International Journal of Computer Applications (0975 – 8887)
Volume 7– No.3, September 2010

Model Transformation from Ontology Model to Content Analysis Model

Wafaa Alakwaa
Computer Science Dept.
Cairo University
Giza , Cairo 12613, Egypt

Akram Salah
Computer Science Dept.
Cairo University
Giza , Cairo 12613, Egypt

ABSTRACT

This paper introduces a technique to extend existing Web engineering methodologies to develop the semantic web applications. We investigate the use of ontology in the domain analysis for the development of web applications. The contribution of this paper is the automatic generation of Content Analysis Model from the Ontology Model. This technique makes a straight forward mapping between Ontology Definition Metamodel (ODM) model's elements and the Content Analysis Model's elements. We further show how this technique could be integrated with many web engineering methodologies, which are based on systematic and automatic chain of transformation during all the development phases. The requirement model that is generated from the Ontology model guarantees that an application terminologies are unified all over the web engineering process.

General Terms

Semantic web, Ontology , Ontology Modeling, Model Transformation , Web Engineering.

Keywords

Unified Modeling Language (UML),Model Driven Architecture(MDA), Ontology, Semantic Web, Ontology Definition Metamodel(ODM), Web Ontology Language (OWL), Web Engineering.

1. INTRODUCTION

Software development techniques are continuously evolving with the goal of solving the main problems that still affect the building and maintenance of software systems: time, costs and error-proneness. Model-driven development (MDD) approaches aim to reduce at least some of these problems providing techniques for the construction of models and the specification of transformation rules, tool support, and automatic generation of code and documentation. The method of resolution of MDD is to first build models, which are independent of the platform, transforming them in later stages to technological dependent models, and to achieve automatic model and code generation based on transformation rules [1].

Web applications vary widely: from small-scale, short-lived services to large-scale enterprise applications distributed across the Internet and corporate intranet. Web Engineering is a new area of Software Engineering, which focuses on the development of Web Systems. It concerns applying systematic, disciplined and quantifiable approaches to develop, operate, and maintain of such Web applications [3]. Several approaches have been proposed for the Web engineering process. These methods provide specific modeling elements for the analysis and design and most of them define a proprietary notation used for the graphical representation of the elements [4].

Most of these Web engineering methodologies are based on separation-of-concerns to define strict roles in the development process and to enable parallel development . The most frequently used models are the content model , the navigation model and presentation model [4].

Ontologies provide shared domain conceptualizations representing knowledge, by concepts , their properties and relations between these concepts, to model the problem domain as well as the solution domain. The Web Ontology Language (OWL)is the most prominent for Semantic Web applications among ontology languages, providing a class definition language for ontologies [5].

To be able to design a Semantic Web application, a new model is introduced, the ontology model. In this paper we show how this new model can be used to extend current Web engineering techniques to develop Semantic Web applications. We call the engineering of ontology based Web applications a " Semantic Web Engineering". The contribution of this paper is the automatic generation of Content Analysis Model from the Ontology Model. Most of the current web engineering methodology propose specific processes to support the systematic or semiautomatic development of Web applications. However, most of the existing web methodologies start the development with modeling requirement and few of the existing Web methodologies start the development cycle with a detailed requirements analysis.

In this paper we propose a semantic web engineering, which starts with the development of ontology model. This model is built using MDA standards technique that were proposed by many researchers. Here we use our profile that was originally created to handle the practical implementation of ontology concepts with a straight forward mapping between ontology and Object oriented concepts. From this ontology model, the Content Analysis Model is being automatically generated. This transformation will lead to create an application that is clear of any ambiguous, with no redundant terminologies. Requirements that are automatic generated from the ontology model, will have no contradiction in its content. Designer and developers then will not use different terminologies for the same concept, neither apply the same term for more than one entity.

The rest of the paper will be organized as follows, section 2 will give an overview of the model driven development and the main Web engineering concepts. An overview about Semantic web and ontology modeling will be shown in section Our methodology to model ontology using class diagram, with our developed UML profile will be shown in section .. Then in section 4, will show the model to model transformation techniques and languages. Section 5 will show the transformation steps with one-to-one mapping between Ontology Definition Metamodel (ODM) and UML Metamodel concepts. Finally section 6 concludes our work and mention similar approaches to use ontology in the web engineering process.

2. OVERVIEW OF MODEL DRIVEN DEVELOPMENT ,WEB ENGINEERING AND REQUIREMENT ANALYSIS

Model-Driven Software Development (MDSD) is becoming a widely accepted approach for developing complex distributed applications. MDSD advocates the use of models as the key artifacts in all phases of development, from system specification and analysis, to design and implementation. They focus on the construction of models, specification of transformation rules, tool support and automatic generation of code and documentation. The central idea of MDD is to separate the platform independent design from the platform specific implementation of applications delaying as much as possible the dependence on specific technologies. Therefore, MDD advocates the construction of platform independent models and the support of model transformations. Software development process then can be viewed as a chain of model transformations [1].

A model is a coherent set of formal elements describing something built for some purpose that is amenable to a particular form of analysis. A model is a simplified representation of a software system and is useful if it allows for a better understanding of the system. Models are built to offer different views of a same system. These views need to be refined and integrated and used to produce code, when possible in an automated way, i.e. with the help of transformation rules. Models are represented using a modeling language. The goal of MDD can be summarized as to provide better separation of concerns, automatic generation of models and code, and traceability between code and models [6].

Web Engineering is a specific domain in which MDD can be successfully applied. Existing model-driven Web engineering (MDWE) approaches already provide excellent methodologies and tools for the design and development of Web applications. They address different concerns using separate models (navigation, presentation, content, etc.), and are supported by model transformation that produce most of the application's Web pages and logic based on the models [1]. However, most of the existing web methodologies start the development with modeling requirement and few of the existing Web methodologies start the development cycle with a detailed requirements analysis [4].

Web engineering community has proposed several languages, architectures, methods and processes for the development of Web applications. In particular, methods for modeling such systems were developed, for example Hera , OOHDM , OO-H, OOWS , UWE , WebML , and W2000 . They focus on the specification of analysis and design models for Web systems, for instance on the construction of navigation and adaptation models. However, the model transformation aspects were neglected by most of these methods [2].

Requirements play a key role in the development of Web applications. But they are often not described properly and may be specified in an ambiguous, vague, or incorrect manner. Typical consequences of poor requirements are low user acceptance, planning failures, or inadequate software architectures. There exists two main type of requirements, functional and nonfunctional requirements. Functional Requirements (FR) specify the capabilities and services a system is supposed to offer. Functional requirement are categorized to many classes, Data requirements, Interface requirements, Navigational requirements, Personalization requirements and Transactional requirements. Data requirements, which is an important type of the functional requirement, also known as conceptual requirements or content requirements, establish how content is represented as a model, showing relationships between concepts and properties accompanied with each concept related to the application domain. Non-functional requirements act to constraint the solution, e.g. portability requirements; reuse requirements, usability requirements, availability requirements, performance requirements, etc. [7].

UML techniques are being used in the production of a requirements analysis model for web applications. UML is now fast becoming an industry standard, has OMG (Object Management Group) acceptance, and a rich set of resources and software development tools available [8].

Some UML-based methodologies suggest starting the analysis process with requirement modeling. They starts the development with creating a class diagram describing the real world entities and concepts in the problem domain, using the name Domain Model for this preliminary class diagram. The general class diagram, which describes the domain is an important basis and a glossary for creating use cases that describe the functional requirements [9].

The study in [10] investigated the possible synergetic values and relationships between the use case and class diagrams in the context of requirements analysis. This study used theories from cognitive psychology as its theoretical and conceptual foundation. The results showed that the use case diagrams and class diagrams depict different aspects of the problem domain, they have very little overlap in the information captured, and both are necessary in requirements analysis.

The experiment study shown in [9] expected that creating a class diagram prior to defining the functional requirements with use cases should yield better results, i.e. better class diagrams and better use cases. This is because objects are more "tangible"/"stable" than use cases; users can identify and describe more easily the objects they are dealing with and their attributes than functions or use cases of the sought system. On the other hand, functions are not "tangible" and may be vague.

3. OVERVIEW OF SEMANTIC WEB AND OWL ONTOLOGY.

The goal of the semantic web is to be "a web talking to machines", i.e. in which machines can provide a better help to people because they can take advantage of the content of the Web. The information on the web should thus be expressed in a meaningful way accessible to computers. The semantic web can also be thought of as an infrastructure for supplying the web with formalized knowledge in addition to its actual informal content [11].
An ontology expresses, for a particular domain, the set of terms, entities, objects, classes and the relationships between them, and provides formal definitions and axioms that constrain the interpretation of these terms. An ontology permits a rich variety of structural and nonstructural relationships, such as generalization, inheritance, aggregation, and instantiation and can supply a precise domain model for software applications [12]. Ontologies are central to the semantic web because they allow applications to agree on the terms that they use when communicating. They are a key factor for enabling interoperability in the semantic web. Ontologies will have to grow and develop with the semantic web and this needs support. Ontologies aim at modeling and structuring domain knowledge that provides a commonly agreed understanding of a domain, which may be reused and shared across applications and groups of people [11].

The semantic Web architecture is a functional, non-fixed architecture [13]. Barnes-Lee defined three distinct levels that incrementally introduce expressive primitives: metadata layer, schema layer and logical layer Languages that support this architecture [14]. Figure 1 shows the main 3 layers of the semantic web architecture, where each of these layers is based on a technology that plays a distinct role in deploying and reusing learning objects on the Semantic Web. Metadata layer based on XML and RDF, schema layer based on RDF schema and finally the logical layer that is based on OWL [15].

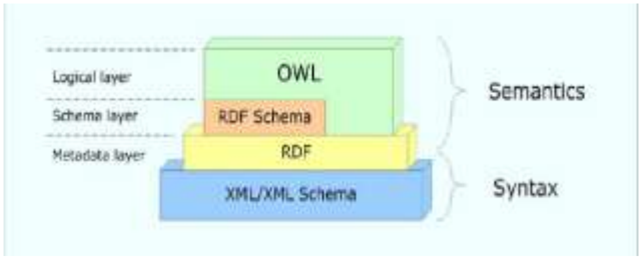

Figure 1: OWL in the Semantic Web Architecture [16].

The Web Ontology Language, which is a W3C effort, is the recent and complete language for describing ontology. The OWL language provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users. The OWL Lite supports the primarily classification hierarchy and simple constraint features. OWL DL supports the maximum expressiveness without losing computational completeness and decidability of reasoning systems. OWL Full is meant for maximum expressiveness and the syntactic freedom of RDF with no computational guarantees [17].

3.1 Ontology Model.

The importance and use of ontology was expanded from being a basic building block of the Semantic web [18], to participate in many software applications and the critical semantic foundation for many rapidly expanding technologies such as software agents, e-commerce and knowledge management [19]. This importance caused for many new tools to be developed to accelerate and aid in building , representation, design and construction of domain ontologies [20]. Most of the current Semantic Web ontologies are developed in AI laboratories. Because of this, the use of ontologies by Software engineers professionals and researchers can be seen as an additional learning experience, and in some cases, of considerably great effort.

Researchers have investigated that a strong coupling exists between the knowledge engineering and software engineering phases of a knowledge-based system. These researches tried to converge between MDA standards and ontology developments. Applying MDA techniques in developing ontologies has been discussed, focusing on what is common among them [21]. The Object Management group (OMG) ,as a consortium which develops standards for various aspects of software engineering which are widely used in industry including UML, has published a RFP (Request for Proposal) that tries to define a suitable language for modeling Semantic Web ontology languages in the context of MDA. This RFP was responsible for modeling Web Ontology Language, which is a W3C effort. This metamodel will make ontology being used in a computing application. Ontology then could be represented as some sort of computer-readable data structure [22].

For this metamodel to be used with UML tools, we have been adapted a UML profile named “Ontology Modeling profile” [33], so all the ODM concepts can be mapped directly to the UML metamodel concepts, especially for those related to statement and individuals. For more detail about our profile reader refers to [33]

4. MODEL TO MODEL TRANSFORMATION.

Transformations are vital for the success of the MDA approach. Expressed exaggeratedly, transformations lift the purpose of models from documentation to first class artifacts of the development process [23].
Model-to-model transformations translate between source and target models, which can be instances of the same or different metamodels. Most existing MDA tools provide only model-to-code transformation, which they use for generating PSMs (in this case being just the implementation code) from PIMs. Model to Model transformation can be used when bridging large abstraction gaps between PIMs and PSMs. It facilitate generating intermediate models rather than go straight to the target PSM [24].

The basic idea of model transformation is presented in Figure 2 where (at the bottom) a transformation operation Mt takes a model Ma as the source model and produces a model Mb as the target model. This operation Mt is probably the most important operation in model engineering. Being models, Ma and Mb conform to metamodels MMa and MMb. Usually, the transformation Mt has complete knowledge of the source metamodel MMa and the target metamodel MMb. Furthermore, the metamodels MMa and MMb conform to a metametamodel, such as the OMG's MOF which in turn conforms to itself [25].

Czarnecki et al. propose a possible taxonomy for model transformation approaches [24]. We will only discuss here the model to model transformation classification approaches. Direct-manipulation approaches can access an internal model representation via an Application Programming Interface (API) for a particular programming language, such as Java. Relational approaches are declarative approaches based on mathematical relations. Basically, a relation is specified by defining constraints over the source and target elements of a transformation. QVT and ATL support the relational approach and additionally provide imperative constructs, i.e. they are hybrid approaches. Graph-Transformation-Based Approaches are declarative approaches based on the theoretical work on graph transformations. Typed, attributed, labeled graphs are particularly suitable to represent UML-like models. Structure-Driven Approaches have two distinct phases: the first phase is concerned with creating the hierarchical structure of the target model, whereas the second phase sets the attributes and references in the target. The overall framework determines the scheduling and application strategy; users are only concerned with providing the transformation rules.Hybrid approaches combine different techniques from the previous categories. In a hybrid rule, the source and/or target pattern are complemented with a block of imperative logic, which is run after the application of the target pattern [24].

The Atlas Transformation Language (ATL) is a hybrid approach. ATL is a hybrid language, i.e. it provides a mix of declarative and imperative constructs. The LHS of a fully declarative rule (so-called source pattern) consist of a set of syntactically typed variables with an optional OCL constraint as a filter or navigation logic. The RHS of a fully declarative rule (so-called target pattern) contains a set of variables and some declarative logic to bind the values of the attributes in the target elements [24].

ATL Development Tools (ADT) is developed under the ATL Eclipse/GMT subproject. ADT is composed of the ATL transformation engine and the ATL Integrated Development Environment (IDE): an editor, a compiler and a debugger. ATL contains a mixture of declarative and imperative constructs [24].

ATL is applied in a transformational pattern shown in Figure 2. In this pattern a source model Ma is transformed into a target model Mb. The transformation is driven by a transformation definition (or a transformation program)written in the ATL language. The transformation definition is a model. The source and target models and the transformation definition conform to their metamodels MMa, MMb, and ATL respectively. The metamodels conform to the MOF meta-meta-model [27].

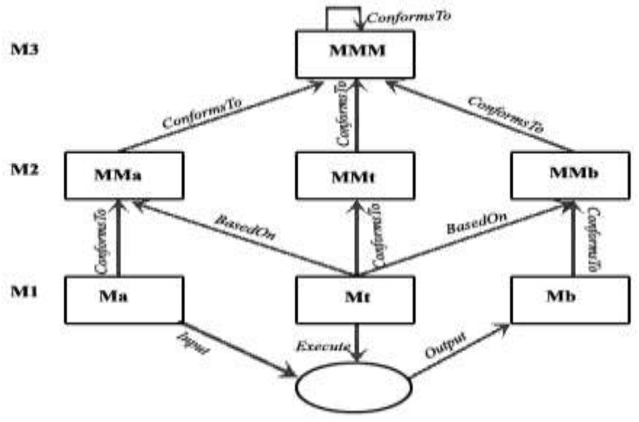

Figure2 : Model transformation pattern\cite{ATL08}}

ATL is inspired by the OMG QVT requirements and builds upon the OCL formalism. The choice of using OCL is motivated by its wide adoption in MDE and the fact that it is a standard language supported by OMG and the major tool vendors. ATL is a hybrid language, i.e. it provides a mix of declarative and imperative constructs [27].

5. TRANSFORMING ONTOLOGY MODEL TO REQUIREMENTS MODEL

Based on the Ontology Modeling Profile given in Section 3.1, we define our approach of deriving the system requirement models from its ontology models. The approach is based on metamodel mappings, i.e. transformations rules are defined to map the ontology model elements from the ontology Modeling Profile elements.

The transformation implements the MDA model transformation pattern, as shown in Figure 3. Both metamodels are specified using the MOF language, which is also an OMG standard. Figure 3 shows how a requirement model is derived from an ontology model by means of the metamodel-based transformations. Note that the generated requirement model is a first draft as its completion may require additional information, partially depending on the developer's decisions.

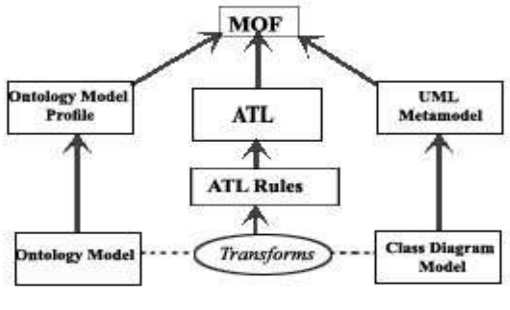

Figure 3: Transformation from Ontology Model to Requirement Model with ATL language.

In an ontology model, the main block is a "graph", which is mapped to a "model" in a requirement model. An "ontology", in an ontology model serves as a "package" containing related concepts and their interrelations with each others. "OWLclass" refers to every concept that is worth to be modeled in an ontology model, we map it to a UML "class" in a requirement model. For more clarity, we have implemented the museum concepts as an example to show how concepts are mapped from Ontology model to the requirement model. In the museum example, from the owl concepts modeled using the Ontology modeling profile as shown in Figure 4,we have generated the classes (painter, painting, museum, artifact and artist) as a content analysis UML class diagram, see Figure 5.

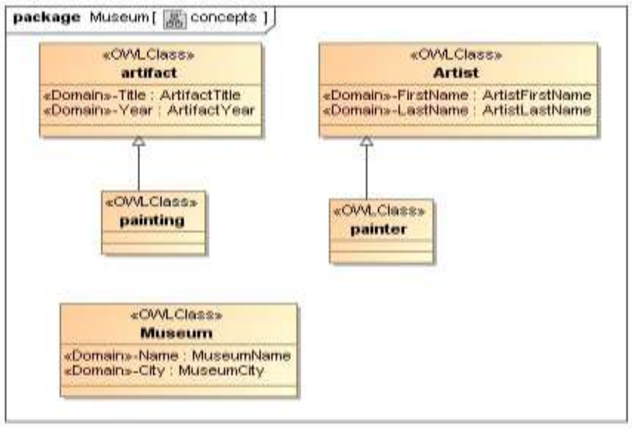
package Museum [concepts]
«OWLClass»
artifact
«Domain»-Title : ArtifactTitle
«Domain»-Year : ArtifactYear
«OWLClass»
Artist
«Domain»-FirstName : ArtistFirstName
«Domain»-LastName : ArtistLastName
«OWLClass»
painting
«OWLClass»
painter
«OWLClass»
Museum
«Domain»-Name : MuseumName
«Domain»-City : MuseumCity

Figure 4: Ontology Model : partial OWL classes in the Museum example

In ATL, the rule that mapped the OWLClass to the UML class is simple as shown in the rule named OWLClass below:

```
rule OWLClass{
from oc : ML!Class(oc.hasStereotype('OWLClass') )
to c : UML!Class
name <- oc.name
isAbstract <- oc.isAbstract      }
```

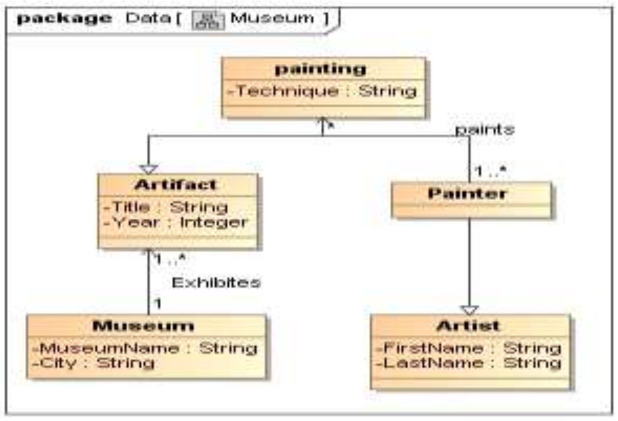

Figure 5:T he resulted classes , attributes and associations in the Requirement Model of the museum example.

DatatypeProperties in an ontology model refer to a relation between the owlclasses (the property's domain) and a primitive types (e.g. Integer, Float, Boolean , String,..etc) which is the property range). In a Content Analysis model, we map them to attributes of a class in the UML class diagram. Attributes in the requirement model are being deduced from many resources. One of them is from the datatypeProperties , as explained above. These datatypeProperties are attached to the owl classes via domain and range stereotypes properties as shown in Figure 4 and Figure 6.

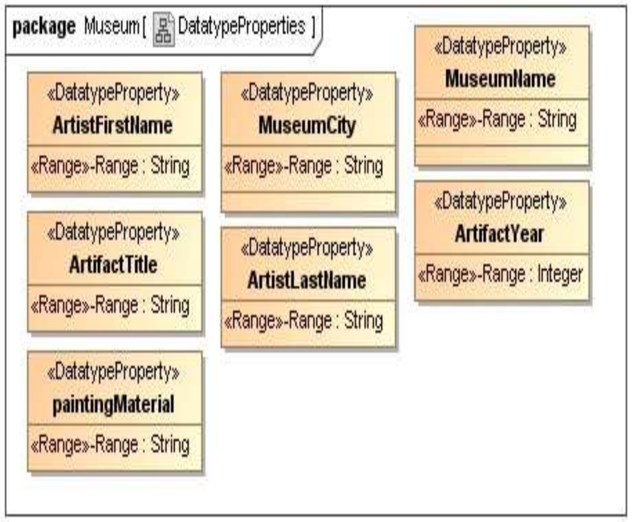

Figure6 : Ontology Model : DatatypeProperties in the museum example.

The code that is used to collect every datatypeproperties and transform them to Attributes , is shown below. In this code we used the lazy rule "thisModule.dtp2umlP(e)" which is responsible for constructing attributes via getting the name and type of them from the source model. Note that the property ownedAttribute in the UML metamodel refers to the attributes

```
ownedAttribute <- oc.getDatatypeProperty ->
collect(e|thisModule.dtp2umlP(e))
```

Attributes in a UML class are also deduced from the relationships with other classes such as, unionOf, InterssectionOf ComplementOf, disjointWith or EquivelentWith. The attributes generated from these dependencies in the Ontology model are accompanied with the previous generated ownedAttribute, as shown below:

```
ownedAttribute<-if oc.hasStereotype('Union') then
            oc.getSuppliers('UnionOf') -> collect(e |
            thisModule.suppU2Prp(e))
            else
            Sequence{}
```

Where the lazy rule named thisModule.suppU2Prp(e) is responsible for constructing attributes via setting name of the attribute as "Union Of" and the type with the dependency's stereotyped "unionOf" supplier.

On the other hand, objectProperties in ontology model relate individuals with each other (domain and range of these properties are individuals). In requirement model this is modeled as associations shown as in Figure 7 .

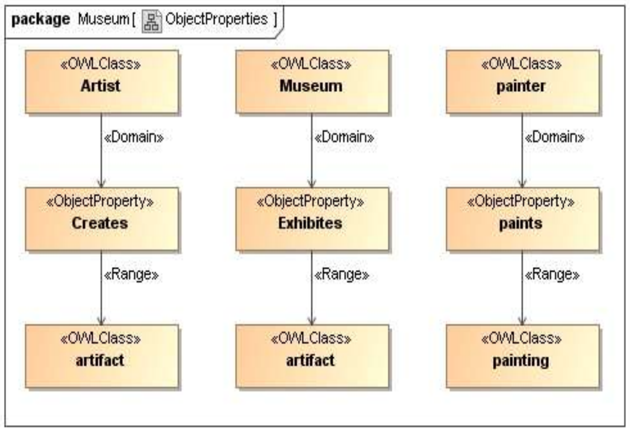

Figure7: Ontology Model : ObjectProperties in the museum example.

The code used to generate the UML class's associations from the Ontology Model's objectProperties is shown below

```
rule objProperty{
  from opc : UML!Class(
                  opc.hasStereotype('ObjectProperty')    )
   to assoc : UML!Association  (
 name  <- ' Association _ ' +
thisModule.AssocID.toString() + '  '+ opc.name ,
ownedEnd <- srcProp,
ownedEnd <- dmnProp)}
```

Note that srcProp and dmnProp are deduced as type of classes participate in the ObjectProperty's domain and range respectively.

Individuals in an ontology models are instances of owlclasses, where they can be mapped to objects in UML. Individuals that participate in a statement are either subject or object, and in this case, slots are used to link individuals with their datatypeproperties or objectProperties. Slots in an Ontology model exists in three places. The first is in individuals, by which individuals are the subject of these properties, and these slots are stereotyped "subjectSlots". The last element which could have slots is the objectProperties instances, by which it refer to its object , and these slots are named "objectSlots". The individuals of the museum example as shown in Figure 8, are transformed to objects as shown in Figure 9.classes are linked to their datatyeProperties instances, slots then are stereotyped "dataSlot". Slots also appear in navigable links, that connect an individuals with their objectProperties instances;

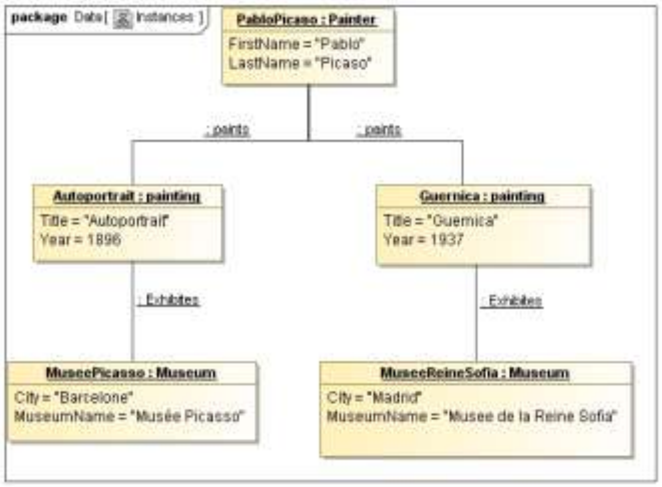

Figure 8: Ontology Model : ObjectProperties in the museum example.

Part of the code responsible for transforming individuals to UML objects is shown below in the rule named "individual":

```
rule individual
{
from individual : UML!InstanceSpecification(
individual.classifier >first().oclIsTypeOf(UML!Class)
and( individual.hasStereotype('OWLClass')or
individual.hasStereotype('Individual')or
individual.hasStereotype('Subject') or
individual.hasStereotype('Object'  )  )  )
to object : UML!InstanceSpecification(

name <- individual.name,
lassifier <- individual.classifier ,
```

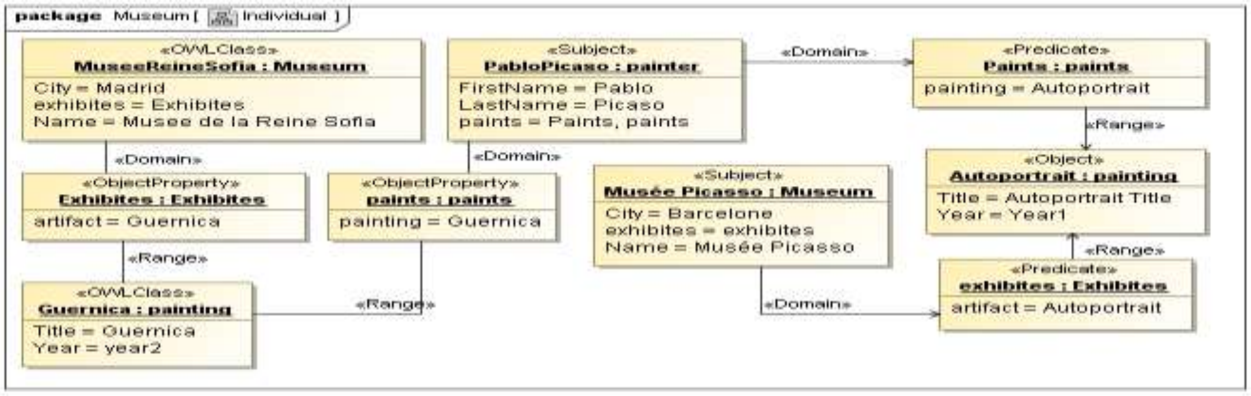
package Museum [Individual]
«OWLClass» MuseeReineSofia : Museum
City = Madrid
exhibites = Exhibites
Name = Musee de la Reine Sofia
«Subject» PabloPicaso : painter
FirstName = Pablo
LastName = Picaso
paints = Paints, paints
«Domain»
«Predicate» Paints : paints
painting = Autoportrait
«Range»
«Domain»
«Domain»
«ObjectProperty» Exhibites : Exhibites
artifact = Guernica
«ObjectProperty» paints : paints
painting = Guernica
«Subject» Musée Picasso : Museum
City = Barcelone
exhibites = exhibites
Name = Musée Picasso
«Object» Autoportrait : painting
Title = Autoportrait Title
Year = Year1
«Range»
«Range»
«Predicate» exhibites : Exhibites
artifact = Autoportrait
«OWLClass» Guernica : painting
Title = Guernica
Year = year2
«Range»
«Domain»

Figure 9: Ontology Model : Individuals in the museum example

And the rule that collect each object's slots is shown below:

```
rule slot {
            from s : UML!Slot
            to os : UML!Slot(
definingFeature <- s.getDF(),
value  <-  if s.value.oclIsTypeOf(UML!InstanceValue)
            then
 let a : UML!InstanceSpecification= s.value.Instace in
a.slot ->first().value
    else  s.value
```

Note that ATL is a transformation language that allows only reading the source model , and writing only the target model, so we make two steps transformation ATL file that run in an Ant build file with the command "superImpose" , so it is also possible to superimpose several transformation modules on top of each other [28].

Note that our transformation used UML metamodel as the metamodel for both source and target metamodels. The source model is accompanied with our Ontology Modeling Profile [33]. The transformation rules check the stereotype of the UML element before transforming to the target elements. More about using ATL and UML profiles could be found in the ATL site [29]. The examples are modeled using Magicdraw 16.6 enterprize edition.

6. CONCLUSION AND RELATED WORK

Many approaches have been proposed to use ontology in the development process of current web engineering methodologies. Xu. et al., in [30], presented a formal approach for extracting OWL DL ontologies from existing UML class diagrams. This approach establishes a precise conceptual correspondence between the two models and relies on a semantics-preserving UML-to-OWL translation algorithm. The proposed approach shows automatic ontology extraction from UML Class Diagrams. In [31], Reif et al. have introduced a technique to extend the existing Web engineering methodologies to develop semantically annotated Web pages. They defined a mapping from XML Schema to ontologies, called WEESA, that can be used to automatically generate RDF meta-data from XML content documents. They also showed how to integrate the WEESA mapping into an Apache Cocoon transformer to easily extend XML based Web applications to semantically annotated Web application. In [32], Gregory et al. presented an approach to the representation of requirements based on an ontology framework, which is the requirement ontology. In the requirement ontology, structures of a sentence are described as relations between sentence components. And meanings of key words are explained by WordNet. Thus, semantics of natural language requirements (NLRs) are captured for further processing.

In our approach, we proposed generating the requirement models from the prior modeled Ontology model. This approach provide an extension to the current web engineering, in order to be able to develop semantic web engineering. Defining transformation rules at metamodel level we achieve a model driven development approach. We present such transformations rules for an early phase in the development life cycle of Web system, which is the basis for an automated generation of analysis model from ontology model. The source modeling elements for our transformations are instances of any ontology model based on our profile specialized for modeling ontology concepts in accordance to the OMG's ODM. The targets of our transformations is the content analysis model that can be used for further web engineering process. The transformation rules are specified in the ATL (Atlas Transformation Language) language. For the success of applying MDD in web engineering techniques we aim to make automatic generation of the Web Ontology Language (OWL) from UML model based on our profile (OMP).This conversion transforms an ontology from its OMP into OWL description. Accordingly, this generated OWL model can be shared with ontological engineering tools (i.e. Protege).

7. REFERENCES

[1]. N. Moreno, J. R. Romero, and A. Vallecillo, “An overview of model-driven web engineering and the mda,” 2008, pp. 353-382.

[2]. N. Koch, “Classi_cation of model transformation techniques used in uml-based web engineering,” Institution of Engineering and Technology, vol. 1, no. 3, pp. 98-111, 2007.

[3]. Y. Deshpande, A. Murugesan, S.and Ginige, S. Hansen, D. Schwabe, M. Gaedke, and B. White, ”Web engineering,” Web Engineering, vol. 1, no. 1, pp. 003-017, 2002.

[4]. M. J. Escalona and N. Koch, “ Metamodelling the requirements of web systems.” in Proc. of 2nd International Conference on Web Information Systems and Technologies, Setubal, Portugal,, April 2006., pp. 310-317.

[5]. C. W. E. T. Fernando Silva Parreiras, TobiasWalter, “Model-driven software development with semantic web technologies,” Tutorial at the 6th European Conference on Modelling Foundations and Applications ECMFA 2010, Paris, France, June 15-18, 2010.

[6]. J. M. Stephen, N. C. Anthony, and F. Takao, “Model-driven development,” IEEE Software, pp. 14-18, Sep./Oct. 2003.

[7]. M. J. Escalona and N. Koch, “ Requirements engineering for web applications: A comparative study.” Web Engineering, vol. 2, no. 3, pp. 192-212, Feb. 2004.

[8]. R. Vidgen, “Requirement analysis and uml, use cases and class diagrams,” Computing and Control Engineering, pp. 12-17, 2003.

[9]. P. Shoval, A. Yampolsky, and M. Last, “Class diagrams and use cases,” in Proc. of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'06),J. Krogstie, T. Halpin, and H. E. Proper, Eds. Namur University Press, Namur, Belgium, EU, 2006, pp. 453-464.

[10]. K. Siau and L. Lee, "Are use case and class diagrams complementary in requirements analysis? an experimental study on use case and class diagrams in uml," Requirements Engineering, Springer London, pp. 229-237, 7 Oct. 2004.

[11]. J. Euzenat, "Research challenges and perspectives of the semantic web," Intelligent Systems,IEEE, vol. 17, no. 5, pp. 86-88, 2002.

[12]. "Ontology engineering" ,from wikipedia, the free encyclopedia.

[13]. Z. P. Je_ and H. Ian, "Metamodeling architecture of web ontology languages," in Proc. of the Semantic Web Working Symposium, July 2001, pp. 131-149.

[14]. D. Dragan, G. Dragan, D. Vladan, and D. Violeta, "A uml profile for owl ontologies," in Proc. of the Workshop on Model Driven Architecture: Foundations and Applications, Linkoping University, Sweden, 2004, pp. 138-152.

[15]. D. Djuric, G. D., D. V., and D. V., MDA-Based Ontological Engineering. World Scientific Publishing Co., Singapore, 2005, pp. 203-231.

[16]. D. Gasevic, D. Djuric, and D. V., "Bridging mda and owl ontologies," Journal of Web Engineering, vol. 4, no. 1, pp. 119-135, 2005.

[17]. K. S. Michael, W. Chris, and L. M. Deborah, "Owl web ontology language guide," http://www.w3.org/TR/owl-guide/, 10 Feb. 2004.

[18]. D. Li, K. Pranam, D. Zhongli, A. Sasikanth, and J. Anupam, "Using ontologies in the semantic web: A survey," UMBC, Tech. Rep., July 2005.

[19]. J. Hans and S. Stefan, "Applications of ontologies in software engineering," in International Workshop on Semantic Web Enabled Software Engineering (SWESE'06), November 2006.

[20]. A. S. C. Seria, C. B. Sabin, C. Liliana, and C. N. Ovidiu, "Survey on web ontology editing tools," http://thor.info.uaic.ro/~busaco/publications/articles/web-ontology-tool-survey.pdf, 2006.

[21]. X. Wang and C. Chan, "Ontology modeling using uml," in Proc. of the Seventh International Conference on Object Oriented Information Systems (OOIS), Calgary, Canada, Aug. 2001, pp. 59 -70.

[22]. C. Coral, R. Francisco, and P. Mario, Ontologies for Software Engineering and Software Technology. Francis: Springer-Verlag Berlin Heidelberg, 2006.

[23]. A. Kraus, "Model driven software engineering for web applications," Ph.D. dissertation, Ludwig-Maximilians-Universitat Munchen, 23rd / 04 2007.

[24]. K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in OOP- SLA03 Workshop on Generative Techniques in the Context of Model-Driven Architecture, 2003.

[25]. J. Bezivin, F. , Fabian Buttner, M. , Gogolla, F. Jouault, I. Kurtev, and A. Lindow, "Model transformations? transformation models!" in Model Driven Engineering Languages and Systems, 2006, pp. 440-453.

[26]. F. Jouault, F. Allilaire, J. Bzivin, I. Kurtev, and P. Valduriez, "Atl: a qvt-like transformation language," in OOPSLA '06: Companion to the 21st ACM SIGPLAN symposium on Object- oriented programming systems, languages, and applications. ACM, 2006, pp. 719-720.

[27]. F. Jouault, F. Allilaire, J. Bzivin, and I. Kurtev, "Atl: A model transformation tool," Science of Computer Programming, vol. 72, no. 1-2, pp. 31-39, June 2008.

[28]. "Atl superimposition," http://wiki.eclipse.org/ATL Superimposition.

[29]. A. T. List, http://www.eclipse.org/m2m/atl/atlTransformations/.

[30]. Z. Xu, Y. Ni, L. . Lin, and H. Gu, A Semantics-Preserving Approach for Extracting OWL Ontologies from UML Class Diagrams. Springer Berlin Heidelberg, 2009.

[31]. R. G., M. Jazayeri, and H. Gall, "Towards semantic web engineering: Weesa - mapping xml schema to ontologies," in In Workshop on Application Design, Development and Implementation Issues in the Semantic Web at the 13th International World Wide Web Conference, 2005, pp. 722-729.

[32]. H. C. Rong Li, Keqing He, "From natural language requirements to requirement ontologies," 2010 2nd International Conference on Future Computer and Communication, vol. 3, pp. 755-758, 2010.

[33]. W. Alakwaa and A. Salah, "Ontology modeling profile , an extension for the ontology uml profile," Aug. 2010.