TRANSFORMATION OF OBJECT-ORIENTED CODE INTO SEMANTIC WEB USING JAVA ANNOTATIONS

Petr Ježek and Roman Mouček
Department of Computer Science and Engineering, University of West Bohemia, Pilsen, Czech Republic

Keywords: OWL, RDF, Semantic web, Object-oriented code, Java annotations, Semantic representation, Data source, Transformation, Restriction mapping, Properties mapping, Relations mapping, Semantic description.

Abstract: This paper deals with difficulties occurring during transformation of schema and data from an object-oriented code to a semantic web representation (RDF, OWL). The authors describe differences in semantic expressivity between the object-oriented approach and the semantic web approach and look for the ways to fill this semantic gap. Then some existing approaches with their difficulties are introduced and a preliminary idea using Java annotations is proposed. Java annotations add missing semantic information into Java code, which is consequently processed by the proposed framework and serialized into output semantic web structure (OWL).

1 INTRODUCTION

At present World Wide Web (WWW) is gradually reaching its limits because description of data semantics is missing. An evolving extension of WWW, the Semantic Web (Berners-Lee, 2001), uses a triple oriented representation described by Resource Description Framework (RDF). RDF triples consist of a subject, a predicate and an object with an assertion that a subject has a property with a value. This data representation is suitable for processing using software machines.

Since expressivity of RDF schema is insufficient in many application domains, there exists an extension called Ontology Web Language (OWL). OWL uses the same RDF syntax and adds the ability to express more information about the characteristics of properties and classes.

Although the idea of the semantic web is promising in the software development, new technologies and object oriented programming (OOP) itself are based on different approaches. Since object-oriented programming is the main stream in the software development and data in current systems are usualy stored in relational databases, a transformation from common structures into the Semantic web is required.

Several aproaches to transform an object oriented code to the semantic web exist. These aproaches with their difficulties are introduced in the second and third sections. The fourth section describes a proposed extension of Java language providing a richer semantic expresivity. The proposed framework ensuring data transformation is also presented.

2 OWL AND OOP DIFFERENCES

2.1 Close/Open World Assumptions

Semantics of classes and instances in RDF is based on description logic and an open-world assumption while object oriented type system is defined as the closed world. In the open-world assumption any web based ontology can add subclasses or additional characteristics to concepts defined in other ontologies. It is not possible in closed systems as Java language is (program code is defined on a close finite domain). These assumptions bring different views on classes, instances and properties in both representations.

2.2 Classes and Instances - Differences

Classes in OOP are regarded as types for instances where each instance has one class as its type. Instances cannot change their type at runtime.

Ježek P. and Mouček R..
TRANSFORMATION OF OBJECT-ORIENTED CODE INTO SEMANTIC WEB USING JAVA ANNOTATIONS.
DOI: 10.5220/0003468602070210
In *Proceedings of the 13th International Conference on Enterprise Information Systems* (ICEIS-2011), pages 207-210
ISBN: 978-989-8425-56-0

Compilers are used at build-time. OWL classes are regarded as a set of individuals where each individual can belong to multiple classes. The class membership can be created and changed in runtime. The class consistency is checked using reasoners.

2.3 Properties - Differences

OOP properties (class fields) are defined locally to class where instances can have values only for the attached properties. Classes encode much of their meaning through methods; class fields are accessible by get/set methods.

OWL properties are stand-alone entities; they can exist without classes. Instances can have arbitrary values. Classes make their meanings explicit in term of statements. All OWL classes and properties are public (OWL Primer, 2006).

3 OOP TO OWL MAPPING

We tested several tools that transform data from an object-oriented code to an OWL representation. These tools were described in (Mouček and Ježek, 2010). From the set of tested tools we selected JenaBean (JenaBean, n. d.) integrated with OWL API (OwlApi, n. d.). By using selected tools we are able to transform an object-oriented model into the semantic web representation. This selection and integration with preliminary results was described more in depth in (Ježek and Mouček, 2010).

Concerning one side transformations the selected tools work quite satisfactorily because object-oriented code has poorer semantics than OWL. However, if we want to use more capabilities of OWL, we have to enrich object-oriented code by missing semantics.

There are several frameworks and tools which try to enrich object-oriented code with additional semantic information which appears in OWL output structure. Some tools exist only as initial proposals while some of them are really implemented.

3.1 ActiveRDF

ActiceRDF is a library for accessing RDF data from Ruby programs. ActiveRDF provides a domain specific language for RDF models; it can address RDF resources, classes and properties programmatically without using e.g. Sparql queries. (Oren, Delbru, Gerke, Haller and Decker, 2007).

This tool solves only a part of OWL and OOP mismatches due to the usage of Ruby that is a dynamic interpreted language. Namely developed framework doesn't need strictly typed classes and properties. Types are evaluated in runtime and can be changed dynamically. An availability to add additional semantics into source codes is missing.

3.2 Semantic Object Framework

Semantic object framework (SOF) utilizes embedded comments in source codes to describe semantic relationships between classes and attributes. Heterogeneous data sources could be processed using implemented parsers (Po-Huan, Chi-Chuan, Kuo-Ming, 2009).

This approach seems to be promising but programmer has to insert RDF/OWL keywords into source code comments directly. It can be an obstacle for object-oriented developers. Moreover, source comments should be used for description of the class meaning, not for insertion of different language syntax.

3.3 eClass

EClass is a solution that changes Java syntax to embed semantic descriptions into the source code. The eClass contains data attributes, methods, inference rules and presentations. It can be implemented as an extension of an existing object-oriented programming language (Liu, Wang, Dillon, 2007).

However, when a commonly used programming syntax is changed, it affects compilers and virtual machines. It is an obstacle to use it in common systems.

4 ANNOTATION FRAMEWORK

4.1 Prerequisites

According to difficulties mentioned above we decided to propose a custom annotation framework which allows us to annotate a common object-oriented language (Java) and provide a transformational mechanism translating annotations into OWL output.

We suppose that the proposed framework will be also used by software engineers and not only by experts in the semantic web field. Thus a framework based on common programming technologies is preferred.

Java annotations (JavaAnnotations, n.d.) are very popular in current software developments. Since

they are also used by many frameworks (Spring, Hibernate, etc.) we decided to implement their language extension as well.

4.2 Annotations Design

4.2.1 Restrictions Mapping

OWL restrictions are difficult to express in Java, hence we defined a set of annotations describing these restrictions. OWL property restrictions have the following general syntax:

```
<owl:Restriction>
     <owl:onProperty rdf:resource="some
     property" />
     (Constraints)
</owl:Restriction>
```

We can define the annotation "Restrictions" analogically:

```
public class SomeClass {

  @Restrictions({"Array of
  Constraints"})
private Object someProperty;
}
```

Let suppose we want to get a serialization of ontology below:

```
<owl:Restriction>
  <owl:onProperty
  rdf:resource="#hasChild" />
  <owl:someValuesFrom
  rdf:resource="#Student" />
</owl:Restriction>

<owl:Restriction>
  <owl:onProperty
  rdf:resource="#hasChild" />
  <owl:allValuesFrom
  rdf:resource="#Child" />
</owl:Restriction>
```

If we suppose the existence of Java classes Person, Child and Student, where Person is a superclass for Child and Student classes, we can define restrictions using Java annotations analogically:

```
public class Person {
   @Restrictions({
      @Restriction(allValuesFrom=
      Child.class),
      @Restriction(someValuesFrom
      =Student.class)
   })
   private Person hasChild;
}
```

4.2.2 Intersection, Union and Complement Mapping

The three types of set-operators can be viewed as representation the AND, OR and NOT operators on classes. OWL intersection and union could be represented using java collections. For OWL complement we defined the corresponding Java annotation. The example below represents the complement of Student class.

```
<owl:Class>
  <owl:complementOf>
    <owl:Class rdf:about="#Student"/>
  </owl:complementOf>
</owl:Class>
```

In Java code we defined the annotation "Complement". Let suppose that the class Employee is inherited from the class Person.

```
@Complement(Student.class)
public class Employee extends Person {
}
```

4.2.3 Relations to Other Properties Mapping

OWL provides the constructs equivalentProperty and inverseOf. We defined a set of Java annotations for these constructs as well. Let consider the following ontology:

```
<owl:ObjectProperty rdf:ID="hasChild">
  <owl:inverseOf
  rdf:resource="#hasParent"/>
</owl:ObjectProperty>
```

Let suppose that the class Parent is inherited from the class Person. We express the inverse property of OWL example above using Java code below.

```
@InverseOf(Parent.class)
public class Child extends Person {
}
```

4.3 Properties Mapping

Because properties are standalone entities in RDF and OWL we utilized interfaces in Java for expressing these properties. We defined an interface with a method obtaining a property value (classes have to implement this method). This approach ensures that more OWL classes can share one OWL property because more Java classes can implement one Java interface. Interface and implemented class look as follows:

```
public interface HasAge {
    public int getAge();
}
public class Person implements HasAge {
private int age;

@Override
public int getAge() { return age; }
}
```

The code above is serialized into the following form:

```
<owl:ObjectProperty
     rdf:about="#HasAge">
<rdfs:domain rdf:resource="#Person"/>
<rdfs:range
     rdf:resource="http://www.w3.org/20
     01/XMLSchema#integer"/>
</owl:ObjectProperty>
```

5 CONCLUSIONS

Many scientific papers deal with a domain description using a specific ontology. These ontologies serve as recognizable data sources accessible by automatic software readers. However, current software systems are usually object-oriented and they operate over large data collections usually stored in relational databases.

Since fundamental differences between semantics of object-oriented code and OWL exist, there is necessary to ensure a suitable mapping.

Because expressive capabilities of OWL are richer than in the case of object-oriented we are looking for the ways to fill this semantic gap.

We investigated several approaches described in this paper. The most of tested frameworks are difficult to use either because added semantic information is insufficient or confused, or the usage of modified compiler or interpreter is required.

As the result we presented an idea based on the concept of Java annotations that can be deployed without substantial difficulties. Our solution is an initial proposal using current Java technologies. It covers essential semantic gaps between mentioned representations. In the near future, we plan to capture more semantic differences to provide a richer semantic description of object-oriented code using annotations. At the same time we plan to develop a framework ensuring fully automated transformation. This approach is going to be realized within development of EEG/ERP database (Ježek and Mouček, 2010) and its registration as a recognized data source within Neuroscience Information Framework (Gupta, 2008).

ACKNOWLEDGEMENTS

This work was supported by grants Ministry of Education, Youth and Sport of the Czech Republic under the grant ME 949 and UWB grant SGS-2010-038 Methods and Applications of Biomedical Informatics.

REFERENCES

Berners-Lee, T., 2001. The Semantic Web. In *Scientific American* (pp. 34-43).

RDF Resource Description Framework (RDF). 2004. Retrieved January, 2011, from http://www.w3.org/RDF/

OWL Ontology Web Language (OWL). 2000. Retrieved January, 2011, from http://www.w3.org/TR/owl2-overview/

Semantic Web for Object-Oriented Software Developers (OWL Primer), 2006, Retrieved January, 2011, from http://www.w3.org/TR/sw-oosd-primer/

Mouček, R., Ježek, P., 2010. System for Storage and Management of EEG/ERP Experiments – Generation of Ontology. In *12th International Conference on Enterprise Information System, Volume 1* (pp. 415-420)

Ježek, P., Mouček, R., 2010. EEG/ERP Portal – Semantic Web Extension. In *Second Global Congress on Intelligent Systems* (pp. 392-395)

JenaBean, n.d. Retrieved January, 2011, from http://www.ibm.com/developerworks/java/library/j-jenabean.html

OwlApi, n.d. Retrieved January, 2011, from http://owlapi.sourceforge.net/

Oren, E., Delbru, R., Gerke, S., Haller, A. and Decker S., 2007. ActiveRDF: object-oriented semantic web protramming, *In Proceedings of the 16th international conference on World Wide Web* (pp. 817-824)

Po-Huan, Ch., Chi-Chuan, L., Kuo-Ming, Ch., 2009. Integrationg Semanic Web and Object-Oriented Programming for Cooperative Desig, In *Journal of University Computer Science, vol. 15, no. 9*

Liu, F., Wang, J., Dillon, S. T., 2007. Web Information Representation, Extraction and Reasoning based on Existing Programming Technology, In *Computational Inteligence 37* (pp. 147-168)

JavaAnnotations, n.d. Retrieved January, 2011, from http://download.oracle.com/javase/1.5.0/docs/guide/language/annotations.html

Ježek, P., Mouček, R., 2010. Database of EEG/ERP experiments, In *Healthinf 2010 – Proceeding Third International Conference on Health Informatics.* Valencia Spain.

Gupta, A., Bug,W., Marenco, L., Qian, X., Condit, C., Rangarajan, A., et al., 2008. Federated access to heterogeneous information resources in the Neuroscience Information Framework (NIF), In *Neuroinformatics*, Volume 6, Number 3, (pp. 175-194)