Converting UML to OWL Ontologies

Dragan Gašević, Dragan Djurić, Vladan Devedžić
University of Belgrade, POB 52, Jove Ilića 154, Belgrade, Serbia and Montenegro
gasevic@yahoo.com, dragandj@mail.ru, devedzic@galeb.etf.bg.ac.yu

Violeta Damjanović
Postal Savings Bank, 27.marta 71, Belgrade, Serbia and Montenegro
vdamjanovic@posted.co.yu

ABSTRACT
This paper presents automatic generation of the Web Ontology Language (OWL) from an UML model. The solution is based on an MDA-defined architecture for ontology development and the Ontology UML Profile (OUP). A conversion, that we present here, transforms an ontology from its OUP definition (i.e. XML Metadata Interchange – XMI) into OWL description. Accordingly, we illustrate how an OUP-developed ontology can be shared with ontological engineering tools (i.e. Protégé).

Categories and Subject Descriptors
I.2.5 [**Artificial Intelligence**] Programming Languages and Software – *Expert system tools and techniques*

General Terms
Design

Keywords
Ontology, UML Profiles, XSLT, OWL

1. INTRODUCTION
In order to bring ontology development process closer to wider software engineering population, some authors propose usage of software engineering techniques, especially the UML since it is the most accepted software engineering standard [5]. However, none of current solutions of this problem supports full ontology definition: definition of non-limited degree of property hierarchy, modeling ontology instances, etc. We believe that these limitations can be overcome using UML's extensions (i.e. UML profiles) [3], as well as other OMG's standards (e.g. *Model Driven Architecture – MDA*).

Accordingly, we have implemented an *eXtensible Stylesheet Language Transformation (XSLT)* that transforms the *XML Metadata Interchange (XMI)* representation of a UML Profile (i.e. *The Ontology UML Profile - OUP*) into the forthcoming *Web Ontology Language (OWL)* [1]. With this UML's model transformation we can extend present UML tools, so they can be used for full ontology development without need for other ontological tools. This solution is a part of the Good-Old-AI research group (http://goodoldai.org.yu) efforts to develop an ontology architecture based on the OMG's initiative [6].

2. FORMAL FRAMEWORK OF OUR SOLUTION
An OMG's initiative should define a suitable framework for ontology development using the MDA standards [6]. According to this we give a proposal of such architecture [2]. The core of this architecture is the Ontology Definition Metamodel (ODM). Here we shortly overview the OUP that is based on the ODM.

`Class` is one of the most fundamental concepts in ODM and OUP. In ODM, Ontology Class concept is represented as an instance of the Meta-Object Facility (MOF) `Class`, and has several concrete species, a: `Class`, `Enumeration`, `Union`, `Intersection`, `Complement`, `Restriction`, and `AllDifferent`. In Figure 1 we show a part of the well-known Wine ontology. `WineDescriptor` is equivalent to the union of classes `WineTaste` and `WineColor`, whereas `WineColor` is an enumeration of `WineColor` instances: `White`, `Rose`, and `Red`. We should note that we have two anonymous classes (`Union` and `Enumeration`). See [2] for details about the OUP.

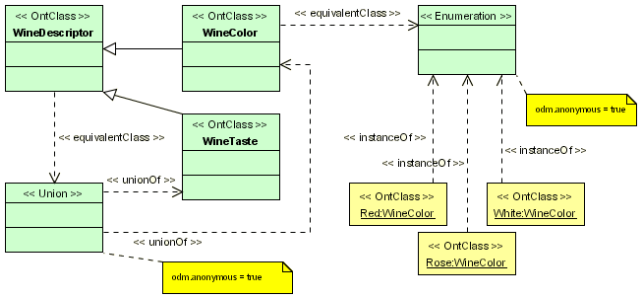

Figure 1. The Ontology UML profile class-oriented stereotypes (an excerpt of the Wine ontology)

3. OVERVIEW OF OUR SOLUTION: XSLT
The main idea of having an UML profile for ontology development is to use present UML tools. In fact, current UML tools mainly support XMI standard – an MDA's XML-based standard for metametamodel, metamodel, and model sharing. Since this format is XML-defined one can employ XSLT to transform XMI documents into target documents that must not be XML documents. These target documents can be written in some ontology language, for example OWL. On the other hand, when we use an approach based on XSLT (i.e. XSLT principle) we do not need to change a UML tool, instead we just apply an XSLT on an output document of the UML tool. Accordingly, we can use well-defined XML/XSLT procedure that is shown in Figure 2.

A UML tool (e.g. Poseidon for UML) can export an XMI document that an XSLT processor can use as the input. 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. On the other hand, since we obtain an OWL described document, we do not need to use any ontology tool, instead we are able to use this ontology description as a final OWL ontology.

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

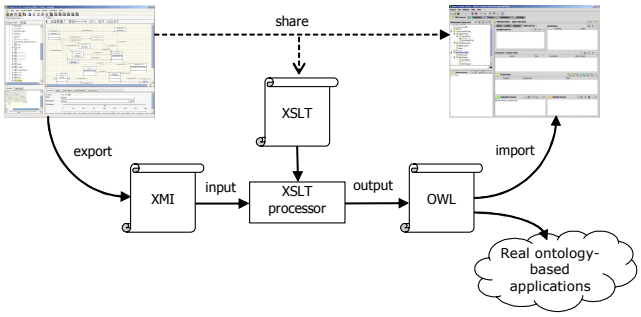

Figure 2. Used XSLT principle: extensions of present UML tools for ontology development

The XSLT, we have implemented for mapping from XMI (OUP-based) format to the OWL XML 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. It is especially difficult because each UML construct is a different stereotype.
- 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 and classes that can be referenced using their ID. Accordingly, we included into OUP *odm.anonymous* tagged values that help us detect these two cases.

In order to depict an output OWL document that we obtain as the XSLT's result, we give Figure 3. This figure shows 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).

4. EXPERIENCES

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 the ODM. We decide to generate OWL ontologies in the fashion similar to the Protégé's OWL plugin. Hence, we have managed to provide an additional way to import Poseidon's models into Protégé through the OWL. Of course, since Protégé has more advanced features for ontology development, an OUP-defined ontology can be further refined.

We have tested our solution on the well-known example of the Wine ontology. Firstly, we represented this ontology in Poseidon 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). Actually, the OUP supports multiple ontologies within the same XMI project, but the XSLT standard and XSLT processors introduce this limitation.

```
<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>
```

Figure 3. Resulting OWL description –classes generated for the OUP model from Figure 2

Currently, we have developed two ontologies using the 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 Porphyry's tree method. The Petri net ontology was developed in order to provide the Semantic Web support for Petri nets [4].

5. CONCLUSIONS

We believe that this solution can be useful to all software engineering practitioners who participate in an ontology development process. Using well-known UML syntax, the practitioners do not need to learn how to use ontology tools. In the future we are planning to improve current implementation, so that it can support development of multiple ontologies (using UML's packages), and show how the Ontology UML Profile can be used for modular ontology development (on the example of the Petri net ontology and Petri net dialects).

6. REFERENCES

[1] Bechhofer, S. et al (2004). OWL Web Ontology Language Reference, W3C Recommendation [Online]. Available: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[2] Djurić, D. et al, Ontology Modeling and MDA. Accepted for pub. in Journal on Object Technology 4, 1 (2005).

[3] Duddy, K. UML2 Must Enable A Family of Languages, Communications of the ACM 45, 11 (Nov. 2002), 73-75.

[4] Gašević, D., and Devedžić, V., Reusing Petri Nets Through the Semantic Web, in Proc. of the 1st European Semantic Web Symposium (Heraklion, Greece, 2004) accepted for publication.

[5] Kogut, P., et al. UML for Ontology Development, The Knowledge Engineering Review 17, 1 (2002), 61-64.

[6] OMG Ontology Definition Metamodel-RFP (2003) [Online]. Available: http://www.omg.org/cgi-bin/doc?ad/2003-03-40