A New Method for Mapping UML Class into OWL Ontology

Noreddine Gherabi
Hassan 1 University, FSTS
Department of Mathematics and Computer Science

Mohamed Bahaj
Hassan 1 University, FSTS
Department of Mathematics and Computer Science

ABSTRACT

In this paper we describe how UML schemes can be converted into OWL Ontology, thus enabling reasoning on them by Semantic Web applications. The proposed solution is based on a three phases approach, the first step is to present the class diagram in the mathematical formulation and the second one is converting the UML Class into encoded text file, finally, the structure of the classification scheme is converted into OWL ontology. We demonstrate the practical applicability of our approach by showing how the results of reasoning on these OWL ontology can help improve the Web systems.

Keywords- component; RDF, OWL, Web ontologie,UML

1.INTRODUCTION

This document aims to define a correspondence between the class diagrams of Unified Modeling Language (UML) [1] and OWL using the mathematical representation of the class diagram.

The OWL Web Ontology Language [2] is designed for use by applications that need to process the content of information instead of just presenting information to humans. The use of ontology Web is growing rapidly since the emergence of the Semantic Web. Most of the data is still modeled and stored in relational databases / objects and therefore out of reach for many applications of the Semantic Web. A critical requirement for the evolution of the current Web of documents into a Web of Data is the inclusion of the vast quantities of data stored in Relational Databases (RDB). The mapping of these vast quantities of data from RDB to the Resource Description Framework (RDF) has been the focus of a large body of research work in diverse domains. Some of the most notable approaches of this kind are R2O [3], D2RQ [4], Virtuoso RDF Views [5, 6] and DartGrid [7].

The R2O approach defines declarative and extensible language (in xml) to describe mapping between given RDB and an OWL ontology or RDFS schema so that tools can process this mapping and generate triples that correspond to source RDB data.

G.Bumans et al [8] demonstrate a very simple standard SQL-based RDB to RDF/OWL mapping approach that is based on defining correspondence between the tables of the database and the classes of the ontology, as well as between table fields/links in the database and datatype/object properties in the ontology.

UML models cannot be easily exchanged over the Web, the reasoning possibilities with UML models being also quite restricted. [9] and [10] propose a simples transformations between UML and ontology representation languages.

Recently, some efforts arose in bringing together UML and the Semantic Web. Nevertheless, most of these approaches do not use all proprieties of the UML diagrams as a data source.

In this direction several researches are underway to transform data into Web semantic (XML/RDF/OWL). J. Fong et al [11] proposes a method for Converting relational database into XML data with DOM. Wu and Hsieh [12] used a technique for mapping UML to XML. They created XSD from class diagrams, but this technique is very complex to generate the XSD file for each class diagram.

In our previous work [13], we have developed a Framework for converting a class diagram into an XML document and show how to use Web files for the design of data based on the classification UML.

Our approach for mapping is based on several steps. First step is to formulate the class diagram into a mathematical structure and transform it into an encoded text file (Section II). After the text file is imported into the system to build the structure of OWL diagram using the algorithm called GenarateOWL described in Section III, and finally the exported OWL file will be validated and stored

2.UML FOR THE SEMANTIC WEB

A. Mathematical representation of UML Class.

Modeling Layer allows the representation of the UML model into a logical representation. For this task, we have applied a mathematical approach for semantically enriching of the UML Model.

In our approach, a class diagram in UML is represented as a set of classes, denoted as 4-tuple:

$$UML-C := \{C | C := (C_N, C_A, C_O, C_R \ \}$$

Where:

C_N : is the name of the class C.

C_A : is the list of attributes associated with this particular class

$$C_A := \{A | A := (A_n, A_t, A_v, A_d \ \}$$

Where A_n is an attribute name, A_t is its type, A_v is the visibility of this attribute (Public, Private or protected) and A_d is a default value if given.

C_O : is a set of operations for defined class C

$$C_O := \{O | O := (O_n, O_t, O_v)\}$$

Where O_n is the name of the operation O, O_t its type and O_v is the visibility of this operation.

C_R : describes the different types of relations that can exist between any pair of classes in the UML diagram.

$$C_R := \{R | R := (R_t, R_c, R_s, R_{tag})\}$$

Where R_t is the type of relationship (Association, Composition, Aggregation or Generalization), R_c is the cardinality specified for the source class, R_s is a source class and R_{tag} defined the target class with which the source class is connected.

B.Mapping process

We now define the sets to describe the structure ontology Web.

When the structure of UML has been obtained, the schema translation starts by applying an appropriate set of rules to map the UML constructs into Web ontology.

The mapping process is done progressively as follows: It starts by mapping the classes of the diagram to OWL classes, then mapping the attributes of each class to data type properties and the relationships into Object type properties.

The mapping process consists therefore of the following steps:

- Each class in the class diagram is translated to owl:class in the Web ontology, a class in OWL technology is represented as follows:

$$< \text{owl:Class rdf : ID ="} C \in C_N \text{"} / >$$

- Translating attributes: Each attribute A is translated into a owl:DatatypeProperty class and represented as :

$$< \text{owl:DatatypePpperty rdf : ID ="} A \in C_A \text{">}$$
$$< \text{rdfs:domain rdf : resource ="\#} C \in \text{UML-C"} / >$$
$$< \text{rdfs:range rdf : resource ="\&xsd, Type} \in A_t \text{"} / >$$
$$< \text{/owl:DatatypePpperty} >$$

- UML Class relationships are translated and defined among Web Ontology based on its type as association, aggregation, composition and inheritance. The association, aggregation and composition are transformed into owl:ObjectProperty class

For example: Consider the relationship between two classes C1 and C2, the representation of the relationship in Web ontology is represented as follows:

$$< \text{owl:ObjectProperty rdf : ID ="Rel} \in C_R \text{">}$$
$$< \text{rdfs:domain rdf : resource ="\#} C_1 \in \text{UML-C"} / >$$
$$< \text{rdfs:range rdf : resource ="\#} C_2 \in \text{UML-C"} / >$$
$$< \text{/owl:ObjectProperty} >$$

- Inheritance: There are in any OWL ontology a superclass named “Thing”, which all other classes are subclasses. With rdfs:subClassOf we can specify a class C1 to be subclass of another class C2; then every instance of C1 is also an instance of C2.

For example, consider the class C1 inherits from C2, the Web Ontology representation is as follows:

$$< \text{owl:Class rdf : ID ="C1} \in R_{tag} \text{">}$$
$$< \text{rdfs:subClassOf rdf : resource ="\#C2} \in \text{Rs"} / >$$
$$< \text{/owl:Class} >$$

- Mapping cardinality: The cardinalities of a relationship are given by specifying minimum and maximum cardinalities.

  For mapping the general cardinality we use:

$$< \text{owl:cardinality rdf : datatype="\&xsd, nonNegativeInteger"} >$$
$$\text{C}ardinality \in R_c < \text{/owl:cardinality} >$$

And for mapping the maximal cardinality of each relationship we use this syntax:

$$< \text{owl:maxCardinality rdf : datatype="\&xsd,}$$
$$\text{nonNegativeInteger"} > \text{C}ardinality \in R_c < \text{/owl:maxCardinality} >$$

3.ALGORITHM FOR GENERATION OF OWL SCHEMA

This subsection presents the GenerateOWL algorithm. Given an UML Class as input, the algorithm goes through a main loop to classify UML Class constructs and generate their equivalents in Web Ontologie

```
Algorithm GenerateOWLs
Input : UML Class
OutPut: OWL Structure
    • CodifyDiagram() // a sub-function to codify the structure
      of the class diagram in a text file
    • ValidateClassDiagram() // this function verify the
      structure of the diagram and then validate it
    • GenerateOWLSchema()//generate all elements of the
      diagram and create a valid OWL file
    • ValidateOWLDocument()//validate the generated OWL
      file
 END.
```

The structure of the OWL generated by the algorithm corresponds to the UML class where each tag OWL is generated for each Class, each attribute and each relationship. In our algorithm the operations are not mapping.

B. Codification of the diagram

In our approach the class diagram is codified in a symbolic form as follows:

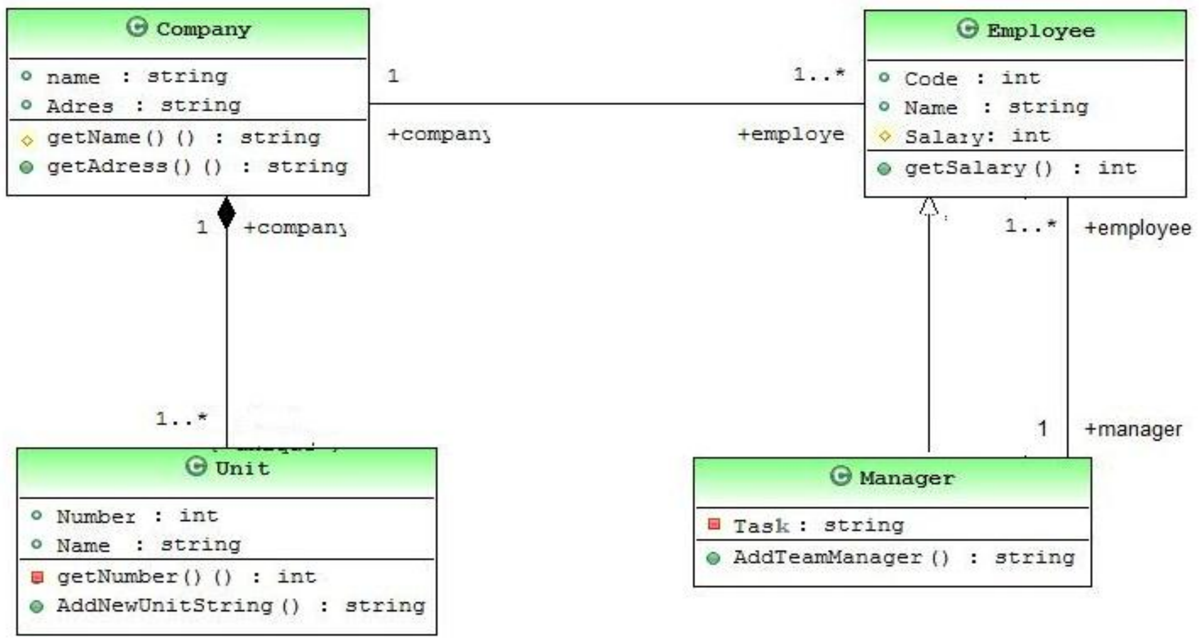

Figure 1. Example of Class Diagram

C_N ; Number-of-Attributes ; $\{C_A\}$/(An:At:Av:Ad);} ; Number-of-Operations ; {Co}/(On:Ot:Ov);} ; Number-of-Relationships ; $\{C_R\}$/(Rt:Rc:RS; Rtag);};

For example: Consider the class “Employee” codified as:

Employee;3;Code:int:Public:;Name:String:Public:;Salary:Int:Protected:;1;getSalary:int:public;2;1..*:Company;1..*:Manager;0;0;0;

In this example the class "Employee" is defined by three attributes: "Code" is the Int type, its visibility is public and its default value is undefined, same, the attribute "Name" is defined by the type String, with visibility Public and no default value, the 3rd attribute "Salary" its type is Int, visibility is protected and no default value. Class "Employee" contains a single operation "getSalary" the type of the return value is Int and its visibility is Public.

Two association relationships that exists with the classes "Company" and Manager its cardinality is 1...*.

C. Mapping OWL File

Consider a class diagram modeling of employees in a company, developed using MyEclipse, is shown in Fig. 2. This model will be used in the examples presented along the paper.
The encoded text file that contains the structure of the diagram is defined as follows:
Employee;3;Code:int:Public:;Name:String:Public:;Salary:Int:Protected:;1;getSalary:int:public;2;1..*:Company;1..*:Manager;0;0;0;
Company;2;Name:String:Public:;Adress:String:Public:;2;getName:string:Protected;getAdress:string:public;1;1..1:Employee;0;0;0;
Unit;2;Number:int:Public:;Name:String:Public:;2;getNumber:int:Private;AddNewUnit:string:Public;0;1;1..*:Company;0;0;
Manager;1;Task:string:Private:;1;AddTeamManager:String:Public;2;1..1:Employee;0;0;1;Employee;

whose nodes represent concepts and whose directed edges represent the formal relations between the concepts.

The basic graph structure of our example ontology is a taxonomy in which all classes are represented by nodes and relation between nodes represent an attribute, association, composition or inheritance relationship.

Figure 2 illustrates the OWL structure of our example.

The global mapping of the diagram is as follows:

```
<owl:Class rdf:ID="Company" />
<owl:Class rdf:ID="Employee" />
<owl:Class rdf:ID="Manager">
<rdfs:subClassOf rdf:resource="#Employee" />
</owl:Class>
<owl:Class rdf:ID="Unit">
```

This representation is not sufficient, because the OWL structure should map all elements of the UML diagram.

The representation of relationships in WEB ontology, mapping from UML class, is defined as:

```
<owl:ObjectProperty rdf:ID="Company-Employee">
<rdfs:domain rdf:resource="#Company" />
<rdfs:range rdf:resource="#Employee" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Company-Unit">
<rdfs:domain rdf:resource="#Company" />
<rdfs:range rdf:resource="#Unit" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Manager-Employee">
<rdfs:domain rdf:resource="#Manager" />
<rdfs:range rdf:resource="#Employee" />
</owl:ObjectProperty>
```

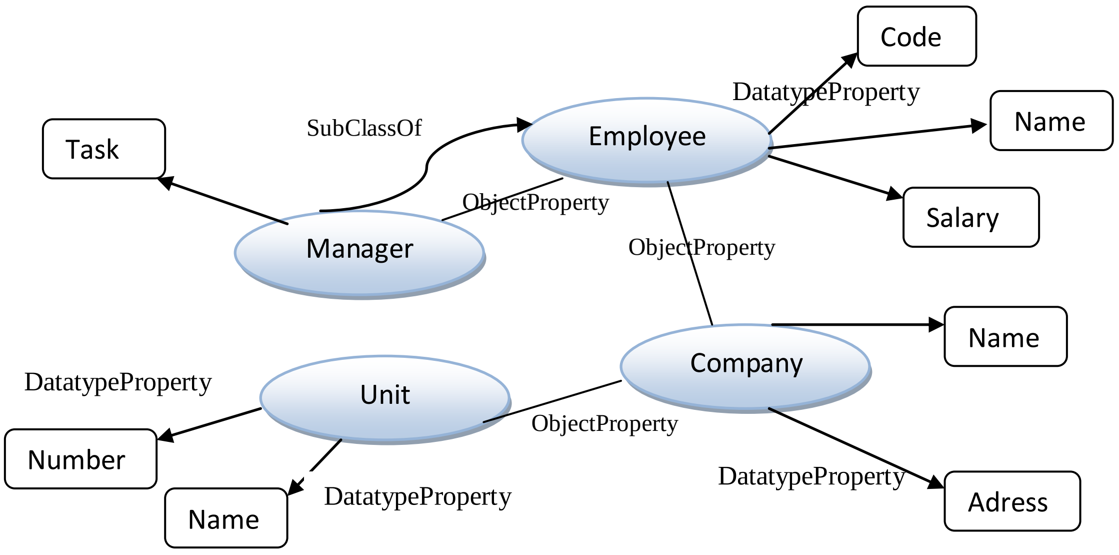

Figure 2. Overview of graphical Ontology representation.

The attributes of the class diagram are mapped as follows:

```
<owl:DatatypeProperty rdf:ID="name">
<rdfs:domain rdf:resource="#Company" />
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="Adress">
<rdfs:domain rdf:resource="#Company" />
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Code">
<rdfs:domain rdf:resource="#Employee" />
<rdfs:range rdf:resource="&xsd;Int" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Name">
<rdfs:domain rdf:resource="#Employee" />
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Salary">
<rdfs:domain rdf:resource="#Employee" />
<rdfs:range rdf:resource="&xsd;Int" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Task">
<rdfs:domain rdf:resource="#Manager" />
<rdfs:range rdf:resource="&xsd;string" />

</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Number">
<rdfs:domain rdf:resource="#Unit" />
<rdfs:range rdf:resource="&xsd;Int" />
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Name">
<rdfs:domain rdf:resource="#Unit" />
<rdfs:range rdf:resource="&xsd;string" />
</owl:DatatypeProperty>
```

```
rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinalit
y>


</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

In summary, the main contributions of this paper are listed as follows. Firstly, we have presented a new approach to discovering simple mapping between classes in UML diagram and ontology. The system captures the semantic structure contained in the UML Classes and eliminates incorrect mappings by validating mapping consistency.

Secondly, we have proposed a novel algorithm to construct contextual mappings, the algorithm reuse simple mappings between UML and web ontology.

4. CONCLUSION

In this paper, we present an efficient technique for generate an OWL structure from UML design. We analyze the existing transformation techniques using all the analysis parameters identified.

We have experimentally evaluated our approach on several examples of UML classes; the results show that the contextual mappings constructed by our approach are useful and meaningful.

5.REFERENCES

[1] M. Fowler and K. Scott, UML Distilled, 2nd Edition, Addison Wesley, Boston, 2000.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. Scientific American, May 2001.

[3] J. Barrasa, A. Gómez-Pérez. Upgrading relational legacy data to the semantic web. In: *Proc. of the 15th* International World Wide Web Conference (WWW 2006), Edinburgh, United Kingdom, 23–26 May 2006,pp. 1069–1070.

[4] D2RQ Platform. Available: http://www4.wiwiss.fuberlin.de/bizer/D2RQ/spec/.

[5] C. Blakeley. RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping). OpenLink Software, 2007.

[6] OpenLink Virtuoso Platform. Automated Generation of RDF Views over Relational Data Sources. Available: http://docs.openlinksw.com/virtuoso/rdfrdfviewgnr.html.

[7] W. Hu, Y. Qu. Discovering Simple Mappings between Relational Database Schemas and Ontologies. In: *Proc. of the 6th International Semantic Web Conference (ISWC 2007), 2nd Asian Semantic Web Conference (ASWC 2007),* Busan, Korea, 11–15 November 2007, LNCS, 4825, pp. 225–238.

[8] Mapping between Relational Databases and OWL Ontologies: an Example Computer Science and Information Technologies. Vol. 756 99–117 P. (2010)

[9] K. Falkovych. Ontology Extraction from UML Diagrams. Master's thesis, Vrije Universiteit Amsterdam, August 2002.

[10] A. Felfernig, G. Friedrich, and D. Jannach. UML as domain specific language for the construction of knowledge based configurations systems. International Journal on Software Engineering and Knowledge Engineering, 10(4):449–470, 2000.

[11] Fong, J. H.K. Wonga, Z. Cheng, Converting relational database into XML documents with DOM. Information and Software Technology. v45. 335-355,2003.

[12] I-Chen Wu, Shang-Hsien Hsieh; "An UML-XML-RDB Model Mapping Solution for Facilitating Information Standardization and Sharing in Construction Industry"; Proceedings. National Institute of Standards and Technology, Gaithersburg, Maryland. September 23-25, 2002, 317-321 pp

[13] Noreddine Gherabi and Mohamed Bahaj. Robust Representation for Conversion UML Class into XML Document using DOM. *International Journal of Computer Applications* 33(9):22-29, November 2011