Rules-Based Approach To Convert Class Diagram Operations To Ontology

Souiou Wafa
LISCO Laboratory, Department of Computer Science
Badji-Mokhtar University
PB. 12, 23000, Annaba, ALGERIA
souiou@yahoo.fr

Bounour Nora
LISCO Laboratory, Department of Computer Science
Badji-Mokhtar University
PB. 12, 23000, Annaba, ALGERIA
nora_bounour@yahoo.fr

***Abstract*— In this paper, we propose a new approach that converts the class diagram operations to OWL ontology in order to find the properties and the relationships between operations, which were not previously apparent. This approach is based on the detailed description of the class diagram as an XMI file and the proposition of transformation rules to automatically generate OWL ontology from UML class diagram. We demonstrate the applicability of our approach with an example.**

Keywords—class diagram, UML, OWL ontology, transformation

I. INTRODUCTION

The use of ontologies is important for many fast-growing technologies. They allow a greater variety of structural and non-structural relationships to concretely represent existing concepts in a domain and the different relationships between them. In the other side, UML is a modeling language adopted by a large community in the conceptual modeling framework, and has therefore benefited from a wealth of experience and a wide range of dedicated tools [1] .

In this article, we propose an approach that extracts ontology from the UML class diagrams. The resulting ontology will help us to find the properties and the relationships between operations which were not previously apparent [1], aiming in the near future to identify services by grouping operations representing a highly coherent set and that are generally of the same nature and the same semantic level . This approach is based on the detailed description of the class diagram as an XMI file, which will be the input of our system.

The rest of this article is organized as follows: In Section II, we present some related works. In Section III, we describe our approach that transforms the class diagram operations to OWL ontology. In Section IV, we illustrate our approach through an example. In Section V, we present results and discussion. Finally, a conclusion and perspectives for further research are presented in section VI.

II. RELATED WORKS

There are many approaches in the literature on how to transform the UML class diagrams to OWL ontology. In [1] the authors propose an approach based on a Meta model for UML class diagrams and a graphical grammar composed of several rules enabling them to transform all that is modeled in their generated environment AToM3 to an OWL ontology stored in a file on the hard drive. The graphical grammar is based on transformation rules; these rules are designed to transform the class diagram in the level of implementation, always to obtain at the end a usable description of ontology. The focus of this work was to Take advantage of the visual expressivity of the notation language UML and the power of ontologies so that the information described by those diagrams can be shared and linked with other information and they could start dealing with the overlaps, gaps, and integration barriers between modeling languages and get greater value out of the information capture. In [2] the authors propose a UML conversion to OWL ontology, the solution is based on the ontology UML profile (OUP) and acts as an extension to the standard UML tools. The objective of this work was to create complete OWL ontologies without needing to use ontology-specialized development tools. In [3], the authors propose an UML transformation to DAML (DARPA Agent Markup Language) aiming to identify the similarities and differences between the two languages and discuss how they can be mapped the one to the other. In [4], the author presented an implementation of the ODM (Ontology Definition Meta model) using the ATL language (Atlas Transformation Language). The focus of this work was to provide a solution for bridging UML or MOF (Meta Object Facility) based modeling tools (e.g. MDA Tools) and tools for the development of the semantic Web and ontology. In [5], the authors propose a method that converts UML class diagrams to an OWL ontology using the OWL / XML language maintaining the semantic characteristics of the diagrams. In [6], the authors propose a detailed comparison between UML and OWL. Authors have aimed in this work to clarify the relationship between the two technologies and provide a solid basis for deciding how to use them together or integrate them. In [7], the authors discuss approaches that focus on the transformation domain models between UML and the languages of the semantic web aiming to identify the different ways in which they treat the conceptual differences between these languages. In [8], the authors present an automatic transformation of ontology OWL 2 from the ISO 19103 data model UML profile class diagrams. The approach of the proposed conversion is based on the UML profile class diagram transformation to ontology and emphasizes the automatic transformation of UML geospatial profile class diagrams to OWL ontology using XMI as intermediate format for UML model. The focus of this work was to identify differences between languages and the incompatible language features.

In this paper, we propose another vision different from that addressed in previous works. This vision is to propose the transformation of UML class diagram to OWL Ontology taking into account the transformation of operations. This transformation will help us later to identify services of our information system and orient it towards a service-oriented architecture.

III. PROPOSED APPROACH

After presenting in the state of the art some work on the transformation of UML class diagram into OWL ontology, we present in this section rules-based approach which converts class diagram operations to OWL ontology. Our approach is based on transformation rules; those rules try to transform the class diagram, always in order to obtain at the end a usable description of ontology. We will use as a tool the ArgoUML which will help us automatically generate the detailed description of the class diagram as an XMI file, the latter will be the input of our system and the resulting ontology is expressed in OWL / XML. Our choice is quickly related to ArgoUML because of the advantages which it presents like its simplicity, and its accessibility [9].

The transformation proceeds in several steps “fig. 1”:
1) Graphic description of class diagram in ArgoUML.
2) Generation of the detailed description of the class diagram as an XMI file.
3) Import the XMI file to the application interface.
4) Apply the transformation rules on the XMI file and an OWL file is generated automatically which contains the OWL ontology.
5) Visualization and use of OWL ontology by using Protégé.

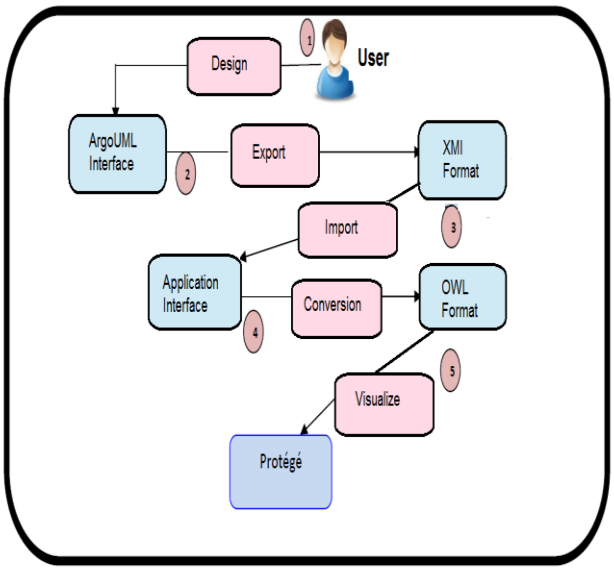


Fig. 1. Transformation sequence

Our approach allows modeling with ontologies the properties of operation [10] such as : Visibility, Type, Properties, Implementation, DirectionArgument and TypeArgument a view in the near future to also model the operation body using ontologies.

A. *Transformation Rules*

Our approach is realized depending on proposed transformation rules “Table. 1”. We suggest a set of rules especially for operations properties and arguments of a class diagram while preserving the semantics of some specific characteristics of the class diagram such as classes, attributes, associations and cardinalities.

TABLE I. OPERATION PROPERTIES TRANSFORMATION

Visibility
This property can take the following values: public, private, protected,package. <owl:DatatypeProperty rdf:ID="OperationVisibility"> <rdfs:domain rdf:resource="#nameOperation"/> <rdfs:range > <owl:oneOf rdf:parseType="Collection"> <OperationVisibility rdf:about="#Public" /> <OperationVisibility rdf:about="#Private" /> <OperationVisibility rdf:about="#Protected" /> <OperationVisibility rdf:about="#Package" /> </rdfs:range > </owl: DatatypeProperty >
Type
The types can be primitive type, or type that is defined in the model.
Property
This property can take the following values: is Query, is Abstract, is Static, is Leaf and Concurrency. </owl:DatatypeProperty rdf ID="OperationProperty"> <rdfs:domain rdf:resource="#nameOperation"/> <rdfs:range > <owl:oneOf rdf:parseType="Collection"/> < OperationProperty rdf:about="#isQuery" /> < OperationProperty rdf:about="#isAbstract" /> < OperationProperty rdf:about="#isLeaf" /> < OperationProperty rdf:about="#isStatic" /> < OperationProperty rdf:about="#Concurrency"/> </rdfs:range > </owl: DatatypeProperty >
Implementation
This property can take the following values: Postconditions, Preconditions, and Body Conditions. <owl:DatatypeProperty rdf:ID="OperationImplementation"> <rdfs:domain rdf:resource="#nameOperation"/> <rdfs:range > <owl:oneOf rdf:parseType="Collection"> <OperationImplementation rdf:about="#Preconditions" /> < OperationImplementation rdf:about="#Postconditions" /> < OperationImplementation rdf:about="#BodyConditions" /> </rdfs:range > </owl: DatatypeProperty >
Direction Argument
This property can take three directions: in, out, inOut. < owl:DatatypeProperty rdf:ID="DirectionArgument"> <rdfs:domain rdf:resource="#Argument"/> <rdfs:range > <owl:oneOf rdf:parseType="Collection"> <DirectionArgument rdf:about="#in" /> < DirectionArgument rdf:about="#out" /> < DirectionArgument rdf:about="#inOut" /> </rdfs:range > </owl: DatatypeProperty >
TypeArgument
This property can have a primitive type or a type defined in the model.

The following lines of code present the operations conversion algorithm:

```
//** Creation of classes
Create Class “NameClass”
Create Class “Operation” sub-class of “NameClass”
```

```
 Create Class “NameOperation” sub-class of “Operation”
 Create Class “Argument” sub-class of “NameOperation”
 //** Creation of DatatypeProperties
 Create DatatypeProperty “OperationVisibility”
Create DatatypeProperty "private","protected","public", "package"
sub-property of "OperationVisibility"
 Create DatatypeProperty “typeOperation”
 Create DatatypeProperty “OperationProperty”
Create DatatypeProperty "isQuery",
"isAbstract", "isLeaf", "isStatic",
"concurrency" sub-property of
"OperationProperty"
Create DatatypeProperty "sequential",
"garded", "concurrent",
sub-property of "concurrency"
Create DatatypeProperty
“OperationImplementation”
Create DatatypeProperty
"Preconditions","Postconditions",
"BodyConditions "sub-property of
"OperationImplementation"
Create DatatypeProperty “DirectionArgument”
Create DatatypeProperty "in", "out", "inOut"
sub-property of "DirectionArgument"
Create DatatypeProperty “TypeArgument”
//** Creating instances
FOR EACH Class do
Create Class “NameClass”
FOR EACH Operation do
Create Class “NameOperation” Sub-ClassOf “Operation”
Create DatatypeProperty “hasVisibility”
With domaine “NameOperation”
With range “GetValue (OperationVisibility)”
Create DatatypeProperty “hasType”
With domaine “NameOperation”
With range “GetType (TypeOperation)”
Create DatatypeProperty “hasProperty”
With domaine “NameOperation”
With range “GetValue (OperationProperty)”
Create DatatypeProperty “hasImplementation”
With domaine “NameOperation”
With range “GetValue (OperationImplementation)”
FOR EACH Argument A do
Create DatatypeProperty “hasDirection”
With domaine “Argument”
With range “GetValue (DirectionArgument)”
Create DatatypeProperty “hasType”
With domaine “Argument”
With range “GetType (TypeArgument)”.
```

IV. CASE STUDY:

We consider the following case study:

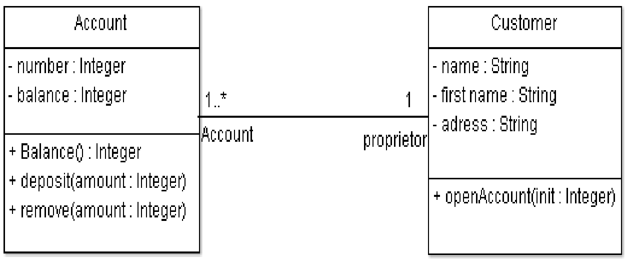
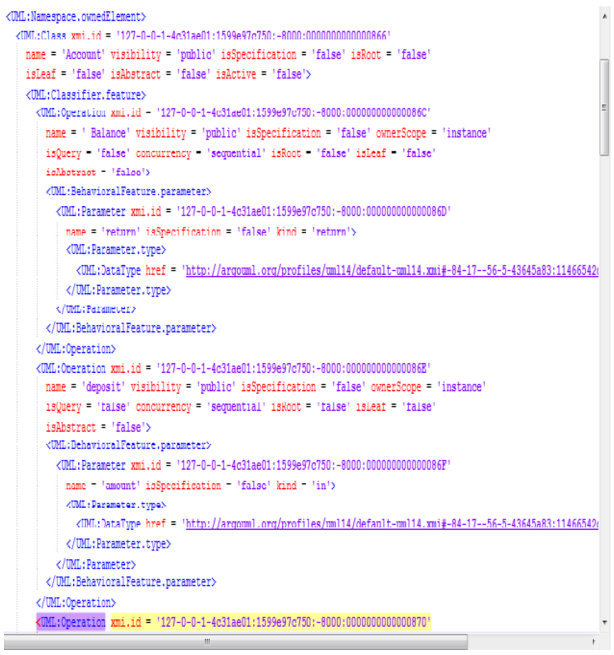


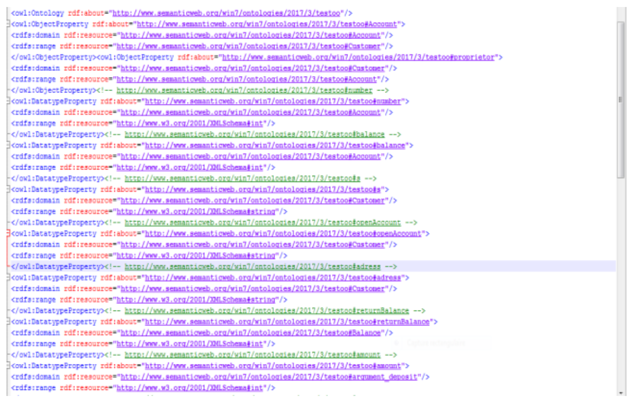
Fig. 2. Example of a class diagram [11]

The generated XMI file “Fig.3” is the input of our system which will analyze it using the previous algorithm and generating an output OWL file contains the resulting ontology of the conversion “Fig.4”.



```
<UML:Namespace.ownedElement>
  <UML:Class xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:0000000000000866'
   name = 'Account' visibility = 'public' isSpecification = 'false' isRoot = 'false'
   isLeaf = 'false' isAbstract = 'false' isActive = 'false'>
   <UML:Classifier.feature>
    <UML:Operation xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:000000000000086C'
     name = ' Balance' visibility = 'public' isSpecification = 'false' ownerScope = 'instance'
     isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false'
     isAbstract = 'false'>
     <UML:BehavioralFeature.parameter>
      <UML:Parameter xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:000000000000086D'
       name = 'return' isSpecification = 'false' kind = 'return'>
       <UML:Parameter.type>
        <UML:DataType href = 'http://argouml.org/profiles/uml14/default-uml14.xmi#-84-17--56-5-43645a83:11466542
       </UML:Parameter.type>
      </UML:Parameter>
     </UML:BehavioralFeature.parameter>
    </UML:Operation>
    <UML:Operation xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:000000000000086E'
     name = 'deposit' visibility = 'public' isSpecification = 'false' ownerScope = 'instance'
     isQuery = 'false' concurrency = 'sequential' isRoot = 'false' isLeaf = 'false'
     isAbstract = 'false'>
     <UML:BehavioralFeature.parameter>
      <UML:Parameter xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:000000000000086F'
       name = 'amount' isSpecification = 'false' kind = 'in'>
       <UML:Parameter.type>
        <UML:DataType href = 'http://argouml.org/profiles/uml14/default-uml14.xmi#-84-17--56-5-43645a83:11466542
       </UML:Parameter.type>
      </UML:Parameter>
     </UML:BehavioralFeature.parameter>
    </UML:Operation>
    <UML:Operation xmi.id = '127-0-0-1-4c31ae01:1599e97c750:-8000:0000000000000870'
```

Fig. 3. The generated XMI file of the class diagram



```
<owl:Ontology rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo"/>
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Account">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Account"/>
<rdfs:range rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Customer"/>
</owl:ObjectProperty><owl:ObjectProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#proprietor">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Customer"/>
<rdfs:range rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Account"/>
</owl:ObjectProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#number -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#number">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Account"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#balance -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#balance">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Account"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#s -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#s">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Customer"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#openAccount -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#openAccount">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Customer"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#adress -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#adress">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Customer"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#returnBalance -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#returnBalance">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#Balance"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty><!-- http://www.semanticweb.org/win7/ontologies/2017/3/testoo#amount -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#amount">
<rdfs:domain rdf:resource="http://www.semanticweb.org/win7/ontologies/2017/3/testoo#argument_deposit"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

Fig. 4. The generated OWL file of the conversion

In order to test the semantic coherence of our resulting ontology, we loaded it under Protegé, and using the OntoGraf plugin we got the following form “Fig.5”.

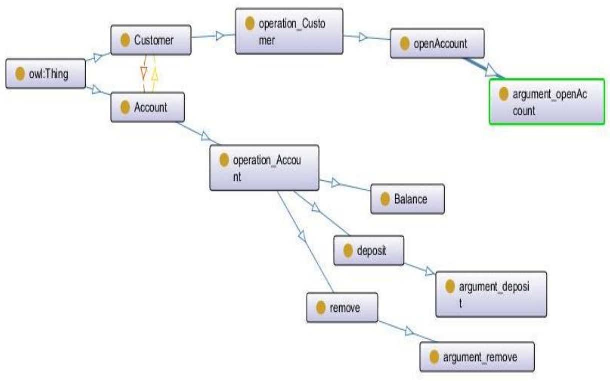
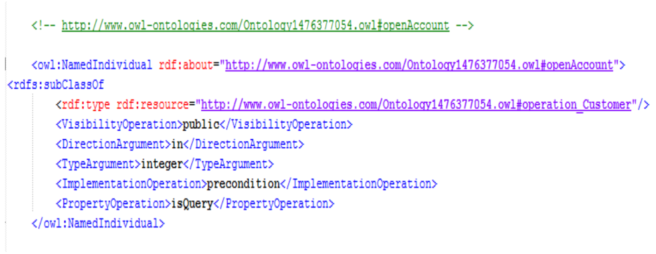


Fig. 5. The OntoGraf diagram of the resulting ontology

V. RESULTS AND DISCUSSION

Despite the differences between OWL ontology language and UML, automatic transformation is feasible, and gives satisfactory results.

To show the utility of our approach, we take as example the operation "openAccount". The following lines of code illustrates the values of the data type properties described previously which proves that the resulting ontology allows us to extract and find properties of operations which were not previously apparent in the class diagram :



```
    <!-- http://www.owl-ontologies.com/Ontology1476377054.owl#openAccount -->

    <owl:NamedIndividual rdf:about="http://www.owl-ontologies.com/Ontology1476377054.owl#openAccount">
<rdfs:subClassOf
        <rdf:type rdf:resource="http://www.owl-ontologies.com/Ontology1476377054.owl#operation_Customer"/>
        <VisibilityOperation>public</VisibilityOperation>
        <DirectionArgument>in</DirectionArgument>
        <TypeArgument>integer</TypeArgument>
        <ImplementationOperation>precondition</ImplementationOperation>
        <PropertyOperation>isQuery</PropertyOperation>
    </owl:NamedIndividual>
```

The main difference between the approach presented in this paper and similar work is the transformation of class diagram operations to ontology. This transformation takes into account the different properties of operation as previously mentioned.

VI. CONCLUSION

Currently ontologies represent a promising solution for the representation and knowledge sharing. It is in this context that the scope of our work lies in order to integrate a new vision of formal representation and knowledge sharing.

In this paper, we proposed rules-based approach which extracts ontology from UML class diagram. The resulting ontology will help us to find the properties and the relationships between operations which were not previously apparent. The input of our system is an XMl file which contains the detailed description of the class diagram and we obtain as result an OWL file of the resulting ontology.

Enjoying the benefits of ontologies, aiming in the near future to:

– Enrich the resulting ontology from other UML diagram.

– Grouping operations representing a highly coherent set and that are generally of the same nature and the same semantic level in order to identify services.

REFERENCES

[1] A. Belghiat and M. Bourahla, “Automatic Generation of OWL Ontologies from UML Class Diagrams Based on Meta-Modelling and Graph Grammars,” *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 6, no. 8, pp. 967–972, 2012.

[2] D. Gasevic, D. Djuric, V. Devedzic, and V. Damjanovi, “Converting UML to OWL ontologies,” in *Proceedings of the 13th international World Wide Web - Alternate track papers & posters - WWW Alt. ’04*, 2004, pp. 488–489.

[3] K. Baclawski, M. K. Kokar, P. A. Kogut, L. Hart, J. Smith, W. S. H. Iii, J. Letkowski, and M. L. Aronson, “Extending UML to Support Ontology Engineering for the Semantic Web,” in *International Conference on the Unified Modeling Language*, 2001, pp. 342–360.

[4] G. Hillairet, “ATL Use Case - ODM Implementation (Bridging UML and OWL).,” 2007. [Online]. Available: http://www.eclipse.org/m2m/atl/usecases/ODMImplementation/, 2007.

[5] M. Bahaj and J. Bakkas, “Automatic Conversion Method of Class Diagrams to Ontologies Maintaining Their Semantic Features,” *Int. J. Soft Comput. Eng.*, vol. 2, no. 6, pp. 65–69, 2013.

[6] C. Atkinson, “A Detailed Comparison of UML and OWL,” pp. 1–58, 2008.

[7] K. Falkovych, M. Sabou, and H. Stuckenschmidt, “UML for the Semantic Web: Transformation-Based Approaches,” 2003.

[8] I. Zarembo, “Automatic Transformation of UML Geospatial Profile to OWL Ontologies,” in *QUAESTI - Virtual Multidisciplinary Conference*, 2013, pp. 225–230.

[9] A. Ramirez and L. Tolke, “ArgoUML User Manual A tutorial and reference description.”

[10] M. Betbeder, “Diagrammes de classes,” pp. 1–16, 2003.

[11] D. Longuet, “UML Cours 3 Diagrammes de classes,” 2017.