Reverse Engineering Process for Extracting Views from Domain Ontology

Soraya Setti Ahmed[1] and Sidi Mohamed Benslimane[2]

[1] Mascara University, Computer Science Department, Algeria
{settisoraya@yahoo.fr}
[2] Djillali Liabes University, Research Laboratory, Computer Science Department
Sidi Bel Abbes, Algeria
benslimane@univ-sba.dz

Abstract. Ontology Modularization is one of the techniques that bear good promises of effective help towards scalability in ontology design, use, and management. The development of proper ontological modules should provide a mechanism for packaging coherent sets of concepts, relationships, axioms, and instances, and a means for reusing these sets in new environments, possibly heterogeneous with respect to the environment the modules were first built. The main contribution of this paper is to describe an approach for extracting views from domain ontology using existential dependency (ED) by reverse engineering process. The extraction process based on ED could provide a coherent fragment of ontology parts together with transitive closure of dependant parts. The goal of reverse engineering process is to output a possible conceptual model, which is more readable to extracting the views, on the basis of the code in which the ontology is implemented. Thus, a set of translation rules is used to convert owl ontology in a UML class diagram.

Keywords: Modularization, Reverse Engineering, Existential Dependency, Ontology Views, Guizzardi Metamodel, UML profiles, OWL.

1 Introduction

Ontology Modularization techniques identify coherent and often reusable regions within an ontology. The ability to identify such modules, thus potentially reducing the size or complexity of an ontology for a given task or set of concepts is increasingly important in the Semantic Web as domain ontologies increase in terms of size, complexity and expressivity[1].

In conceptual modelling, the Foundational Ontology is needed as domain independent theoretical basis to guide and validate models of particular domains, as using of right modelling concepts and rules is making a great influence on the quality of Information Systems [2]. For such purpose, the transformations between conceptual models (expressed, for example, in UML) and ontological models, expressed in ontological languages (for example, OWL) are needed. The extraction process using lightweight ontologies like UML and OWL generates strictly unnecessary classes and individuals, for this reason the first step of our approach is based on the reverse engineering process whose goal is to output a possible conceptual model, which is more readable to extracting the views, on the basis of the code in which the ontology

is implemented [3] and [4]. Thus, a set of translation rules is used to convert owl ontology in a UML class diagram.
The rest of the paper is organized as follows: In section 2 we describe the architecture and the main steps of our approach. Section 3 introduces implementation of our system. Finally, we conclude this paper and outline our future work in section 4.

2 Our approach

In this section, the global architecture of our system is presented. Figure 1 illustrates the main steps of the proposed approach.

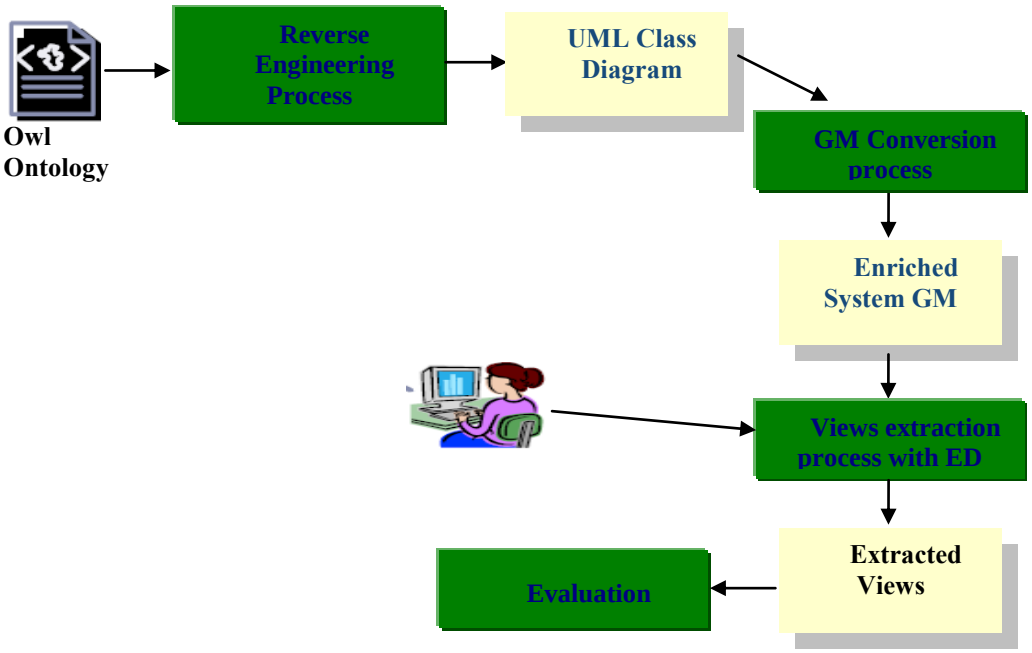

Fig.1. Main steps of our approach

2.1 Reverse engineering process

The designer initiates transformation of domain ontology described in OWL file into UML class diagram by reverse engineering transformation. At first, system transforms ontology classes, then object and data type properties, and finally constraints.

Algorithm of mapping OWL Ontology to UML class diagram

```
Input: OWL file ontology
Output: UML class diagram
Begin
For all  OWL class (concept) defined into  ontology do
Create  UML class with same  name.
      If the ontological  class  is sub class of restriction then
For all restriction do
          If  type of  this  restriction is : cardinality, minCardinality or maxCardinality then
        transforme these  in   multiplicities for  propriety specified on Property of  restriction
               Else
Define the name of role of toClasse classe with object property name specified in    onProperty.
         Endif
 Endfor
      Endif
       If this   class is sub class of other class then
                                  Define   UML generalisation element

       Endif
 Endfor
 For all DataTypeProperty Do
          Create an attribute whose domain is class and whose range is the type of property
  Endfor
  For all  ObjectProperty Do
               Create UML association whose domain is class and whose range is class
               Endfor
End
```

Table 1 summarised the important rules of mapping Owl2Uml

OWL constructer	UML constructer
DatatypeProperty	Property ownedAttribute
ObjectProperty	Property memberEnd
InverseOf	Binary Association
subClassOf, subPropertyOf	superClass,Genearlization
Cardinality, MinCardinality, MaxCardinality	Multiplicities
Ontology	Package Ontology
Union, Intersection	Generalization isDisjoint, isCovering
one of	Enumeration
Individual	Instance

Table 1. Rules of mapping Owl2Uml

2.2 GM conversion process

A Conversion tool implements a transformation from UML class diagram obtained in step 2 to Guizzardi Metamodel (GM) [5]. We introduce a formal ontology, the GM to resolve some highlighted anomalies. We adopt GM to enrich our diagram with several existential dependencies and to define some extraction rules under tree main structural relationships in GM such as association, subtype and part whole.
Guizzardi's concepts *kind, subkind, phase, role* and *relator* are all represented as stereotypes of the UML metaclass *Class*, for example, and all inherit the semantics of Class in UML. Any UML metaclass can be stereotyped.

Some examples of transformation rules:

Rule1: In UML Class Diagram, a collection of instances of classes are, respectively, instances of UML G-M profiles including concrete classes (<<kind>>, <<subkind>>, <<quantity>>, <<collective>>, <<phase>> and <<role>>).

Rule 2: In UML Class Diagram, concrete classes (and their instances) are related via UML G-M profiles including properties (<<mediation>>, <<derivation>>, <<characterisation>>, <<material>> and <<formal>>) as well as complex objects or part-whole (*subQuantityOf, subCollectionOf, memberOf, componentOf*).

Rule3: In UML CD, concrete classes (and instances) can be categorised accordingly by UML G-M profiles via abstract classes (<<category>>, <<roleMixin>> and <<mixin>>) and other rules.

2.3 Views extraction process with ED

This step present extraction cases and rules for how these views can be extracted using existential dependency, especially where the ontology is constructed using the GM formal ontology. We note that user in this case should specify certain individuals and classes. The extraction process produces a more focused and smaller portion and reduces the costs to the user. There are several systems under the 3G-M like systems of (kind, phase, role, mixin, quality, formal, relator, material, mode, Q-parthood, C-partood, M-parthood and system of CF-parthood). All these systems contribute to the ED.

Some examples of extraction cases and rules:

System of Kind: Super kind is Mandatory (+M), subkind is mandatory (+M), siblings are optional (-M): This case applies general rules "requires all superclasses' and 'siblings optional".

System of Relator: A relator is mandatory (+M) and mediated classes are mandatory (+M)
A mediated class is mandatory (+M), a relator is mandatory (+M) and a pair of mediated classes is mandatory (+M): Every instance of mediated class does not make sense without every instance of another (pair) mediated class witch the relator mediates to.

System of Role:
Superkind is an ultimate substance sortal that supplies a principle of identity. Superkind does not make sense without the roles and vice versa. Supermixin (role mixin) is **optional** (-M) since it does not supply a principle of identity.
An application **may not** (-M) need sibling roles since they carry an incompatible principle of identity supplied by its superkind respectively. An individual must be not a member of its siblings. This case applies general rules: ***"some superclasses optional"*** and ***"siblings optional".***
Superkind is mandatory (+M), role is mandatory (+M), supermixin is optional (-M) and sibling roles are optional (-M).

2.4 Evaluation

Correctness of the extracted views translates the fact that no information is lost in the process.
Information preservation may be defined as the fact that the result of a query addressed to the collection is functionally (i.e., not from a performance viewpoint) the same as the result of the same query addressed to the original ontology.

3 Implementation

The architecture of our system has been conceived to follow a Model-Driven Approach. In particular, we have adopted the OMG MOF (Meta-Object Facility) metamodeling architecture [6]. In order to describe constraints in UML/MOF (meta) models, the OMG also proposes the declarative formal language OCL (Object Constraint Language) [7]. On the formalization of the UML profile we have used OCL expressions mainly to: define how derived attributes/associations get their values; define default values of attributes/associations, *i.e.*, define their initial values; specify query operations and specify invariants, *i.e.*, integrity constraints that determine a condition that must be true in all consistent system states.

The full set of OCL expressions including: OCL expressions to specify derivation rules; OCL expressions to define default values; OCL expressions to specify operations created to support some OCL derivation rules and invariants, and invariants to model the constraints stated on the UML profile. An example of an OCL invariant representing the essential parthood axioms is shown in the code below. One can notice that in this expression the modal existential dependence constraint of essential parthood from UFO (Unified Foundational Ontology) is emulated via the existence condition (lower cardinality ≥ 1) plus the immutability constraint (isReadOnly = true).

```
Inv: if (self.isEssential = true) then self.target-> forAll(x | if x.oclIsKindOf(Property)
then ((x.oclAsType(Property).isReadOnly = true) and ((x.oclAsType(Property).lower
>= 1)) else false endif) else true endif
```

4 Conclusion and future work

In This paper we describe our approach for extracting views from domain ontology by reverse engineering process witch consists of transforming the OWL file ontology of E-Tourism into UML class diagram. there is an implementation of the metamodel proposed by Guizzardi [8] by using MDA (Model-Driven Architecture) technologies, in particular, the OMG MOF (Meta-Object Facility) and OCL (Object Constraint Language).
Future work will concern the implementation of process of extracting views with rules proposed here to confirm the useful of our approach.

References

1. Doran,P.,Tamma, V.,Payne,T,R Pal misano,I . : An entropy inspired measure for evaluating ontology modularization. in :5th International conference on knowledge capture(KCAP'09).(2009)
2. Rajugan,R.,Tharan,S.,T.S.Dillon.: Modeling views in the layered view model for XML using UML, journal of Web information System 2 (2006) 95-117.
3. Chikofsky,E.J.,Cross II, J. H., 1990 Reverse engineering and design recovery: a taxonomy. Software Magazine 7 (1990) 13-17.
4. Fernandez-Lopez,M.,Gomez Pérez,A.: Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, Vol. 17:2, 129– 156. © 2002, Cambridge University Press
5. Guizzardi, G., "On Ontology, ontologies Conceptualizations, Modeling Languages, and (Meta)Models", Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV, Olegas Vasilecas, Johan Edler, Albertas Caplinskas (Editors), ISBN 978-1-58603-640-8, IOS Press, Amsterdam, (2007).
6. Object Management Group (OMG):Meta Object Facility MOF core Specification, v2.0,Doc # ptc/06-01-01 (2006)
7. Object Management Group (OMG): Object Constraint Language, v2.0, Doc.# ptc/06-05-01 (2006)
8. Guizzardi,G.: Ontological Foundations for Structural Conceptual Models, Ph.D. Thesis, University of Twente, The Netherlands (2005)