

Robust Representation for Conversion UML Class into XML Document using DOM

Noredidine GHERABI
Hassan 1 University, FSTS
Department of Mathematics
and Computer Science

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

ABSTRACT

This paper presents a Framework for converting a class diagram into an XML structure and shows how to use Web files for the design of data warehouses based on the classification UML. Extensible Markup Language (XML) has become a standard for representing data over the Internet. We use XSD schema for define the structure of XML documents and validate XML documents.

A prototype has been developed, which migrates successfully UML Class into XML document based on the formulation mathematics model. The experimental results were very encouraging, demonstrating that the proposed approach is feasible efficient and correct.

General Terms

ORDB, Modeling

Keywords

UML, XML, XML schema, DOM.

1. INTRODUCTION

XML was originally envisaged as a language for defining new document formats for the Web and can be considered a meta-language: a language for defining markup languages. XML is a text-based format that provide mechanisms for describing document structures using markup

With the current revolution in the use of the Web as a platform for application development, XML (eXtensible Markup Language) [1] was the first interest to many e-business applications.

This document aims to define a correspondence between the class diagrams of Unified Modeling Language (UML) [2] and XML Schema using the mathematical representation of the class diagram and technology DOM to build the XML structure.

A number of approaches to convert the XML Schema in UML diagram or vice versa have been described in other works.

Since both the technologies UML and XML are just beginning to grow, not much work has been done in the domain of mapping between these two techniques. Grady Booch et al[3], describes a graphical notation in UML for designing XML Schemas, in this paper the authors describe the relationship between UML and the SOX schema (a forerunner to XML Schema) used by CommerceOne.

Until recently, there has been no effective means to design a XML schema from the object-oriented concepts without exposing designers to problems of low-level implementation.

Bird, Goodchild and Halpin[4] have proposed a method that uses a conceptual language of "role object model" to generate XML schemas.

Wu and Hsieh [5] 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.

Mikael R. Jensen et al[6], present an algorithm for conversion but this time in the opposite direction of our approach, they have developed algorithms for automatically constructing UML diagrams from XML schema.

In another way, Abdelsalam Maatuk et al[7] propose an approach for migrating existing Relational Databases (RDBs) into Object-Relational Databases (ORDBs). Transforming conceptual models (e.g., EER, UML class diagrams) into ORDB has been studied over the past years [8, 9, 10, 11 and 12] but is not yet fully satisfactory solution provider.

J. Fong et al [13] proposes a method for Converting relational database into XML data with DOM. Ref. [14] presents techniques for converting relational databases into object oriented databases. The data conversion involves unloading relations into sequential files, and reloading them into object-oriented databases with constraints preservation.

We believe that this article will be among the best papers to present algorithms that automatically extract XML data directly from UML Class, retaining important semantic information.

2. PROPOSED METHOD

The basic steps of our approach are shown in Figure 1

The first step is to present the class diagram in the form of mathematical formulation, and then codify it in a well-defined structure this technique is detailed in section 2.1. The second step, the class diagram codified will be validated using a validation algorithm (Sections 3.1 and 3.2). After, in step 3 the class diagram validated will be imported into the system with the XSD schema. In Step 4 the system generates an XML and defines the structure of the class diagram using DOM (Section 3.3). And finally this XML file is validated and stored.

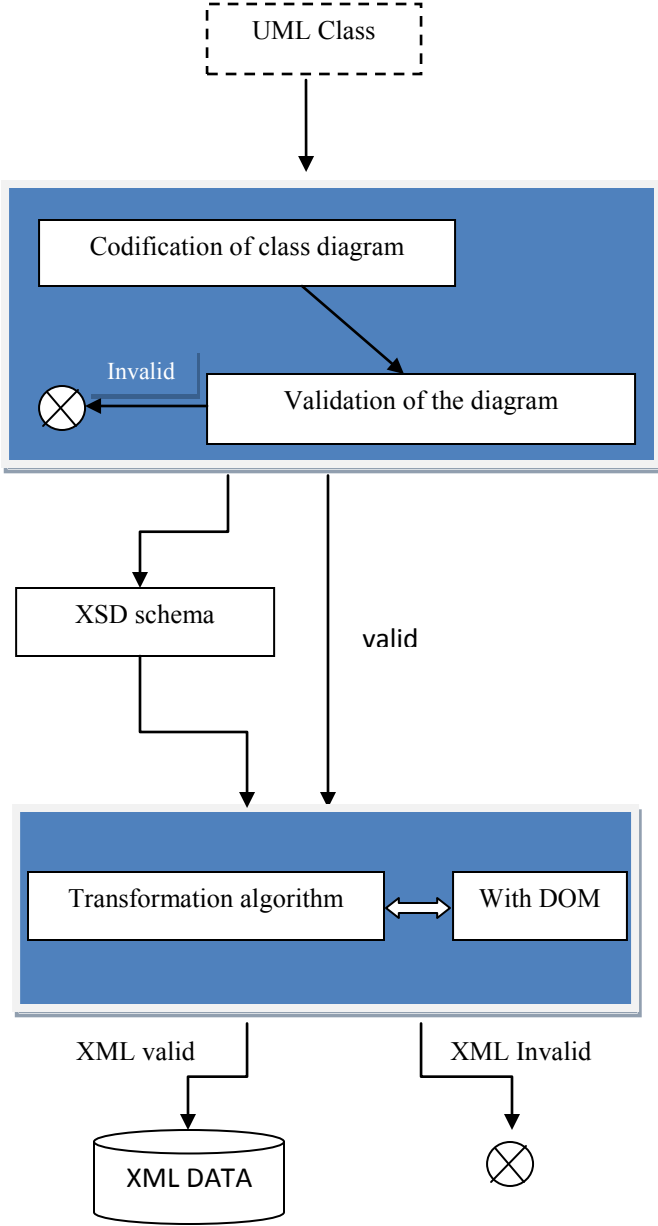


Fig.1: Schema of our proposed approach.

2.1 Class UML and XML Models

In our approach a class diagram in UML is represented as a set of classes, is denoted as 4-tuple where the first element is the name of the UML class, the second element is a list of attributes, the third element is the set of methods and the latest element is the relationships:

$$Class := \{C | C := (C_N, C_A, C_M, C_R)\}$$

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_M : is a set of methods for defined class C

$$C_M := \{M | M := (M_n, M_t, M_v)\}$$

Where M_n is the name of the method M , M_t its type and M_v is the visibility of this method.

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

$$C_R := \{R | R := (R_t, R_c, R_r)\}$$

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

We now define the sets to describe the XML structure diagram.

XML – Classes = (Class – name, Attributes, Methods, Relations) / Class - name $\in C_N$, Attributes $\in C_A$, Methods $\in C_M$, Relations $\in C_R$

The set XML-CLASS describes the structure of the complete class diagram. It contains 4-tuple where the first element of the set is the name of the UML class. The other elements are three sets, where Attributes is a list of attributes associated with this particular class, this attribute belongs to the entire C_A and defined as:

$$Attribute = \{(Attr - name, Attr - type, Attr - visibility, Attr - default) / Attr - name \in A_N, Attr - type \in A_t, Attr - visibility \in A_v, Attr - default \in A_d\}$$

The third element of the set XML-Classes describes the set of Methods, is a list of all the methods that define the class, and defined as:

$$Method = \{(Method - name, Method - type, Method - visibility) / Method - name \in M_N, Method - type \in M_t, Method - visibility \in M_v\}$$

And the last element is a list of all existing relationships with this class

$$Relation = \{(Rel - type, Rel - cardinality, Rel - Class Relation) / Rel - type \in R_t, Rel - cardinality \in R_c, Rel - Class Relation \in R_r\}$$

Where:

- *Attr-Type or Method-type* = { Numeric, String, Text, NULL... }
- *Attr-visibility or Method-visibility* = { Public, Private, Protected }
- *Rel-type* = { AGGREGATION, COMPOSITION, ASSOCIATION, GENERALIZATION }
- *Rel-cardinality* = { 0..*, 1..*, 0..1, 1 }

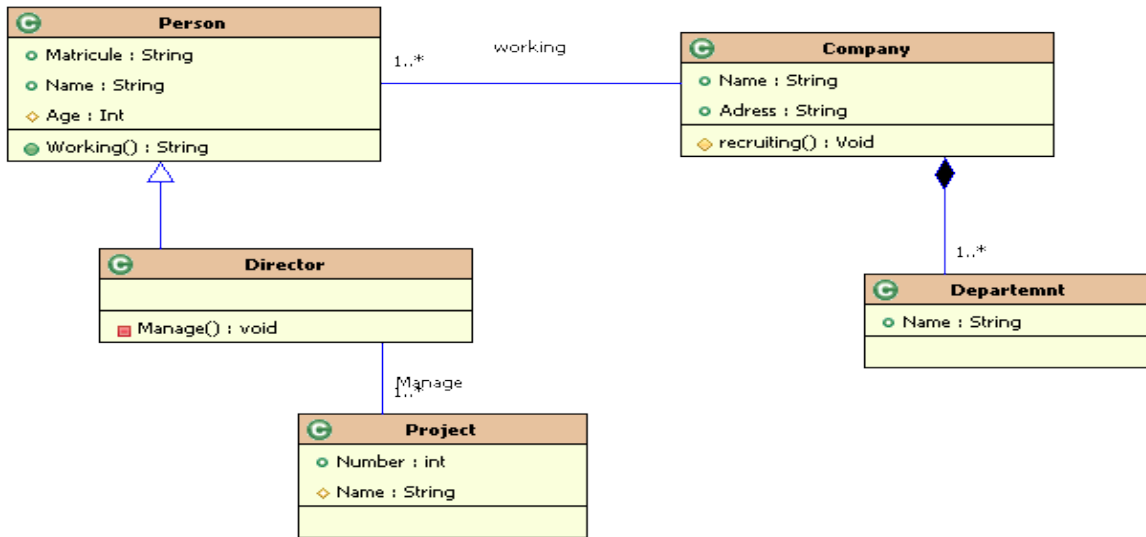


Fig 2: Example of the class diagram

Rel-cardinality: is the set of cardinalities used to describe the quantitative relationship between elements in the UML data model. The cardinalities of a relationship are given by specifying minimum and maximum cardinalities

3. OUR ALGORITHM TO TRANSFORM THE UML DIAGRAM INTO XML DOCUMENT

The algorithm to generate the XML file from the UML class is divided into three steps. The first step, **ReadUML**, is a sub-function to codify the structure of the class diagram in a text file and then import it into the system. The second part, we verify the structure of the diagram and then validate it, and the third step of the algorithm, called the sub-function **GenerateXML**, is to generate all elements of the diagram and create a valid XML

file. Finally, the function **ValidateXMLDocument** is used to validate the generated XML file

In general the structure of the XML generated by the algorithm corresponds to the UML class where a tag XML is generated for each Class, each attribute, each Method and each relationship. All elements having a parent/ child relationship in the XSD schema are connected in the UML diagram.

The general algorithm to convert the UML diagram in XML format is:

```

Algorithm GenerateXMLClass
ReadUML()
ValidateClassDiagram()
GenerateXML()
ValidateXMLDocument()
END.
    
```

3.1 Codification of the diagram

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

C_N ; **Number-of-Attributes** ; {C_A} / (A_n:A_t:A_v:A_d); ; **Number-of-Methods** ; {C_M} / (M_n:M_t:M_v); ; **Number-of-Relationships** ; {C_R} / (R_t:R_c:R_r); ;

For example: Consider the class diagram in Figure 2. Attributes or methods publics, privates and protected are presented respectively by the following symbols: ●, ■, ◆.

In this example the class "Person" is defined by three attributes: "Maticule" is the string 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 "Age" is of type Int, visibility is protected and no default value.

Class "Person" contains a single method "Working" the type of the return value is String and its visibility is Public.

One association relationship that exists with the class "Company" and its cardinality is 1..*

There is no other existing relationship, except the inheritance relationship that will be represented in the encoding of the specialized class "Director" and not in the general class "person". Therefore, the codification of the class "Person" that will be imported into the system is as follows:

Person;3;Matricule:String:Public;;Name:String:Public;;Age:Int:Protected;;1;Working:String:public;1;1..*:Company;0;0;0;

3.2 Validation UML Diagram

In the validation of a class diagram, you must respect the standards of UML 2.0, before moving on to the step of conversion; therefore, there are various techniques to follow for validation.

Our system follows these steps for the validation of class diagram:

- Class names must be unique and the number of attributes of each class must be figured, and if possible the type, the visibility and the default value of each attribute will be presented in the diagram.
- The number of associations must be mentioned, over the association must be provided by two existing classes in the diagram with its cardinality.
- Similarly, the number of relations of composition and aggregation of each class with other classes should also be mentioned and their cardinalities.
- Inheritance relationship that connects two classes or more classes, in a way generalization / specialization, must be presented in the diagram.
- Algorithm verifies all the relations between classes and respects the validation of a UML class diagram, in order to validate the entire diagram which will be imported into the system for conversion.

3.3 Generating XML File

3.3.1 XML Schema Definition

First, we defined the XML Schema Definition (XSD) for each class diagram. We think that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. Here are some reasons:

- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML.
- XML Schemas support data types.

- XML Schemas support namespaces.

This XSD is stored and prepared to use it to validate the generated XML file, and its structure is defined as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

<xsd:element name="Class" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Attribute" minOccurs="0"
maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Attr-Type" type="xsd:string"/>
            <xsd:element name=" Visibility" type="xsd:string"/>
            <xsd:element name="Dvalue" type="xsd:string"/>
          </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Method" minOccurs="0"
maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="Method-type" type="xsd:string"/>
          <xsd:element name="Visibility" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="name-Method" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Relationships" >
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="ASS" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="Cardinality" type="xsd:string"/>
                <xsd:element name="Class-Relation" type="xsd:string"/>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="Aggregation" minOccurs="0"
maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="Cardinality" type="xsd:string"/>
                <xsd:element name="Class-Relation" type="xsd:string" />
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="Composition" minOccurs="0"
maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="Cardinality" type="xsd:string"/>
                <xsd:element name="Class-Relation" type="xsd:string" />
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

```

</xsd:complexType>
</xsd:element>
<xsd:element name="Generalization" minOccurs="0"
maxOccurs="unbounded">
  <xsd:complexType>
  <xsd:sequence>
  <xsd:element name="Class-Relation" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="name-Class" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
</xsd:schema>

```

3.3.2 Algorithm for generation XML Document.

Now we can generate the XML file corresponding to the class diagram, for this we developed an algorithm that constructs the XML structure based on the technique of DOM.

In this algorithm the system through all the classes of the diagram and extract the attributes and methods of each class and the types of relationships between a class and other classes in the diagram.

Algorithm to build the structure of the XML file from a UML diagram:

```

C=first Class
While C!=Null
C=C.Next
Create New Element E

```

```

Create New Attribute Name
E.Name=CN
NbA=Number of attributes of the Class C

For i=1 to NbA do
Create New Attribute Attr(i).Name
Attr(i).Name=CA(i).An
Create New Element A
A(i).type= CA(i).At
A(i).visibility= CA(i).Av
A(i).defaultvalue= CA(i).Ad
End for

```

```

NbM=Number of methods of the Class C
For i=1 to i=NbM do
Create New Attribute Method(i).Name
Method(i).Name=CM(i).Mn
Create New Element M
M(i).type= CM(i).Mt
M(i).visibility= CM(i).Mv
End For

```

```

NbR=Number of Relations of the Class C
Create New Element R
For i=1 to i=NbR do
If Rel(i) is association then
Create New Element ASS
ASS.type= CR(i).Rt
ASS.cardinality= CR(i).Rc
ASS.ClassRelation= CR(i).Rr
Elseif Rel(i) is Composition then
Create New Element Composition
Composition.type= CR(i).Rt
Composition.cardinality= CR(i).Rc

```

```

Composition.ClassRelation= CR(i).Rr
Elseif Rel(i) is Aggregation then
Aggregation.type= CR(i).Rt
Aggregation.cardinality= CR(i).Rc
Aggregation.ClassRelation= CR(i).Rr
Elseif Rel(i) is Generalization then
Generalization.ClassRelation= CR(i).Rr
End If
End For
End While

```

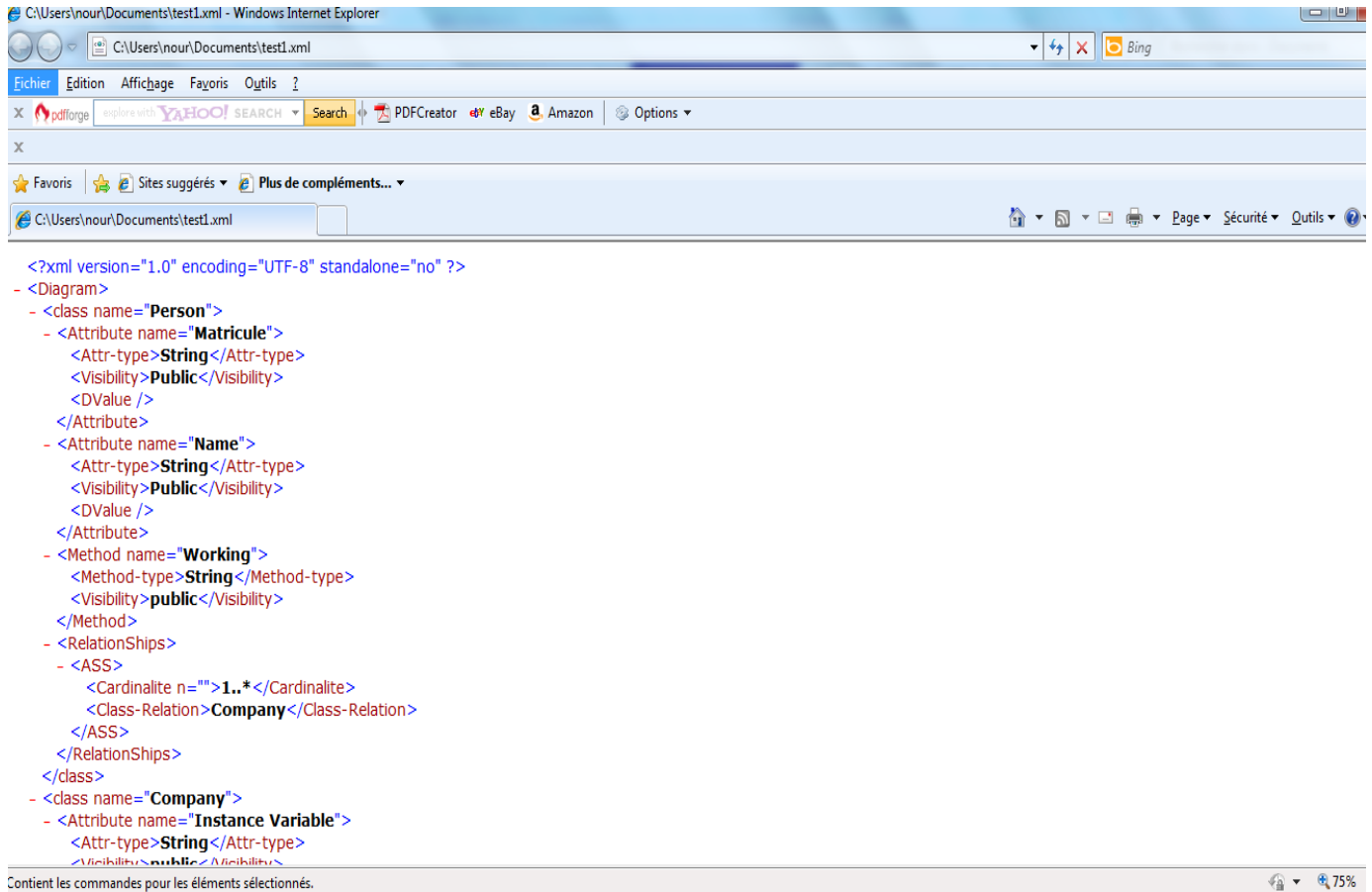


Fig 3: The output XML corresponding to the example of the class diagram illustrated in the Figure 2

4. CASE STUDY

We have developed a tool that performs the mapping of UML to XML using the above specifications. Our platform conversion is developed by the Java language, as a proof of concept for this work (See Fig.4).

The system takes as input an encoded text file that contains the overall structure of the diagram.

Figure 2 shows a section of a UML class diagram developed using MyEclipse. The encoded text file that contains the structure of the diagram is defined as follows:

```

Person;3;Matricule:String:Public;;Name:String:Public;;Age:Int:
Protected;;1;Working:String:public;1;1..*:Company;0;0;0;

Company;2;Name:String:Public;;Adress:String:Public;;1;Recrui
-ting:Void:Protected;1;1..1:Person;0;0;0;

Department;1;Name:String:Public;;0;0;1;1..*:Company;0;0;

Director;0;1;Manage:Void:Private;1;1..1:Project;0;0;1;Person;

Project;2;Number:Int:Public;;Name:String:Protected;;0;1;1..*:
Director;0;0;0;
  
```

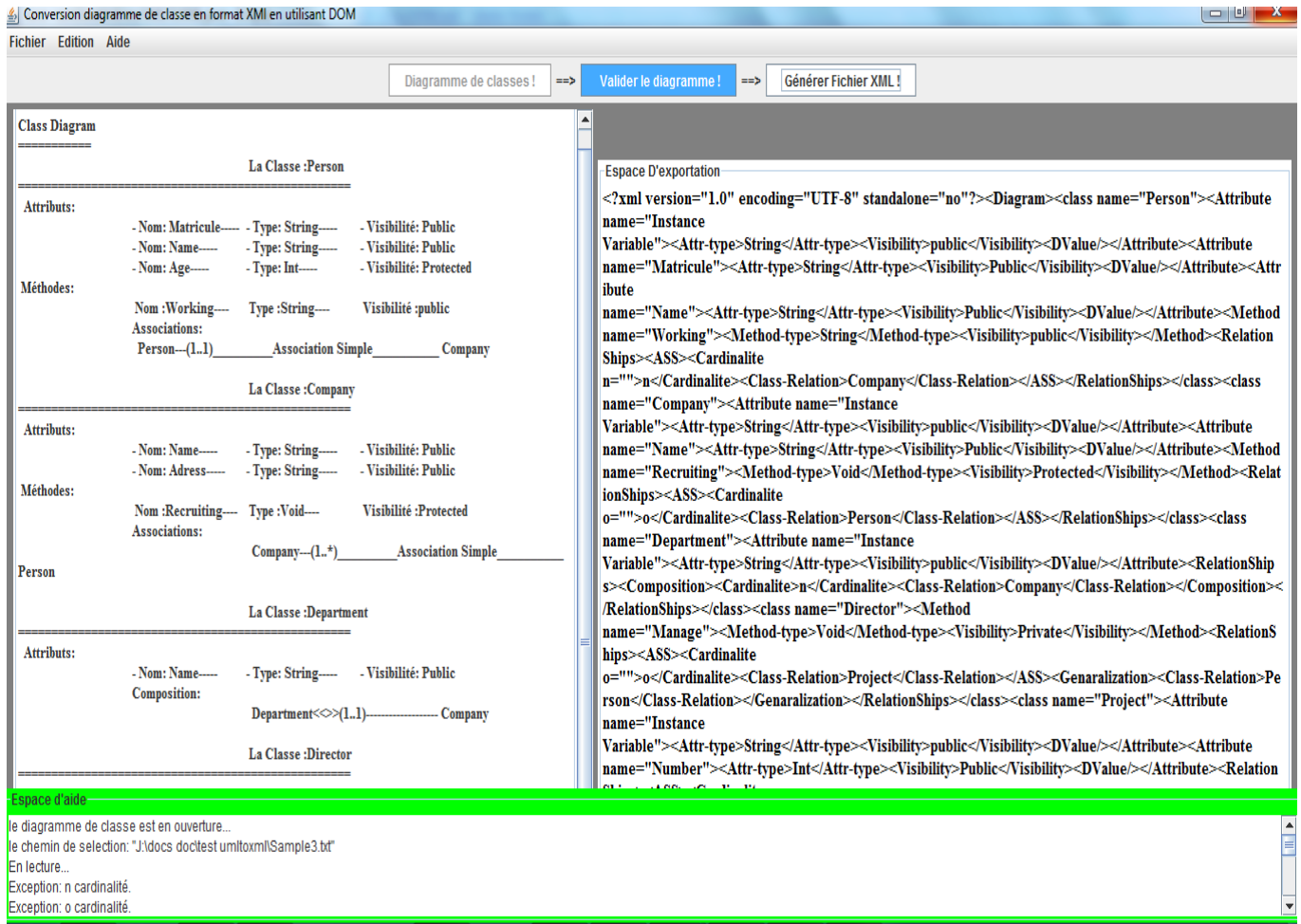


Fig 4: Our tool for the converting of class diagram to XML document

After, the system converts the diagram in XML format based on the series of specifications identified in this document. The input file has a format that we have defined to allow the conversion. Each entity UML is chosen to be represented in a text-based format and then converted to an element or attribute, depending on the symbolic representation of the specifications. Figures 3 and 4 respectively describe our platform conversion and XML data corresponding to the class diagram used in Figure 1 of section 3.

In this paper, we present a survey of transformation techniques that are used to generate an XML structure from UML design. We analyze the existing transformation techniques using all the analysis parameters identified in the survey. The table in the figure 5 shows a study of different aspects to highlight the strengths and weaknesses of different techniques.

5. CONCLUSION

This paper has presented a simple solution for mapping UML object-oriented model into XML document. The framework and the process of mapping approach were discussed in this document. A simple and consistent mathematical formulation has been proposed here. In addition, a mapping tool was developed to facilitate the automatic generation of XML data, and also to validate the input structure UML and the structure of XML output.

We have demonstrated the application of our approach to the transformation of UML class into XML document, and obtained a better performance compared with some previous methods.

	XML Schema (XSD)	Transformation Into XML data	Using DOM	Using DTD	Mapping				
					Attribute	Method	Composition	Aggregation	Generalization
K. Narayanan et al [15]	Yes	No	No	No	Yes	No	Yes	No	Yes
N. Routledge et al [16]	Yes	No	No	No	Not indicated				
I-Chen Wu et al [5]	Yes	No	Yes	No	Yes	Yes	Not indicated		
Grady Booch et al [3]	Yes	No	No	No	Yes	No	Yes	Not indicated	Yes
John Heintz et al [17]	No	No	No	Yes	Yes	No	No	Yes	Yes
OUR	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes

Fig 5: A comparative study with other mapping methods

6. REFERENCES

- [1] W. J. Pardi, XML in Action, Microsoft Press, Washington, 1999.
- [2] M. Fowler and K. Scott, UML Distilled, 2nd Edition, Addison Wesley, Boston, 2000.
- [3] Grady Booch, Magnus Christerson, Mathew Fuchs, Jari Koistinen; "UML for XML Schema Mapping Specification"; Rational Software Corp. and CommerceOne Inc., December 1999.
- [4] BIRD, L., GOODCHILD, A. and HALPIN, T. (2000): Object Role Modeling and XML Schema. Proc. International Conceptual Modeling Conference, Salt Lake City, USA, 309-322, Springer.
- [5] 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
- [6] Mikael R. Jensen, Thomas H. Møller, Torben Bach Pedersen. Converting XML Data to UML Diagrams for Conceptual Data Integration. DIWeb'2001. pp.17~31
- [7] Grant, E. S., Chennamaneni, R. and Reza, H.: Towards Analyzing UML Class Diagram Models to Object-Relational Database Systems Transformations. In Databases and Applications, pp. 129{134, 2006.
- [8] Marcos, E., Vela, B., and Cavero, J. M.: Extending UML for Object-Relational Database Design. In 4th Int. Conf. on the Unified Modeling Language, vol. 2185, pp. 225{239, 2001.
- [9] Marcos, E., Vela, B. and Cavero, J. M.: A Methodological Approach for Object-Relational Database Design using UML. Soft. and Syst. Modeling, vol. 2, pp. 59{75, 2003.
- [10] Maatuk, A., Ali, M. A. and Rossiter, N.: An Integrated Approach to Relational Database Migration. In IC-ICT '08, pp. 1 {6, Bannu, Pakistan, 2008.
- [11] Maatuk, A., Ali, M. A. and Rossiter, N.: Converting Relational Databases into Object relational Databases. In JOT, Vol. 9, No. 2, March-April 2010 .
- [12] Urban, S. D., Dietrich, S. W. and Tapia, P.: Succeeding with Object Databases: Mapping UML Diagrams to Object-Relational Schemas in Oracle 8. John Wiley and Sons, Ltd, pp. 29{51, 2001.
- [13] Fong, J. H.K. Wonga, Z. Cheng, Converting relational database into XML documents with DOM. Information and Software Technology. v45. 335-355,2003.
- [14] J. Fong, Converting relational to object-oriented databases, ACM SIGMOD RECORD 26 (1) (1997) 53–58.
- [15] K Narayanan, S Ramaswamy in Proceedings of the 4th Workshop in Software Model Engineering (2005)
- [16] N. Routledge, L. Bird, A. Goodchild "UML and XML Schema" Australian Computer Science Communications Volume 24 Issue 2, January-February 2002
- [17] John Heintz and W. Eliot Kimber, 2000, Using UML to define XML document types. In Proceedings of isogen international.