UML to OWL Mapping Overview

An analysis of the translation process and supporting tools

David de Almeida Ferreira [1], Alberto Manuel Rodrigues da Silva [2].

1) INESC-ID/Instituto Superior Técnico, Lisbon, Portugal
david.ferreira@inesc-id.pt

2) INESC-ID/Instituto Superior Técnico, Lisbon, Portugal
alberto.silva@acm.org

Abstract

This paper presents a brief overview of the translation techniques between the OMG's UML modeling language and ontological engineering languages such as the W3C's standard for Semantic Web, the Web Ontology Language (OWL). The main objective of this paper is to analyze the feasibility and easiness of modeling an ontology by using common and mature UML CASE tools, thus reducing the gap between software engineering practitioners and AI techniques for ontology creation. The analysis and practical application of a proposed MOF-based Ontology Definition Metamodel (ODM) and the respective Ontology UML Profile (OUP) allows the straightforward comparison between MOF and OWL concepts.

Keywords: Modeling, Translation, Semantic Web, Ontology, UML, MOF, ODM, OUP, OWL.

1 Introduction

Nowadays we are observing a significant effort of introducing meaning (semantics) to data stored in the Internet. The W3C started to develop an ambitious framework called "Semantic Web" to define common formats for knowledge data interchange for overcoming the original Web limitation of only interchanging data through documents [Miller 2004], which constitute a coarse-grain level of knowledge bundles. To achieve this goal, the Semantic Web effort also engages the definition of a XML-based language for recording how data relates to real world objects allowing a human or a machine to store, access, and interchange that knowledge with a wide range of databases, which are connected through a lattice of concepts.

The main objective of the Semantic Web framework is to enable data to be shared and reused across application, enterprise, and community boundaries in a straightforward manner. Domain ontologies are the formal organization of domain-specific knowledge, thus constituting the most important part of Semantic Web applications. Besides the advantages in terms of knowledge interchange through heterogeneous environments, the developed ontology-based framework reduces the gap between the wider software engineering population and the disregarded, or even unknown, potential of Artificial Intelligence (AI) techniques, like reasoning.

The main purpose of this paper is to gather information about the current state of the art in this investigation area, and present some initiatives that allow one to create an ontology by using Unified Modeling Language (UML) CASE tools, and further refinement with ontology-specific tools. One of these initiatives, which provided a strong and consolidate theoretical background on the field, proposes an Ontology Definition Metamodel (ODM) [Djuric 2004]. This MOF-based

metamodel was used as the starting point to defining the Ontology UML Profile (OUP) [Djuric 2004], which allows the creation of an ontology by using the UML modeling language.

Initially, in Section 2, the reader will be presented with a brief overview of Semantic Web Architecture and the Web Ontology Language (OWL). Section 3 contains the theory that supports the conceptualization of an Ontology Definition Metamodel (ODM) by using the OMG's Model Driven Architecture (MDA). Section 4 provides some insight on the peculiarities of the translation process and ontology engineering tools, and Section 5 describes a practical case study. Finally, the last section presents the main conclusions of this research.

2 Semantic Web Architecture's Overview

The World Wide Web "next big step" is the broad adoption of the W3C's Semantic Web framework, enabling the interchange of computer-understandable data over the Internet [Miller 2004]. The adoption of a XML-based language provides the interoperability needed and, simultaneously, enables information enrichment with metadata tags, thus endorsing it with meaning through a precise semantics specification. Semantic Web architecture consists in a functional, non-fixed, three-level architecture. Each one of these levels incrementally introduces new primitives providing more expressive power, namely:

– **Metadata layer:** since most of RDF [Herman 2006a] usage is for describing data about data ("metadata"), in which assertions about other assertions are made, this layer justifies its existence by providing a set of basic constructs that fulfill this low-level functionality.

– **Schema layer:** it's required to declare the existence of new properties and constraining their usage. This type of meta-assertions enables one to perform elementary validations on a document. At the schema layer, the language allows simple assertions about the acceptable combinations of lower-level constructs. These basic constraints are easily expanded into more powerful logical layer expressions; however, the expressive power has to be restricted to guaranty it is still logically well-defined, hence enabling further semantics addition in the next level without compromising the usage of inference-engines at the logical layer.

– Logical layer: provides the mechanisms for writing logic into documents by defining logical rules. This allows document type's inheritance deduction, documents validation against a set of self-consistency rules, and query resolution by conversion from unknown terms into known ones; thus the potential usages of RDF are only limited by imagination.

Figure 1 presents the languages that support Semantic Web and the respective place of OWL [Herman 2006b] in the overall architecture. As depicted, XML is responsible for the syntactical level, whereas the semantic level is provided by RDF, RDF Schema, and especially by OWL.

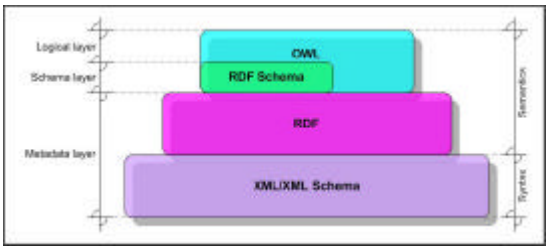

Figure 1 – Semantic Web Architecture (adapted from [Djuric 2004])

The OWL has three sub-languages which are a subset of one another. Each one of them has more expressive power than the inner sub-language, allowing one to model problems at different

complexity levels. This OWL's partitioning was introduced as a countermeasure for the trade-offs of using more expressive languages instead of simpler ones: (1) by using an unrestricted language as OWL Full (which provides capabilities for unconstrained representation of knowledge), we lose the guaranty of the OWL DL support for calculations and reasoning in finite time with the use of inference-engines; however (2) the usage of OWL DL may constitute a complexity overkill for modeling a simple ontology. Therefore, with this clear separation, one can choose the OWL subset that best fits for a specific problem at hands. A brief description of each one of these OWL's sub-languages and their main features is provided by [McGuinness 2004].

3 ODM's Adoption of MDA Standards

The initiative adopted as a reference for mapping UML to OWL was [Djuric 2004]. Its main goal is to provide a widely adopted, industry standard, graphical modeling language for easy ontology creation. This initiative was chosen since it presented a solid work on this area, strongly consolidated by a robust theoretical groundwork. To understand the followed approach one has to bear in mind some of the main concepts in which it is based on. To inherit the main advantages of OMG's previous work (in terms of standard-based definitions, interoperability by intra-level bridging, and also metamodeling and translation mechanisms) the authors specified an Ontology Definition Metamodel (ODM), founded on OMG's Model Driven Architecture (MDA). Only after this brief MDA's main concepts overview one can fully comprehend the proposed approach.

3.1 OMG's MDA Paradigm

Model Driven Architecture (MDA) is a framework for software development presented and maintained by the Object Management Group (OMG). The core of MDA paradigm is based on the importance of models in the software development process. Within MDA the software development process is driven by the activity of modeling a specific software system. It separates domain knowledge from platform knowledge. The more clearly this separation of concerns (between these two logically independent fields of knowledge) is made, the more fundamental MDA's advantages are emphasized, namely: (1) separation between domain and platform expertise, enabling reusability; (2) quick adaptability to domain and technology changes, ensured by efficient maintenance and system's documentation processes; (3) high productivity by generating working, high-quality applications; (4) easiness of integration with legacy or other third party systems by enhancing portability and interoperability; and (5) leveraging by reuse of existing IT technologies.

As presented in Figure 2, MDA's metamodeling architecture presents a four-layer organization and entails several OMG's additional standards. The mentioned MDA's layers are: (1) meta-metamodel (M3) layer, (2) metamodel (M2) layer, (3) model (M1) layer, and (4) instance (M0) layer. The previously referred complementary standards, which MDA requires, are: (1) Meta-Object Facility (MOF), (2) Unified Modeling Language (UML), (3) and XML Metadata Interchange (XMI). Once again, by looking at Figure 2, one can observe that on top of MDA's architecture is the OMG's Meta-Object Facility (MOF) meta-metamodel.

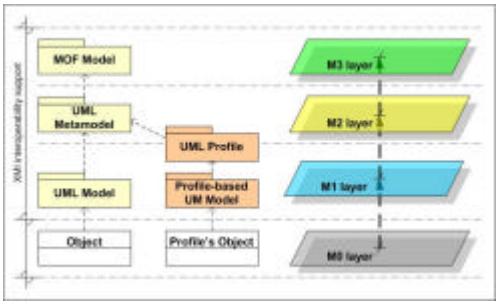

Figure 2 – OMG's MDA Metamodeling MOF-based Architecture

MOF defines an abstract language and framework for specifying, constructing and managing technology neutral metamodels, being the foundation for defining any modeling language, even allowing MOF's self-definition. MOF also defines a framework for implementing repositories that hold metadata described by metamodels. The main goal of having this four-level stratified structure with a common meta-metamodel is to support multiple models and metamodels, their extensibility, integration, and also generic model and metamodel management.

3.2 A MOF-based Ontology Definition Model (ODM)

The creation of an Ontology Definition Metamodel (ODM) [Djuric 2004] based on the MDA's principles and defined by the Meta-Object Facility (MOF) allows the definition of an Ontology UML Profile that enhances the ontological engineering creation process with commonly used UML notation and, simultaneously, allows the comparison between the MOF and OWL concepts in a straightforward manner. Another desired advantage, inherited from adopting MDA's as the base framework, is a bidirectional mapping between OWL and ODM, ODM and Ontology UML Profile (OUP), and even OUP and other UML profiles, through XSL Transformations.

The proposed ODM comprehends common ontology concepts, namely the ones present in OWL (the reference W3C standard), thus providing a starting point to this metamodel. To understand it, one has to clarify the main differences between metamodeling based on MDA, and the functional architecture used for ontology languages definition. While MDA has fixed and well-defined four-level architecture, the schema layer (RDFS) constitutes a non-standard and non-fixed-layer metamodeling architecture, making certain elements in model to have dual roles. This ambiguity has two major implications: (1) the impossibility of using modelers and, since it occurs in a low-level layer, (2) it propagates this problem to all the languages defined on top of it, namely OWL. In OWL DL, these problems are partially solved by introducing new modeling elements that are used for defining an ontology. However, since OWL Full allows unconstrained use of RDFS constructs it completely inherits RDFS's problems. Therefore, ODM for OWL Full cannot be modeled directly by using MOF. Having in mind the previous enunciated premises, ODM was primarily designed to support OWL DL, whereas OWL Full is only supported partially – concepts that do not break the MDA's fixed-layer architecture.

3.3 The Ontology UML Profile

In order to make use of graphical UML's modeling capabilities, an ODM must provide a corresponding UML Profile. This profile is required for graphical editing of ontologies using standard UML diagrams as well as benefiting of other advantages of using established UML CASE tools, such as Poseidon for UML, which was used in the practical case study in Section 5. Both UML models and ODM models are serialized in the XMI format so the two-way

transformation between them can be done by using XSL Transformations. Since OWL is also based on a layered XML-based platform, as illustrated in Figure 1, it is possible to define another pair of XSL Transformations through which we can attain a bidirectional mapping between UML and OWL, having ODM as intermediary representation. This enables the definition of an ontology by simply using a common UML modeler and a XSLT processor, thus producing the required OWL file that can be further refined by using a specific Ontology CASE tool as Protégé. Having a solid theoretical background that supports the previously described ODM, which provides the foundation to define this OUP, the potential of using additional well-defined bidirectional XSL Transformations of OUP into another technology-specific UML Profiles is unlimited, enabling the usage of ontology engineering techniques in the design of other domains and vice versa.

As depicted in Figure 2 one can observe that UML profiles are a workaround to support new languages without introducing new elements at the metamodel layer (M2). Therefore, UML profiles conceptually situate in the middle of M2 and M1 layers, since they are defined by M2 layer primitive constructs but used at M1 layer as M2 elements.

Figure 3 presents the OMG's UML superstructure specification [OMG 2005]. One can verify that formally UML and OWL are quite different. Some relevant observations [Hart 2004] to keep in mind when comparing them are: (1) UML model's content belongs mostly to the M1 layer and the M0 model just requires obeying to the M1 model's specification; (2) the linkage between a `Class` and an `Association` must be mediated by a `Property`; (3) the M1 model is built from structural elements; (4) both `Class` and `Association` are classifiers, having by inheritance a specific type; (5) a `Class` always has a `Property` which is the structural element that implements it; and (6) a `Property` may or may not be owned by one or more `Classes`. A `Property` owned by at least one `Class` is called navigable (memberEnd), whereas a `Property` not owned by any `Class` is called not navigable (ownedEnd).

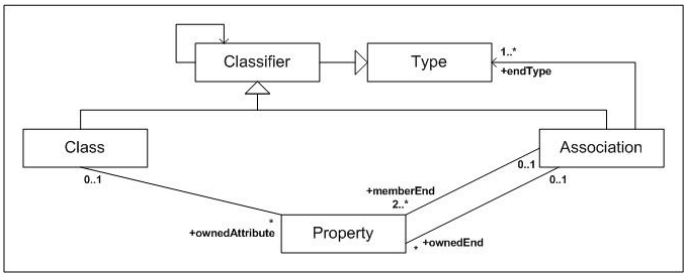

Figure 3 – OMG's UML Superstructure Specification (adapted from [OMG 2005])

This brief overview of the UML superstructure will be useful when analyzing the Ontology UML Profile (OUP) available at [Djuric 2004], which describes in detail the main concepts related to ODM and the Ontology UML Profile (OUP).

4 Ontology Engineering Tools

This section gives an overview of the tools used during the UML-to-OWL mapping process analyzed in the practical case study of transforming a UML domain model into an ontology.

4.1 XSLT Processor

The XSLT Processor initially used during the practical case study (to support simple XSL Transformation tests) was Altova XMLSpy [Altova 2006], which is a powerful tool to aid the

creation of advanced XML-based applications. It revealed to be at the same time flexible enough to allow one to work with large and complex XML-based documents by providing multiple views and options that best suit to one's working preferences. This tool significantly increases productivity by allowing one to quickly develop higher-quality, standards-conformant XML-based applications. Its end-to-end development process' coverage range includes support for: (1) project management; (2) XML document and DTD creation and editing; (3) XPath development; (4) schema-aware XQuery and XSLT 1.0/2.0 development and debugging; (5) graphical WSDL and Web Services development; (6) SOAP development and debugging; (7) database interaction; and (8) VS.NET and Eclipse integration with code generation options.

Besides the Altova XMLSpy, the Microsoft .NET Framework [Microsoft 2006a] was also used to develop a simple program to process the XSL Transformations provided, in a later phase of the undertaken case study. The .NET Framework Class Library offers a rich set of features to support XSL Transformation, namely the *XslCompiledTransform* [Microsoft 2006a] class which implements the XSLT processor in this framework, supporting the W3C XSLT 1.0 recommendations. Moreover, it also provides an extremely flexible XPath-Navigator Class [Microsoft 2006b], which enables one to easily navigate and manipulate XML documents.

4.2 UML Modeler

Poseidon for UML [Gentleware 2006] offers a rich Graphical User Interface (GUI) and, besides the portability offered by the Java Virtual Machine (JVM), it has the advantage of providing a free, non-commercial, single-user Community Edition, which contributed for its mainstream adoption. Despite not being open source, this freely available version supports all UML diagrams, reverse engineering for Java code, and code generation. Poseidon is a sophisticated UML drawing program with a rich set of useful usability features and natively supports UML 2.0. Its main features are: (1) comprehensive suite of powerful features; (2) inherently simple functionality; and (3) Eclipse integration (for professional edition only).

4.3 Ontology IDE

Protégé [SMIOSC 2006] is a free, open-source platform supported by a growing user community, encompassing wide-ranging expertise areas. This application offers a coherent set of tools to design domain models and to build knowledge-based applications by using ontologies. Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be extensively customized and provides a good support for creating knowledge models and entering data for populating the knowledge-base. Through its plug-in architecture and a well-defined Java-based Application Programming Interface (API), Protégé can be easily extended for building more complex (or integrate with external third-party) knowledge-based tools and applications.

The Protégé tool supports two different approaches for modeling an ontology:

– **Protégé-Frames Editor:** allows one to easily manage frame-based ontologies, in accordance with the Open Knowledge Base Connectivity (OKBC) protocol (Figure 4).

– **Protégé-OWL Editor:** allows one to create ontologies for the Semantic Web, namely by using the W3C's Web Ontology Language (OWL), which allows deriving logical consequences through its well-defined formal semantics specification, i.e. to deduce facts not literally present in the ontology, but implicitly entailed by the constructs' semantics.

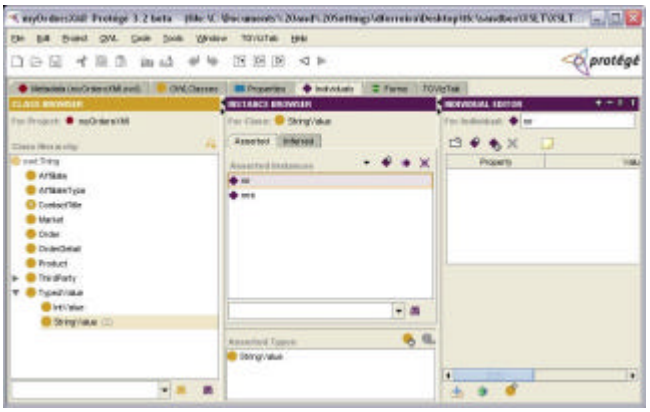

Figure 4 –Protégé-Frames Editor Screenshot

4.4 OWL Reasoners

FaCT++ [Tsarkov 2006] is the new generation of the well-known FaCT OWL-DL reasoner, using the established FaCT algorithms, but with a different internal architecture and presenting enhanced features. This application is implemented using C++ in order to create a more efficient software tool, and to maximize portability. It supports the standardized XML interface to Description Logics systems developed by the DL Implementation Group (DIG), which allows it to run in server mode to support reasoning features required by other third-party tools.

Jena [HPLabs 2006a] is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine [HPLabs 2006b].

5 A Practical Case Study

This section's main goal is to demonstrate the practical appliance of the concepts and techniques introduced in the previous sections through a concrete case study. This example, besides serving as a proof-of-concept, provided a good theoretical insight, consolidated by experience. The feedback gathered was vital to realize the pending idiosyncrasies in this expertise field of UML-to-OWL transformations through an Ontology Definition Metamodel (ODM) and Ontology UML Profile (OUP).

The case study consists in the analysis of the ontological engineering process involved in the creation of an ontology for a typical information system. The presented system, MyOrders, is an academic modeling exercise. Its domain model class diagram is illustrated in Figure 5.

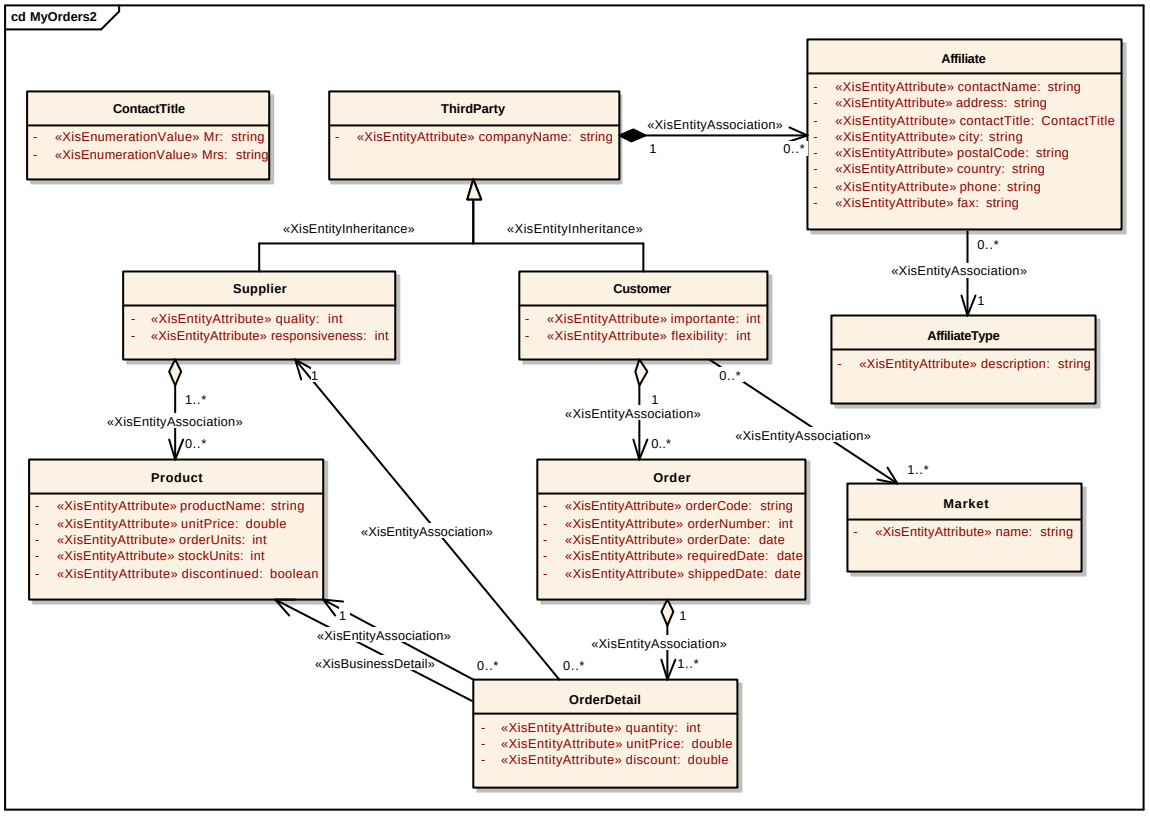

Figure 5 – MyOrders System's Domain Model Class Diagram

The main activities involved in the ontological engineering process, presented in Figure 6, were:

– The first step consisted in analyzing existing XLST processors, namely XML Spy and .NET Framework Class Library features. At this stage it was noticed that there were some problems when generating OWL files with XML Spy having as input the UMLtoOWL.xslt file available at [Gasevic 2006]. Simply, it generated header errors when importing it with Protégé 3.2 beta. Therefore, it was implemented a simple application that generates parameterized OWL files by consuming as input the mentioned XSLT file with the mapping rules and a XMI file generated by Poseidon enhanced with OUP.

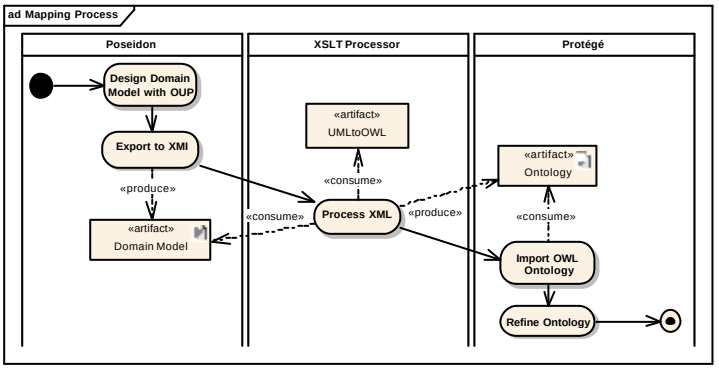

Figure 6 – UML-to-OWL Mapping Process

– The next step consisted in understanding existing OUP examples, to explore its true potentialities, since there are some significant paradigm differences when comparing MDA-based and Ontology-based approaches, as discussed in Section 3.

– Bearing in mind all the required concepts, the concrete modeling task started. This activity revealed most of the idiosyncrasies pointed out in the theoretical sections, namely the diagram cluttering (consult Figures 7 and 8) by explicitly defining UML attributes with the `ObjectProperty` and `DataTypeProperty` stereotypes.

– The process to achieve an .owl file consists simply in developing the model with OUP stereotypes in Poseidon, export it to XMI, run the application, and import the produced .owl file with Protégé. The main objective of this practical case study was achieved. However, there were serious difficulties to understand some unexpected errors related to the XSL Transformations provided (available at [Gasevic 2006]) which do not fully support OUP stereotypes, namely it offered a limited set of `DataType` properties that had to be enhanced to attain the case study goal, and some UML Associations' stereotypes that where not possible to transpose to Protégé, thus the produced ontology had to be further refined with Protégé to obtain the expected result.

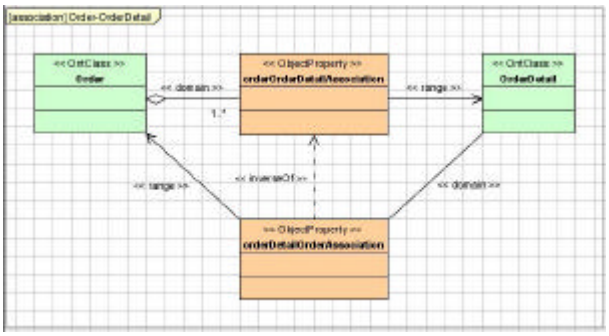

Figure 7 – Technique for Mapping an UML Association with OUP

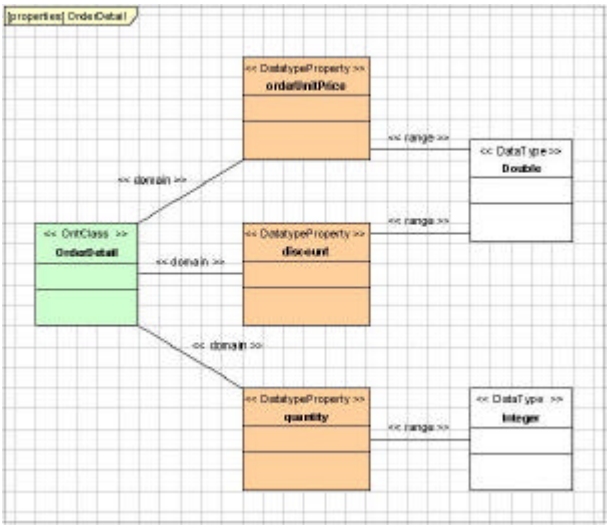

Figure 8 – Technique for Mapping an UML Attributes with OUP

6 Conclusions

An ontology describes the most important concepts and relationships in a particular domain, providing a vocabulary for it as well as a computer-based meaning's specification of terms used in the vocabulary. Ontologies range from taxonomies and classifications, database schema, to fully axiomatized theories. Recently, ontologies have been adopted in many business and scientific communities as a way of sharing, reuse, and process domain knowledge. Ontologies

are now central to many applications, namely scientific knowledge portals, information management and integration systems, electronic commerce, and semantic web services.

To be widely adopted by the software development community and to succeed in real-world applications, knowledge engineering and ontology modeling must catch up with mainstream software trends, namely OMG's standards. The introduction of an Ontology Modeling Architecture and its related techniques will provide a good support and ease the integration between existing or forthcoming software tools and applications, adding value to both sides.

The Ontology Definition Metamodel (ODM) and Ontology UML Profile (OUP) analyzed in this paper are allegedly in accordance with the OMG's RFP initiative for ontology modeling, constituting a strong bet in this scientific research area, and can be considered as a part of the effort to specify a standard ontology metamodel. The possibilities of defining other AI metamodels in MOF should and probably will be explored in a near future. This means that MDA and MOF will most likely constitute the integration/bridge point between common and AI-related metamodels.

Despite of the enormous value of this contribution to both software engineering and ontological engineering areas, this proposal has several issues to solve, namely the adoption of graphical icons or forms to represent OUP stereotypes, instead the usage of color schemes to enhance diagram's readability, and to further develop the XSL Transformation rules to fully support OUP mapping to OWL and, eventually, bidirectional transformations.

Finally, the reader was presented with a practical proof-of-concept application of the presented theory and techniques for modeling a typical Information System by using XSL Transformations and CASE tools (Poseidon and Protégé). This case study allowed pointing out some idiosyncrasies of the theoretical techniques, emphasizing the need for future work.

7 References

Altova, "XML Editor for Modeling, Editing, Transforming, and Debugging XML Technologies", (2006). http://www.altova.com/products/xmlspy/xml_editor.html, [accessed on 04/06/2006].

Djuric, D., "MDA-based Ontology Infrastructure", *International Journal on Computer Science and Information Systems*, 1, 1, (2004), 91–116.

Gasevic, D., "UMLtoOWL: Converter from UML to OWL", (2006), http://afrodita.rcub.bg.ac.yu/~gasevic/projects/UMLtoOWL, [accessed on 02/06/2006].

Gentleware, "Poseidon for UML", (2006), http://gentleware.com/, [accessed on 06/06/2006].

Hart, L., et al., "OWL Full and UML 2.0 Compared", (2004).

Herman, I., Swick, R., Brickley, D., "Resource Description Framework (RDF)", (2006), http://www.w3.org/RDF, [accessed on 04/06/2006].

Herman, I., Hendler, J., "Web Ontology Language (OWL), (2006), http://www.w3.org/2004/OWL, [accessed on 04/06/2006].

HP Labs Semantic Web Programme and Open Source Community, "Jena: A Semantic Web Framework for Java", (2006), http://jena.sourceforge.net, [accessed on 04/06/2006].

HP Labs Semantic Web Programme and Open Source Community, "Jena 2 Inference Support", (2006), http://jena.sourceforge.net/inference, [accessed on 04/06/2006].

McGuinness, D., Harmelen, F., "OWL Web Ontology Language Overview", (2004). http://www.w3.org/TR/owl-features, [accessed on 06/06/2006].

Microsoft, "XSLT Transformations: System.Xml.Xsl.XslCompiledTransform Class", (2006), http://msdn2.microsoft.com/en-us/library/14689742.aspx, [accessed on 03/06/2006].

Microsoft, ".NET Framework Class Library: System.Xml.XPath.XPathNavigator Class", (2006), http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemxmlxpathxpathnavigatorclasstopic.asp, [accessed on 03/06/2006].

Miller, E., "The Semantic Web", (2004), http://www.w3.org/2004/Talks/0120-semweb-umich, [accessed on 28/10/2006].

Object Management Group (OMG), "Unified Modeling Language: Superstructure", (2005). http://www.omg.org/docs/formal/05-07-04.pdf, [accessed on 06/06/2006], version 2.0.

Stanford Medical Informatics and Open Source Community, "Protégé, an Ontology Editor and Knowledge Acquisition System", (2006), http://protege.stanford.edu, [accessed on 06/06/2006].

Tsarkov, D., "FaCT OWL-DL external reasoner", (2006), http://owl.man.ac.uk/factplusplus, [accessed on 06/06/2006].