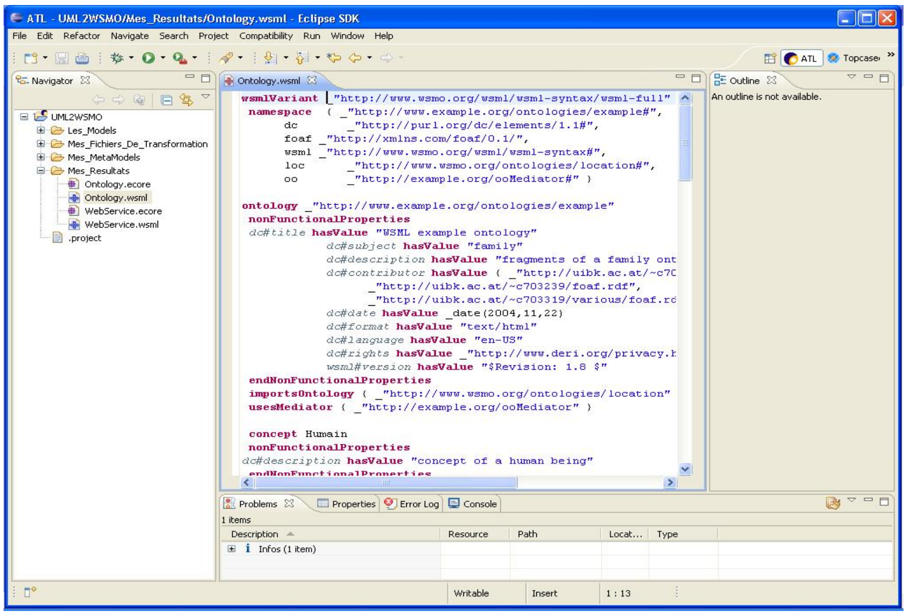
To show the feasibility of our approach, we have developed a tool on the Eclipse environment (Eclipse Ganymade 3.4.2), an ATL project is created, the UML metamodel profile is modeled by UML diagrams 2.1 Integrated in EMF eclipse, and the WSMO metamodel is modeled by the Ecore language (Ecore diagram), then two ATL files containing the transformation rules (UML2WSMO and WSMO2WSML) are written in ATL. For each execution, we introduce a source model conforms to our source metamodel and we obtain a WSMO document.



Topcased Modeling - UML2WSMO/Les_Models/Ontology.umldi - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
ATL
Topcased Mo...
*Ontology.umldi
/UML2WSMO/Les_Models/Ontology.umldi
Select
Class
Slot
100%
package Ontology
_"http://www.example.org/ontologies/example"
dc#description hasValue "concept of a human being"
<<Concept>> Boy
dc#relation hasValue {ABoy,CompletenessOfChildren}
dc#relation hasValue {FunctionalDependencyAge}
<<Concept>> Humain
+hasName : String
+hasWeight : String
+hasWeightInKG : String
+hasBirthdate : String
+hasObit : String
+hasBirthplace : String
+hasCitizenship : String
+isAlive : Boolean
<<Concept>> Man
<<relation>> ageOfHuman
+Param_1 : Integer
+Param_2
1+hasParent
dc#relation hasValue ManDisjointWoman
<<axiom>> IsAlive
+isMarriedTo
1 +hasAncestor
<<axiom>> ChildDef
dc#relation hasValue CompletenessOfChild
definedBy ?x[isAlive hasValue _boolean("true")] :- naf ?x[hasObit hasValue ?obit] memberOf Human.
<<axiom>> FunctionalDependencyAlive
<<axiom>>
<<axiom>> ValidChild

Fig.5. Class diagram of source model

To illustrate our approach, we take the human ontology as an example, the source model is expressed in a UML2.0 class diagram in conformity with our source metamodel (see Fig.5). The transformation engine takes the source model serialized in XMI [20] format as input and executes the transformation rules contained in UML2WSMO ATL file that generates a WSMO ontology in Ecore format, then the projector tool executes the ATL query file WSMO2WSML to transform the resulting WSMO ontology into WSML format. The obtained WSML document of the human ontology is depicted in Fig. 6. At the end, the output of our system is imported by the WSMO STUDIO tool to validate the correctness of our transformations.



```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-full"
 namespace  { _"http://www.example.org/ontologies/example#",
      dc        _"http://purl.org/dc/elements/1.1#",
      foaf _"http://xmlns.com/foaf/0.1/",
      wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
      loc       _"http://www.wsmo.org/ontologies/location#",
      oo        _"http://example.org/ooMediator#" }

ontology _"http://www.example.org/ontologies/example"
 nonFunctionalProperties
 dc#title hasValue "WSML example ontology"
            dc#subject hasValue "family"
            dc#description hasValue "fragments of a family ont
            dc#contributor hasValue { _"http://uibk.ac.at/~c70
                  _"http://uibk.ac.at/~c703239/foaf.rdf",
                  _"http://uibk.ac.at/~c703319/various/foaf.rc
            dc#date hasValue _date(2004,11,22)
            dc#format hasValue "text/html"
            dc#language hasValue "en-US"
            dc#rights hasValue _"http://www.deri.org/privacy.h
            wsml#version hasValue "$Revision: 1.8 $"
 endNonFunctionalProperties
 importsOntology { _"http://www.wsmo.org/ontologies/location"
 usesMediator { _"http://example.org/ooMediator" }

 concept Humain
 nonFunctionalProperties
dc#description hasValue "concept of a human being"
```

```
1 items
Description        Resource   Path   Locat...   Type
Infos (1 item)
```

Fig.6. The resulting Human ontology file in WSML format

6 Conclusions and Future Work

Semantic Web Services can potentially change the way software is both developed and used. In order to realize that great promise, the software development community must embrace the technology. The barriers to adoption must be bridged in a manner that leverages the capabilities of developers.

We have been developing an MDA-based approach for facilitating such adoption by using Metamodels at PIM levels in such way that it more adequately hides the details of semantic web service technologies and allows the developer to focus on creating models of semantics web services, Then a transformation model consisting of a defined set of rules that specify the correspondence between the elements of both source and target metamodels, is applied to generate a WSMO ontology encoded in WSML.

Our current investigations involve developing the framework in such a way that it improves the transformation model by handling the logical expressions to cover "axiom" and covers the other parts of WSMO like capability, mediators and goals.

7 References

1. "WSMO Studio." vol. 2010, 2009: http://www.wsmostudio.org.
2. Kerrigan M.: "WSMOViz: An ontology visualization approach for WSMO," London, United kingdom, 2006, pp. 411-416.
3. Timm,, J.T.E.: "A model-driven framework for the specification, grounding, and execution of semantic Web services." vol. PH.D: Arizona State University, 2008, p. 170.
4. Mukerji J.M.: "MDA Guide Version 1.0.1." vol. 2010: OMG Group, 2003.
5. O. M. Group, "OMG Unified Modeling Language (UML)." vol. 2010: Object
6. Wang Q., Garousi V., Madachy R., Pfahl D., Bendraou R., J.-M. Jezéquél J.M. and F. Fleurey F. : "Combining Aspect and Model-Driven Engineering Approaches for Software Process Modeling and Execution," in Trustworthy Software Development Processes. vol. 5543: Springer Berlin / Heidelberg, 2009, pp. 148-160.
7. Roman D., Lausen H., and Keller U.. Web service modeling ontology (WSMO). Final Draft D2v1.3, WSMO, 2006. Available from:http://www.wsmo.org/TR/d2/v1.3/.
8. Jouault, F., Kurtev, I. : Transforming Models with ATL, Proceedings of the Model Transformations in Practice Workshop at MoDELS'05, Montego Bay, Jamaica, 2005.
9. Gasevic, D., Djuric, D., Devedzic, V. : MDA-based automatic OWL ontology development, International Journal on Software Tools for Technology Transfer, June 2006.
10. Hori, M., Euzenat, J., Patel-Schneider, P., F. : OWL (Web Ontology Language) XML Presentation Syntax W3C Note 11 June 2003, http://www.w3.org/TR/owl-xmlsyntax/
11. Brambilla M., Ceri S., Facca F.M. , Celino I., Cerizza D., and Valle E.D.: "Model-driven design and development of semantic Web service applications," ACM Transactions on Internet Technology, vol. 8, 2007.
12. "Business Process Model and Notation (BPMN)," in Version 1.2. vol. 2010: OMG, 2009.
13. Stefano C., Piero F., Aldo B., Marco B., Sara C., and Maristella M. :Designing Data-Intensive Web Applications: Morgan Kaufmann Publishers Inc., 2002.
14. Sanchez D.M. , Acuna C.J., Cavero J.M. , and Marcos E. : "Toward UML-Compliant semantic web services development," International Journal of Enterprise Information Systems, vol. 6, pp. 44-56.
15. Cáceres P., Marcos, E., Vela, B. A. :"MDA-Based Approach for Web Information System Development," in Workshop in Software Model Engineering, 2003.
16. Lausen H., Bruijn J., Polleres A., and Fensel D. : WSML - A Language Framework forSemantic Web Services. In Proc. of the W3C Workshop on Rule Languages for Interoperability, Washington DC,USA,2005.
17. Haller A., Cimpian E., Mocan A., Oren E., and Bussler C. WSMX - A Semantic ServiceOriented Architecture. In Proceedings of the International Conference on Web Service (ICWS 2005), 2005.
18. Atkinson, C. and Kühne, T.: Profiles in a strict metamodeling framework, Science of Comp.Prog., Vol. 44, No. 1 (2002) 5-22
19. http://www.w3.org/Submission/2005/SUBM-WSMO-20050603/.
20. OMG. XML Metadata Interchange (XMI) specification, version 2.0, formal/03/05/02, may 2003.