

Approaching OWL and MDA Through Technological Spaces

Dragan Gašević¹, Dragan Djuric¹, Vladan Devedžic¹, Violeta Damjanovic²

¹FON – School of Business Administration, University of Belgrade, POB 52, Jove Ilica 154, 11000 Belgrade, Serbia and Montenegro

²Postal Savings Bank, 27.marta 71, Belgrade, Serbia and Montenegro
gasevic@yahoo.com, dragandj@gmail.com, devedzic@galeb.etf.bg.ac.yu, vdamjanovic@gmail.com

Abstract. Web Ontology Language (OWL) and Model-Driven Architectures (MDA) are two technologies being developed in parallel, but by different communities. They have common points and issues and can be brought closer together. Many authors have so far stressed this problem and have proposed several solutions. The result of these efforts is the recent OMG's initiative for defining an ontology development platform. However, the problem of transformation between an ontology and MDA-based languages has been solved using rather partial and ad hoc solutions, most often by XSLT. In this paper we analyze OWL and MDA-compliant languages as separate technological spaces. In order to achieve a synergy between these technological spaces we define ontology languages in terms of MDA standards, recognize relations between OWL and MDA-based ontology languages, and propose mapping techniques. In order to illustrate the approach, we use an MDA-defined ontology architecture that includes ontology metamodel and ontology UML Profile. Based on this approach, we have implemented a transformation of the ontology UML Profile into OWL representation.

1. Introduction

The integration of the ongoing software engineering efforts with the concept of the Semantic Web is not a new idea [1]. The main question is how to develop the Semantic Web ontologies using well-accepted software engineering languages and techniques in order to provide the wider practitioner population to develop and use ontologies in real-world applications. Currently, there is a special interest group within Object Modeling Group (OMG) that pursuing to converge many different proposals regarding this problem [2]. The result of this effort should be a standard language (i.e. metamodel) based on the Model Driven Architecture (MDA) standards [3] and Web Ontology Language (OWL).

Technological spaces have recently been recognized in order to figure out how to work more efficiently by using the best possibilities of different technologies [4]. In our approach we identify similarities between two technological spaces: MDA-compliant languages and OWL, regarding their epistemological organization and layered architecture. For example, MDA has the four-layer metamodeling architecture whereas the ontology languages have the tree-layer architecture according to [5]. In order to develop valid transformations we should find equivalences among them. Also, the XML technological space is important for our analysis since both MDA and OWL use XML formats for sharing metadata. As result we give recommendations how to develop transformations between the MDA languages and OWL as well as which technologies can be used for implementation.

The next section formally defines MDA, metamodeling, UML Profiles, and technological spaces. Section 3 shortly depicts an example of an MDA-based ontology development architecture, which we defined according to the OMG's proposal. Using this architecture we give conceptual solution for mapping between MDA-compliant ontology languages and OWL in section 4. Section 5 contains an XSLT-based implementation example for transforming an ontology UML Profile into OWL, as well as our experiences in using this transformation. Section 6 discusses related work, whereas section 7 gives final conclusions. This work is a part of the effort of the GOOD OLD AI research group (<http://goodoldai.org.yu/>).

2. Formal framework

In this section we describe the MDA-supported standards, give important definitions regarding these standards, and define technological spaces.

2.1. MDA basics

Our work is based on the Model Driven Architecture (MDA) – an Object Modeling Group (OMG) ongoing software engineering effort. The central part of MDA is the four-layer architecture that has a number of standards for each of its layers (see Figure 1). Most of MDA standards are developed as metamodels using metamodeling. The top-most layer (M3) is called meta-metamodel and the OMG's standard defined at this layer is Meta-Object Facility (MOF). In term of MDA a metamodel makes statements about what can be expressed in the valid models of a certain modeling language. In fact, a metamodel is a model of a modeling language [6]. Examples of the MDA's metamodels are UML and Common Warehouse Metamodel (CWM). The MDA's metamodel layer is usually marked as M2. At this

layer we can define a new metamodel (e.g. modeling language) that would cover some specific application domains (e.g. ontology development). The next layer is the model layer (M1) – a layer where we develop real-world models (or domain models). In terms of the UML models that means creating classes, their relations, states, etc. There is an XML-based standard for sharing metadata that can be used for all of the MDA's layers. This standard is called XML Metadata Interchange (XMI). Of course, we should explain the bottom-most layer – the instance layer (M0). There are two different approaches about this question, and we note both of them:

1. The instance layer contains instances of the concepts defined at model (M1) layer (e.g. objects in programming languages).
2. The instance layer contains things from our reality – concrete (e.g. Lassie is a instance of the Dog class, etc.) and abstract (e.g. UML's classes – Dog, etc.) [7].

In this paper we advocate the second approach, but we should give a more details about its impact on UML. In UML both classes and objects are at the same layer (model layer) in the MDA four-layer architecture. Actually, MDA's layers are called linguistic layers. On the other side, concepts from the same linguistic layer can be at different ontological layers. Hence, UML classes and objects are at different ontological layers, but at the same linguistic layer.

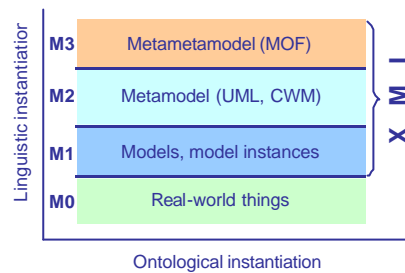


Fig. 1. The four-layer Model Driven Architecture and its orthogonal instance-of relations: linguistics and ontological

2.2. Specific MDA metamodels and UML Profiles

One possible solution for using MDA capacities in some specific domains is to develop a metamodel, which would be able to model relevant domain concepts. Having defined a domain specific metamodel we should develop suitable tools for using that metamodel. However, it is rather expensive and time consuming so we try to use well-developed tools. Practically, present software tools do not implement many of the MDA basic concepts. The problem of tools can be overcome using UML Profiles – as a way for adapting UML for specific purposes. UML Profiles extend the UML metamodel with application-specific primitives (through stereotypes, tagged values, and constraints), and hence these primitives can be used as the regular UML concepts. Having understood UML Profiles in this way one can count UML as a family of languages [8].

A very important question is about the place of UML Profiles in the MDA's four-layer architecture. The UML specification states that UML Profiles are defined at the metamodel layer (M2), and thus they are meta-concepts. Here we use a definition of UML Profiles in a strict metamodeling framework [9] where UML Profiles are placed at both the metamodel layer (M2) and the model layer (M1).

2.3. Technological spaces

Nowadays, it is quite often that using only one technology in solving different engineering problems is not enough. For example, software engineers can benefit from ontological engineering, or database developers can find useful improvement in using the XML technology. Problems of bridging different technologies are discussed in [4] where the term of technological spaces is introduced. A technological space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. Although some technological spaces are difficult to define, they can be easily recognized (e.g. XML or MDA technological spaces). In order to get synergy of different technological spaces we should create bridges between them, and some of these bridges are bi-directional. The bridges can be created in a number of ways (e.g. in the XML technological space by using XSLT, in ontological engineering through transformations that can be mapped into XSLT, etc.). The Semantic Web is a field integrating XML and ontological engineering technological spaces.

Currently, there is an OMG's initiative (i.e. Request for Proposal – RFP) titled MOF 2.0 Query/View/Transformation (QVT) [10]. This is a platform neutral part of the MDA aiming to define a language for querying and transforming models as well as viewing metamodels. Atlas Transformation Language (ATL) is an example of submission for the OMG's QVT RFP. ATL can be used to bridge different technological spaces, and the potentials of this language are shown in [11] where it is used to transform XSLT documents to XQuery. Although this transformation can be done inside the XML technological space through XSLT, because an XSLT document is a valid XML document, it is performed in both MDA and XML technological spaces.

3. Ontology development and MDA

The problem of using UML for ontology development has been addressed in [12] for the first time. In fact, this was a pioneering work in integrating MDA and ontologies. Until now there were another few attempts to use MDA standards for the benefit ontological engineering. Currently, there is an initiative (i.e. RFP) within the OMG aiming to define a suitable language for modeling Semantic Web ontology languages in the context of MDA [3]. According to this RFP we have developed our proposal of such architecture [13]. In our approach for modeling ontology in the scope of MDA, we defined several specifications (which is inherited from the OMG's RFP [3]):

- Ontology Definition Metamodel (ODM)
- Ontology UML Profile (OUP) – a UML Profile that supports UML notation for ontology definition
- Two-way mappings between OWL and ODM, ODM and OUP and from OUP to other UML profiles.

In the next two subsections we describe our definitions for ODM and OUP. The main attention is given to OUP since our implementation example is based on this UML Profile.

3.1. Ontology metamodeling architecture

ODM was designed to enclose common ontology concepts. We used OWL as a good starting point for constructing ODM, since OWL is an actual W3C's recommendation [14]. We defined ODM using MOF. ODM gives us a metamodel-based semantic foundation [6] for ontology languages, so we can use the MDA's capabilities for ontology development. Details about ODM can be seen in [13].

3.2. Ontology UML Profile: source language

Class is one of the most fundamental concepts in ODM and OUP. In ODM, Ontology Class concept is represented as an instance of the (MOF) Class, and has several concrete species: Class, Enumeration, Union, Intersection, Complement, Restriction, and AllDifferent. In Figure 2 we show a part of the well-known Wine ontology. WineDescriptor is equivalent to the union of WineTaste and WineColor classes, whereas WineColor is an enumeration of WineColor instances: White, Rose, and Red. We should note that we have two anonymous classes (Union and Enumeration).

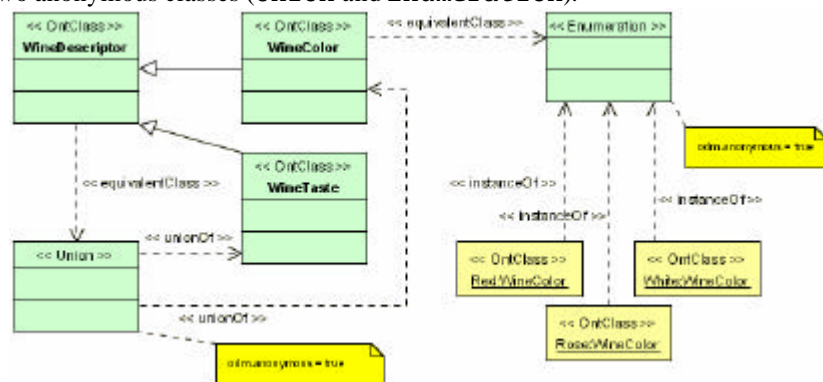


Fig. 2. The Ontology UML profile class-oriented stereotypes (an excerpt of the Wine ontology)

In UML, an instance of a Class is an Object. ODM Individual and UML Object have some differences, but they are similar enough, so in OUP, Individual is modeled as a UML Object (shown in Figure 2). Here we had difficulties deciding on what stereotype to attach to UML objects to make them represent ODM individuals. It would be natural to have a stereotype with the name «Individual», but the UML's specification explicitly prompts that the stereotype for an object must match the stereotype for its class. Accordingly, in OUP we have attached the «OntClass» stereotype to the OUP's class instances.

Since Property is a stand-alone concept it can be modeled using a stand-alone concept in UML. That concept can be the UML Class' stereotype «Property». However, Property must be able to represent relationships between Resources (Classes, Datatypes, etc. in the case of UML), which the UML Class itself is not able to do. ODM defines two types (subclasses) of Property – ObjectProperty and DatatypeProperty, and thus OUP has corresponding stereotypes. An example of a Class Diagram that depicts ontology properties modeled in UML is shown in Figure 3. In OUP we use the «Restriction» stereotype to refine property's restrictions. As a result we have an association (e.g. stereotype «someValuesFrom») between a class and an unnamed «Restriction», and two stereotyped dependencies from «Restriction» – «onProperty», and for example, «someValuesFrom». However, adding this «Restriction» construct in OUP is not the same as adding a class into property domain. Actually, it is mapped as a super class for the given class (e.g. see the locatedIn property in Figure 3 that has the «OntClass» Region as the *someValuesFrom* restriction).

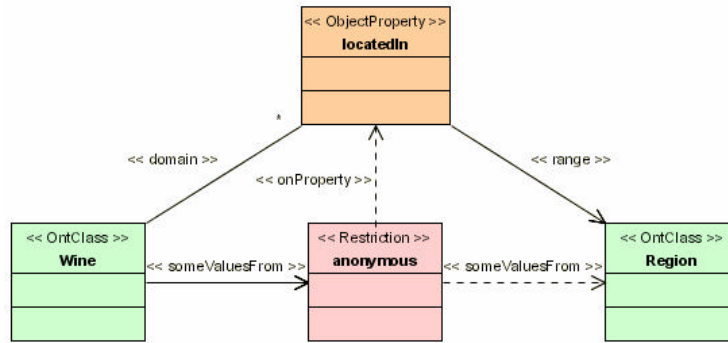


Fig. 3. The OUP class property and restriction on the example of the Wine ontology

ODM Statement is a concept that represents concrete links between ODM instances – Individuals and DataValues. Since in UML a Class' instance is an Object, in OUP, Statement is modeled with Object's stereotype «ObjectProperty» or «DatatypeProperty». UML Links are used to represent the subject and the object of a Statement. To indicate that a Link is the subject of a Statement, the LinkEnd's stereotype «subject» is used, while the object of the Statement is indicated with the LinkEnd's stereotype «object». LinkEnd's stereotypes are used because Link cannot have a stereotype in UML. These Links are actually instances of properties «domain» and «range». In brief, in OUP, Statement is represented as an Object with two Links – the subject Link and the object Link, which is shown in Figure 4. Here we have a statement that says the Region's instance MendocinoRegion is locatedIn SonomaRegion, and its adjacentRegion is CaliforniaRegion. Unlike other MDA-based solutions for ontology development, ODM and OUP support modeling of body of knowledge (i.e. class instances).

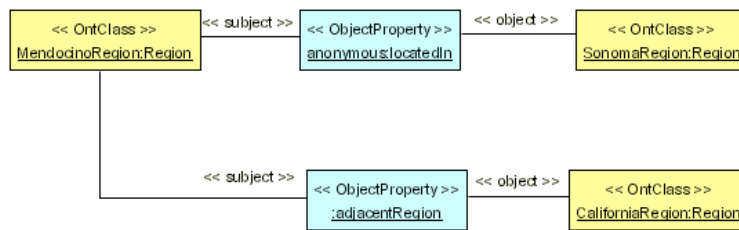


Fig. 4. OUP fully supports ontology body of knowledge (i.e. instances) through the OUP statements: the Wine ontology example

4. Conceptual solution

The presented metamodels (ODM and OUP) are MOF-compliant languages defined in the context of the MDA's metamodeling architecture. However, they are not enough by themselves, but they need interaction with real-world ontologies, such as OWL ontologies. It is obvious that we should develop transformations, which will provide conversions between the MDA's ontology languages and OWL. Our main idea is to explain all these transformations in the context of technological spaces.

4.1. Relations between technological spaces

Figure 5 shows all technological spaces we recognized as important for MDA standards and ontological engineering to be used cooperatively. In the MDA technological space we defined ODM and OUP. It is important to note that ODM is defined at the M2 layer, while OUP resides at both M1 and M2 layers according to [9]. Concrete real world models are at the M1 layer and they consist of classes and their instances. That means, we have two ontological layers at the M1 layer [7]. Following this idea we can say that the UML classes are at the O1 ontological layer whereas the UML objects are at the O0 ontological layer. For all MDA's layers one can use XMI, an XML-compatible format for sharing metadata.

The OWL technological space includes the W3C's recommendation for the Web Ontology Language. In this technological space we identify different abstraction layers in order to find relations with the MDA technological space. Two bottom-most layers are marked with O1 and O0. At the O1 layer we build ontologies, i.e. we create classes, properties, relations, and restrictions. On the other hand, ontological instances are at the O0 layer in the OWL technological space. In this paper the authors described an ontology language (i.e. Ontology Inference Layer – OIL) with another ontology language (i.e. Resource Definition Framework (Schema) RDF(S)). In fact, they create a meta-ontology. Finally, we can say there is a meta-ontology that defines OWL and this meta-ontology is at the O2

layer. Note that the explained three-layer organization of ontologies is already known in AI as Brachman's distinction of knowledge representation systems.

We can give some important statements in order to provide transformations between these technological spaces:

1. The O2 layer (meta-ontology) has the similar role with the metamodel (M2) layer (e.g. ODM and OUP) – they both specify an ontology language
2. Both O1 and O0 layers have the similar role with the MDA's model (M1) layer. In fact, this conclusion comes from the Atkinson and Kühne's ontological and linguistic layers [7] where one linguistic layer (in this case M1) can contain many ontological layers (in this case O1 and O0).

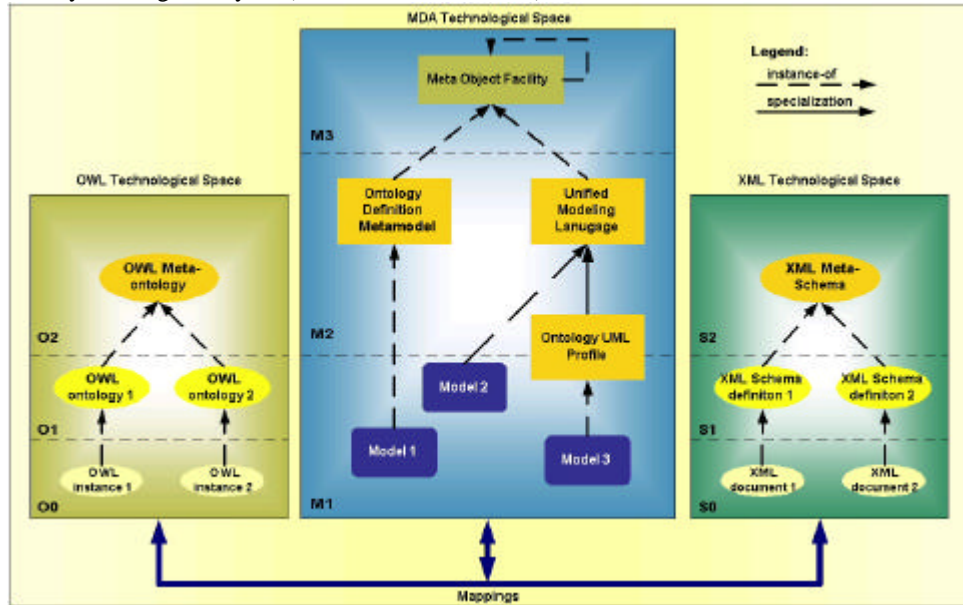


Fig. 5. An overview of technological spaces, which are important for the collaborative use of the MDA-compliant ontology languages and OWL, and their mutual relations: MDA, OWL, and XML technological spaces.

Since both these technological spaces use XML for sharing their metadata we can include a new technological space in this discussion. Of course, this is the XML technological space, which also has its own layered organization. This organization is very similar with the OWL's one, but it is defined in terms of syntax (not semantics). We observe the XML technological space in terms of the W3C's XML Schema recommendation. We shortly note these layers: the S2 layer is a schema for schemas (i.e. meta-schema) – this schema defines validity of XML Schema definition documents; domain specific XML vocabularies (i.e. schemas) are defined at the S1 layer; and concrete XML documents are at the S0 layer. Epistemologically, one can say that these three layers are equivalent with the OWL's layers (i.e. $S2 \Leftrightarrow O2$, $S1 \Leftrightarrow O1$, $S0 \Leftrightarrow O0$). Accordingly, there exists the same relation between MDA's layers and XML's layers (i.e. M2 with S2, M1 with both S1 and S0). Finally, we can define OWL and MDA's languages in terms of the XML Schema. That mean, we can define schemas that specify the OWL's XML syntax and an XML syntax of a metamodel (e.g. ODM and OUP).

4.2. Transformations between technological spaces

It is obvious from the previous descriptions that we cannot provide direct mappings between the MDA technological space and the OWL technological space. In fact, this transformation can only be defined through the XML technological space. Pragmatically, it is important that we should define a pair of transformation in order to enable two-way mapping (one transformation for either direction) between all OWL ontologies and all ontologies represented in an MDA-based ontology language. That means, we develop these transformations on the basis of the "meta-definitions" of OWL (i.e. its meta-ontology) and the MDA-compliant language (i.e. an metamodel). Note that this transformation principle is in accordance with the Bézivin's principle of metamodel-based model transformation [15]. Figure 6 shows transformation (OUP/OWL transformation) of an OUP model (i.e. an OUP document in the XMI format) to its equivalent OWL ontology (i.e. an OWL document in XML format). In other words, the transformation maps the MDA's M1 layer into its corresponding OWL's layers (O1 and O0). The most suitable implementation for this transformation is XSLT since we convert an XML document into another XML document. The opposite transformation (from OWL to OUP) can also be implemented in XSLT. In this case we suggest using RDF Query Language (RDQL) to transform OWL documents although this language is intended for querying RDF and OWL documents. However, the recent research in the Semantic Web Community shows that RDQL can be used as an ontology transformation language as well [16]. In fact, the Sesame query language (SeRQL) is used as a transformation language, but it can be considered as an RDQL dialect. Note that this transformation can be implemented in Java, but Java should be empowered with an OWL parser (e.g. Jena [17]).

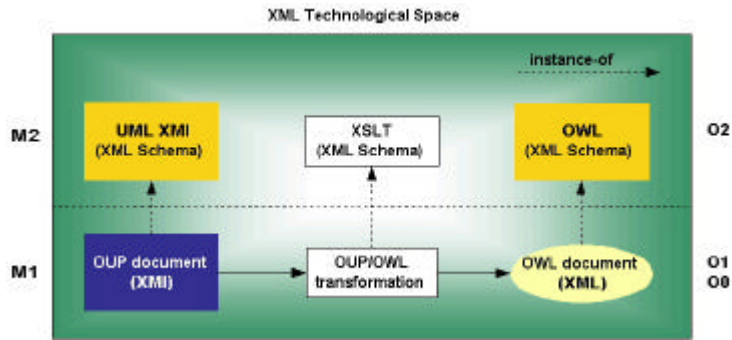


Fig. 6. An example of a transformation in the XML technological space: the transformation of OUP into OWL

Figure 7 gives an example of transformations in the MDA technological space. This figure is organized according to the transformation schema given in [18] for transforming XML Schemas to application models. This example shows that there are no problems regarding different metamodeling layers, since both metamodels are defined at the same layer (i.e. the M2 layer). Actually, the OUP's metamodel can be resided at both M1 and M2 layers. For the sake of symmetry we placed the OUP metamodel only at the M2 layer in Figure 7. As a matter of fact, this can be true if we would use a standard UML Profile for ontology development without user's extensions (see [9] for details). Note that this transformation can also be implemented through the XML technological space in terms of XML schemas and XML documents.

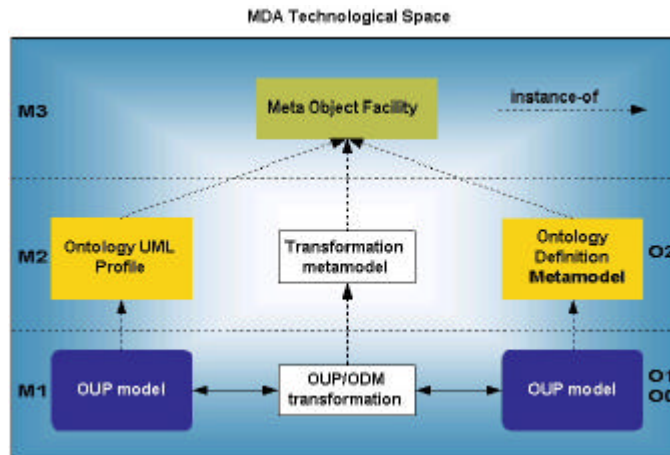


Fig. 7. Language transformation in the MDA technological space: transformations between OUP and ODM

5. An implementation example: An XSLT approach

In the previous section we have explained the conceptual solutions for transforming MDA-based languages (i.e. ODM and OUP) and OWL. In this section we show an implementation example that transforms an OUP-based ontology to its equivalent OWL ontology [19].

5.1. Implementation details

The main idea of having a UML profile for ontology development is to use regular UML tools (e.g. Poseidon for UML) that can export an XMI document as the input into an XSLT processor. An OWL document is produced as the output, and this format can be imported into a tool specialized for ontology development (e.g. Protégé), where it can be further refined. The XSLT, which we have implemented for mapping from the OUP XML format (i.e. UML XMI) to the OWL description, contains a set of rules (i.e. templates) that match XMI constructs and transform them into equivalent OWL primitives. While developing these rules we had to face some serious obstacles resulting from evident differences between source and target format. We note some of them:

- The structure of an XMI document is fairly awkward since it contains full description of an UML model.
- The OUP, in some cases, uses more than one UML construct to model one OWL element. For example, to model the *someValuesFrom* restriction using OUP (see Figure 3), we need three UML classes and three relations.
- UML tools can only draw UML models, but they do not have an ability to check the completeness of an OUP ontology. Thus, the XSLT is incurred to check XMI documents.

- The XSLT must make difference between classes that are defined in other classes (nested classes that can not be referenced from other classes using their ID), and classes that can be referenced using their ID. Accordingly, we included the *odm.anonymous* tagged value into OUP. This tagged value helps us detect these two cases.

Taking into account previously presented facts, one can deduce that developed XSLT is too large to be included in this paper. In order to depict an output OWL document that we obtain as the XSLT's result, we give Figure 8. Figure 8a gives the OWL description classes we have defined in Figure 2. It is interesting to note how OUP's classes that have tagged value *odm.anonymous* are mapped into OWL (e.g. WineDescriptor has an equivalent anonymous class that is defined as an union of WineTaste and WineColor classes). In Figure 8b we show the OWL description for the *locatedIn* property, which has the Region class as its range, and both Region and Wine classes as its domain. Figure 8c contains OWL instances defined as statements' parts in Figure 4. This feature empowers our solution to generate both ontology armature (classes, properties, etc.) and ontology instances (body of knowledge). This feature is not supported in other MDA-based proposals for ontology development.

<pre> <owl:Class rdf:ID="WineDescriptor"> <owl:equivalentClass> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#WineTaste"/> <owl:Class rdf:about="#WineColor"/> </owl:unionOf> </owl:Class> </owl:equivalentClass> </owl:Class> <owl:Class rdf:ID="WineTaste"> <rdfs:subClassOf rdf:resource="#WineDescriptor"/> </owl:Class> <owl:Class rdf:ID="WineColor"> <rdfs:subClassOf rdf:resource="#WineDescriptor"/> <owl:equivalentClass> <owl:Class> <owl:oneOf rdf:parseType="Collection"> <WineColor rdf:about="#Red"/> <WineColor rdf:about="#Rose"/> <WineColor rdf:about="#White"/> </owl:oneOf> </owl:Class> </owl:equivalentClass> </owl:Class> </pre> <p style="text-align: center;">a)</p>	<pre> <owl:ObjectProperty rdf:ID="locatedIn"> <rdfs:range rdf:resource="#Region"/> <rdfs:domain rdf:resource="#Region"/> <rdfs:domain rdf:resource="#Wine"/> </owl:ObjectProperty> <owl:Class rdf:ID="Wine"> <!-- ... --> <rdfs:subClassOf rdf:resource="#PotableLiquid"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#locatedIn"/> <owl:someValuesFrom rdf:resource="#Region"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class> <Region rdf:ID="SonomaRegion"/> <Region rdf:ID="CaliforniaRegion"/> <Region rdf:ID="MendocinoRegion"> <locatedIn rdf:resource="#SonomaRegion"/> <adjacentRegion rdf:resource="#CaliforniaRegion"/> </Region> </pre> <p style="text-align: center;">b)</p>
	<pre> <Region rdf:ID="SonomaRegion"/> <Region rdf:ID="CaliforniaRegion"/> <Region rdf:ID="MendocinoRegion"> <locatedIn rdf:resource="#SonomaRegion"/> <adjacentRegion rdf:resource="#CaliforniaRegion"/> </Region> </pre> <p style="text-align: center;">c)</p>

Fig. 8. The resulting OWL description: a) classes generated for the OUP model from Figure 2; b) OWL's Object property descriptors for the model from Figure 3 c) OWL statements from Figure 4

5.2. Practical experience

The developed solution acts as an extension for standard UML tools and thus enables us to create complete OWL ontologies without need to use ontology-specialized development tools. We have decided to use *Poseidon for UML* since it supports all requirements for OUP. We have tested our solution on the well-known example of the Wine ontology. Firstly, we represented this ontology in *Poseidon for UML* using OUP. Then we exported this extended UML into XMI, and after performing the XSLT, we obtained an OWL document. Finally we imported this document into Protégé using its OWL plugin. The current XSLT version has a limitation since it does not support packages (i.e. the OUP multi-ontology development).

So far, we have developed two ontologies using OUP that we later transformed in OWL using the XSLT. These two ontologies are: the ontology of Saints and philosophers, and the Petri net ontology. The first ontology was developed using the method - Three of Porphyry. The Petri net ontology was developed in order to provide the Semantic Web support for Petri nets [20].

6. Related work

In Table 1 we give an overview of current solutions that address the problem of bridging MDA and ontology languages, their formal definition, kinds of model interchange description they use, proposals for mapping implementation, and target ontological languages. Cranefield [12] has found connections between the standard UML and ontological concepts: classes, relations, inheritance, etc. He provided a practical software support in the form of two XSLTs that were developed to enable transformation of the UML XMI format to RDFS and Java classes. Backlawski and his colleagues have introduced two approaches for ontology development. The first one extends the UML metamodel by introducing new meta-classes [21]. The second one has an independent ontology metamodel defined using MOF, which they named Unified Ontology Language (UOL) [22]. Falkovych and her associates [23] used an UML-separated hierarchy to define kinds of ontology properties in order to enable transformation of UML models into equivalent DAML+OIL descriptions. A practical mapping from UML models to the DAML+OIL is implemented using the XSLT.

Table 1. An overview of present UML and MDA based ontology development frameworks and their transformations to the Semantic Web languages

Approach	Metamodel	Model description	Transformation mechanism	Generated ontology language
<i>CraneField</i>	Standard UML	UML XMI	XSLT	RDFS, Java classes
<i>Baclawski et al</i>	UML Profile, MOF-based ontology language	(not given - UML XMI and MOF XMI can be used)	-	DAML
<i>Falkovych et al</i>	Standard UML	UML XMI	XSLT	DAML + OIL
<i>Protégé</i>	Protégé metamodel	Protégé XMI	Programmed (Java, XML parser, Jena, MDR)	OWL, RDF(S), DAML+OIL, XML, UML XMI, Protégé XMI, ...
	Standard UML	UML XMI		

7. Conclusions

In this paper we showed a formal approach in which we tried to make closer MDA-based ontology languages and the W3C's OWL recommendation using the idea of technological spaces. We hope that this work can be useful as a practical contribution to the OMG's efforts in finding a suitable MDA-based technique for the Semantic Web ontologies that will bring ontology development process closer to software engineers.

In the future we will finish our current work aimed at providing support for transformations between the OUP (i.e. the UML XMI format) and the ODM (i.e. the ODM specific XMI format), as well as between OWL and ODM. In this way, we will have an entire metamodeling platform compliant to the OMG's ontology initiative.

References

- Kogut, P., et al: UML for Ontology Development, *The Knowledge Engineering Review*, Vol. 17, No. 1 (2002) 61-64
- Ontology Definition Metamodel Request for Proposal, *OMG Document: ad/2003-03-40*, (2003)
- Miller, J. and Mukerji, J., (eds.): *MDA Guide Version 1.0*, *OMG Document: omg/2003-05-01*, http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf (2003)
- Kurtev, I. Et al: *Technological Spaces: An Initial Appraisal*, In *Proceedings of the Confed. Int'l Conferences CoopIS, DOA, and ODBASE 2002*, Irvine, CA, USA (2002)
- Decker, S. et al: *The Semantic Web: The Roles of XML and RDF*, *IEEE Internet Computing*, Vol. 4, No. 5 (2000) 63-74
- Seidewitz, E.: *What Models Mean*, *IEEE Software*, Vol. 20, No. 5 (2003) 26-32
- Atkinson, C., Kühne, T.: *Model-Driven Development: A Metamodeling Foundation (Spec. issue on Model-Driven Development)*, *IEEE Software*, Vol. 20, No. 5 (2003) 36-41
- Duddy, K.: *UML2 Must Enable A Family of Languages*, *Communications of the ACM*, Vol. 45, No. 11 (2002) 73-75
- Atkinson, C. and Kühne, T.: *Profiles in a strict metamodeling framework*, *Science of Comp.Prog.*, Vol. 44, No. 1 (2002) 5-22
- MOF 2.0 Query/Views/Transformations Request for Proposal, *OMG Document ad/2002-04-10*, (2002)
- Bézivin, J.: *First experiments with the ATL model transformation language: Transforming XSLT into XQuery*, In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the context of MDA*, Anaheim, CA, USA (2003)
- CraneField, S.: *Networked Knowledge Representation and Exchange using UML and RDF*, *Journal of Digital information*, Vol. 1, No.8 (2001) <http://jodi.ecs.soton.ac.uk>
- Djuric, D. et al: *Ontology Modeling and MDA*, *Journal on Object Technology*, Vol. 4, No. 1 (2005) forthcoming.
- Bechhofer, S. et al: *OWL Web Ontology Language Reference*, *W3C Recommendation*, <http://www.w3.org/TR/2004/REC-owl-ref-20040210> (2004)
- Bézivin, J.: *From Object Composition to Model Transformation with the MDA*, In *Proceedings of the 39th Int'l Conference and Exhibition on Technology of Object-Oriented Languages and Systems*, Santa Barbara, USA (2001) 350-355.
- Stuckenschmidt, H. et al: *Exploring Large Document Repositories with RDF Technology: The DOPE Project*, *IEEE Intelligent Systems*, Vol. 19, No. 3 (2004) 34-40.
- McBride, B.: *Jena: A Semantic Web Toolkit*, *IEEE Internet Computing*, Vol. 6, No. 6 (2002) 55-59.
- Kurtev, I. and van den Berg, K.: *Model Driven Architecture based XML Processing*, In *Proc. of the ACM Symp. on Document Engineering*, Grenoble, France (2003) 246-248.
- Gašević, D. et al: *Converting UML to OWL ontologies*, In *Proceedings of the 13th International WWW Conference*, NY, USA (2004) forthcoming
- Gašević, D. and Devedžić, V.: *Reusing Petri Nets Through the Semantic Web*, In *Proc. of the 1st European Semantic Web Symposium Heraklion, Greece* (2004) forthcoming.
- Baclawski, K. et al: *Extending the Unified Modeling Language for ontology development*, *Int. J. Software and Systems Modeling*, Vol. 1, No. 2 (2002) 142-156
- Baclawski, K. et al: *UOL: Unified Ontology Language*, *Assorted papers discussed at the DC Ontology SIG meeting*, <http://www.omg.org/cgi-bin/doc?ontology/2002-11-02> (2002)
- Falkovych, K. et al: *UML for the Semantic Web: Transformation-Based Approaches*, *Knowledge Transformation for the Semantic Web*, IOS Press, Vol. 95 (2003) 92-106
- Ceccaroni, L. and Kendall, E.: *A graphical environment for ontology development*, In *Proceedings of The Second International Joint Conference on Autonomous Agents & Multiagent Systems*, Melbourne, Australia (2003) 958-959