Towards a bidirectional transformation of UML and XML models

Ludovic Menet, Myriam Lamolle

EEE'08

Towards a bidirectional transformation of UML and XML models

Ludovic Menet*, , Myriam Lamolle***

* Laboratoire d'Informatique et de Communication (LINC)
IUT de Montreuil, University of Paris8
140 rue de la Nouvelle France, 93100 Montreuil. France
+33148703461
Corresponding author: l.menet@iut.univ-paris8.fr

** Orchestra Networks, direction of R&D
75 boulevard Haussman, 75008 Paris. France
ludovic.menet@orchestranetworks.com

ABSTRACT. *Since the end of the nineties, XML has become the standard to exchange and send information on Internet. The W3C has recommended the use of XML Schema to define the structure of XML documents. To date, the graphical modelling of XML Schema models is not standardized. The introduction of a models definition formalism is a mean to make modelling more accessible. UML is a modelling object language which is more and more used and recognized as a standard in the software engineering field, which makes it an ideal candidate for modelling of XML Schema models. In this paper, we introduce the specificities of UML formalism to facilitate the definition of XML models. A semantic enrichment is done in UML and XML Schema with the aim of achieving a bi-directional mapping between these two standards.*

KEYWORDS: *UML, UML Profile, XML, XML Schema, Mapping, Metamodel.*

1. Introduction

It is now recognized that the XML (eXtensible Markup Language, W3C, 2000) language has become the standard to exchange information across Internet. XML Schema (W3C, 2004) is a formalism allowing to describe the structure of an XML document more easily than with a simple DTD (Document Type Definition). The use of XML seems to be adapted to the definition of models but it implies an extensive knowledge of this language. A lot of softwares such as Altova XML Spy and oXygen XML Editor have been developed to model graphically XML Schema models as trees. These softwares allow to optimise the modelling of XML schemas but each of them proposes a different formalism of representation. This observation allows us to think that a standard formalism should be used in order to make easier the modelling and the exchange of these models. This paper aims to introduce some modelling techniques of XML Schema models by the use of UML (Unified Modelling language) class diagrams.

UML proposes a powerful graphical modelling formalism widely used in software engineering. Our objective is to introduce all the richness of the UML formalism in order to make the modelling and the exchange of XML Schema models easier. The introduction of the UML formalism is performed by a mapping between the metamodels of UML and XML Schema. To do this, we proceed in several steps:

– Definition of a metamodel describing the concepts supported by XML Schema ;

– Definition of some XML Schema extensions in order to introduce the object specificities of UML. These extensions use the mechanisms of extensions recommended by XML Schema to be conform to the norm of the W3C;

– Definition of an UML profile to specialise UML to the semantic of XML Schema;

– Definition of the mappings between the metamodels of UML and XML Schema;

– Application of the mappings the use of the pivot format XMI (XML Metadata Interchange format) and XSLT stylesheets describing the mappings.

The first three points have been exposed in (Menet and Lamolle, 2007) and (Menet and Lamolle, 2008). In this paper we will focus on the last two points.

2. State of the art

(Levendovszky, 2002) defines the mapping as a set of transformation rules of models allowing to translate the instances of a source metamodel into an instance of a target metamodel.

(Baïna et al., 2006) are focused on the works of (Kalfoglou and Schorlemmer, 2003) to bring a mathematical definition of the interoperability of enterprises applications. After the study of these works, we notice that this definition can be applied to the mapping of models. Indeed, we consider two metamodels M_A and M_B, A and B are said interoperable if and only if a bijective mapping of M_A to M_B that we call *f* exists. The bijection of *f* guaranties that we can construct an instance of the B model from the instantiation of the A model (using *f*) and construct an instance of the A model from the instantiation of the B model. From this definition, three interoperability levels are defined, and mappings are identified between the languages A and B:

– Level 2: there is a total isomorphism between M_A and M_B. Then, all concept of M_A has its equivalent in M_B and vice versa. This means that M_A and M_B are equivalents;

– Level 1: there is a partial isomorphism between M_A and M_B. Then, there is a subset of M_A that we call M'_A and a subset of M_B (M'_B) such as the interoperability between M'_A and M'_B is of the level 2, these subsets are then equivalents;

– Level 0: there is no partial isomorphism between M_A and M_B. However, it is possible that some non bijectives mappings exist between M_A and M_B. In this case, we cannot talk about a semantic interoperability between A and B.

The interoperability of level two is the most difficult to establish in the sense that it is not common to have two metamodels totally equivalents. In this study, we are going to establish a partial matching between M_A and M_B computing the part of M'_A and M'_B compared to M_A and M_B getting then an interoperability of level 1 between XML Schema and UML.

(Carlson, 2001) deals with the transformation of UML models to XML Schema models and in the other way, getting then a bijective mapping between these two formalisms. The mapping realised uses an UML profile defining some XML Schema specific concepts. This profile is used in order to extend the semantic of an UML model to the semantic of an XML Schema model. This approach allows a mapping of a large part of the concepts introduced by XML Schema, but does not take in count some concepts such has groups (list, union), identity constraints (key, keyref, unique), genericity constraints (substitutionGroup), etc. Moreover, some important UML concepts such has aggregation, composition, association and documentation, are not taken in count during the transformation of an UML model into an XML Schema Model. This mapping has been applied by HyperModel (Carslon, 2006). HyperModel is a plug-in for the Eclipse IDE. This tool is fully operational but suffers of some limitations during the transformation of an UML model into a XML Schema model:

– Some concepts are not mapped (aggregation, composition, etc.);

– Some elements are mapped many times implying a redundancy of information and an inconsistency in the target model;

– A loss of information, more precisely about cardinality constraints, appears on some models;

– Some XML Schema models generated by Hypermodel are not valid according to the specifications of the W3C.

(Routledge and al., 2002) address the mapping between UML and XML Schema in a traditional way through the three levels approach from the databases domain, namely the conceptual, the logical and the physical levels. In the context of an UML class diagram, the conceptual level describes objects and their relations. The logical level represents the XML Schema data structure through an UML profile. The physical level directly represents the XML Schema model. As for Carlson's works, some UML specific elements such as aggregation, composition and other are not taken in count.

Other works have been realised in the same context by (Conrad and al., 2000), (Wu and Hsieh, 2002) and (Kurtev and al., 2003) but also suffer of the same limitations that the works previously presented.

In (Menet and Lamolle, 2007) and (Menet and Lamolle, 2008), we have enriched conceptually XML Schema by adding extensions allowing to define object relations, and UML by the definition of a profile specialising UML to the semantic of XML Schema, with the aim of establishing correspondences between these two

formalisms. These correspondences allow us to specify bidirectional mappings between UML and XML Schema.

3. UML / XML Schema mapping specifications

We will now introduce the rules of mapping between UML and XML Schema elements, including the concepts describing the relations of dependency.

3.1. Classes

In our rules, an UML class is mapped to an XML Schema element (<*xs:element*>) having a complex structure or to a global complex element (<*xs:complexType*>). In our previous works, we have introduced through an UML profile some stereotypes specialising UML to the semantic of XML Schema. Some of these stereotypes allow to add additional information to UML classes. So, the stereotypes <<Sequence>>, <<Choice>> and <<All>> are used for example to indicate that a classe has to be transformed into an XML Schema complex element having respectively a structure of the form <xs:sequence>, <xs:choice> or <xs:all>. The notion of root element has been defined in the semantic of XML Schema with the stereotype <<Root>>. If an UML class is declared as abstract, this property is materialised by the '*abstract=true*' attribute in the corresponding XML Schema element. These rules given, it is possible to perform the reverse operation, defining an UML class from an XML Schema element having these properties.

3.2. Attributes

UML attributes are transformed into simple XML Schema element <*xs:element*> with the attribute value <*xs:type*> a data type. The corresponding XML Schema element generated is included by a <*xs:sequence*>, <*xs:choice*> or <*xs:all*> element according to the stereotype applied to the UML class. We have introduced the stereotype <<Attribute>> in order to specify that the UML attribute has to be transformed into an XML Schema attribute <*xs:attribute*> and not into an element <*xs:element*>.

3.3. Cardinalities

UML cardinalities are transformed into attribute <*xs:minOccurs*> or <*xs:maxOccurs*> held by the corresponding element. Cardinalities may concern associations, aggregations, compositions and attributes.

3.4. Natives data types

The natives data types defined by UML such as *int*, *double*, *float*, *string*, etc., are mapped to the corresponding XML Schema entities, repectively <*xs:int*>, <*xs:double*>, <*xs:float*>, <*xs:string*>, etc.

3.5. Redefined data types

A redefined data type allows to specify constraints on data types or to define new business types. We have defined the stereotype <<*SimpleType*>> allowing to indicate that the corresponding UML element is a redefined type. In UML, we defined the properties of the redefined data type with an annotation. The corresponding XML Schema element is constructed from the information defined in this annotation. In our mapping, we include the XML Schema constraints on data types. It is then possible for a redefined data type to specify some constraints conform to the specification of the W3C such as *length, min/max length, enumeration, fractionDigits, totalDigits, min/max Inclusive, min/max Exclusive.*

3.6. Generalisation / Specialisation

An UML class stereotyped <<*abstract*>> is declared as abstract and is a generalisation of one or many concepts (classes). In our previous works, (Menet and Lamolle, 2007) and (Menet and Lamolle, 2008), we have defined XML Schema extensions allowing to materialise "object" relations. Generalisation and specialisation are mapped in XML Schema to an element with the following extension:

```
<xs:annotation>
    <xs:appinfo>
    <osd:generalisation> < !-- or specialisation -->
            <osd:conceptPath>
                Path of the sub / super concept in the schéma
            </osd:conceptPath>
    </osd:generalisation > < !-- or specialisation -->
    </xs:appinfo>
</xs:annotation>
```

3.7. Aggregation and composition

The aggregation (respectively composition) is mapped to an XML Schema element with the following extension:

```
<xs:annotation>
    <xs:appinfo>
    <osd:aggregation />  < !-- respectively composition -->
    </xs:appinfo>
</xs:annotation>
```

UML cardinalities are represented by the XML Schema attributes <*xs:minOccurs*> et <*xs:maxOccurs*>.

3.8. Notes and documentations

Documentation or a note is a low variant of constraint; it is a text for the reader but not executable by a computer. In an UML class diagram, a note is linked to the element it refers. Notes and documentations are translated into XML Schema by the following extension:

```
<xs:annotation>
    <xs:documentation xml:lang = « en_EN »>
            <osd:description> Text </osd:description>
    </xs:documentation>
</xs:annotation>
```

The *xml:lang* attribute allows to specify the associated language to the note or documentation.

4. Mappings application

We have presented in the previous section the mappings between UML and XML Schema basic elements. UML is a graphical formalism for models representation and XML Schema is a description language for structured models. The application of our mappings needs to be able to represent XML Schema and UML models in a common formalism.

4.1. XML Metadata Interchange Format (XMI)

XMI is a formalism of models representation, recommended by the ODMG (Object Management Group, 1999) created for the purpose to exchange models between modelling tools. The UML representation of a model represents the abstract level of this one; the concrete level is materialized by the XMI representation of this model. XMI can be used as a pivot format to represent and manipulate an UML model. An XMI document has the structure of an XML document. An XMI document contains two important parts: a *"header"* and a *"content"*. The *"header"* part defines the information about the model such as owner, description, date, etc. The *"content"* part defines the content of the model, i.e. the classes, stereotypes, associations, etc. Several softwares (such as ArgoUML, Rational Rose, Objecteering, etc.) include some functionalities of import and export of UML models in the XMI format. However, these softwares can generate for the same UML model a valid XMI document but not having the same structures, making difficult the exchange of models between applications. The major drawback requires us to apply our own mappings to generate XML Schema, correspondingly to the W3C standards, in order to facilitate the exchange and interoperability of models between applications.

4.2. XSL Transformation

XMI can be used to represent an UML model as an XML document, then allowing to establish a correlation between an UML model and an XML Schema model. XSLT is a language allowing to apply transformation rules to an XML document producing an other XML document. These transformation rules are described in an XSL stylesheet. The figure 1 describes the process of transformation:

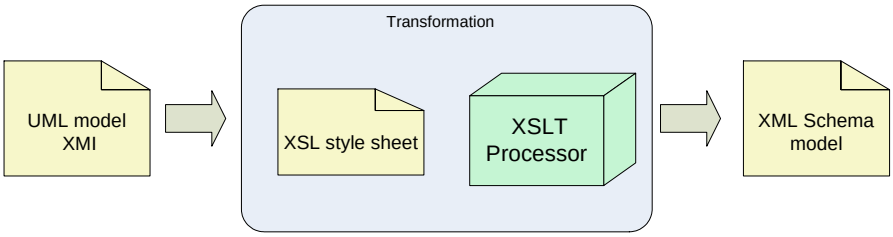

Fig. 1. *XSLT process transformation.*

4.3. Case study

This section presents a part of an UML model and its equivalent in XML Schema after the application of the mapping rules that we have established. We also compare this conversion with a mapping realised with HyperModel. Figure 2 presents an UML class diagram of a simplified network train. This UML model is composed of concepts and relations between them. In our example, we highlight the relationships of association, aggregation, composition and redefined types. Then we have for example defined the concept *Train* as composed of an *engine, wheels* (notion of composition), being able to have *wagons* (notion of aggregation) and having properties such as a *type* and a *trademark*. We associate a driver of type *PersonType* with a train. The concept *PersonType* defines properties such as *firstname*, *lastname* and *birthday*. We have also defined the property *email* representing the use of a redefined type. The class *Email* uses the stereotype <<*SimpleType*>> indicating that it is a redefined type in the XML Schema sense. The properties of this redefined type are expressed in an UML annotation specifying values for *SimpleType::base*, *SimpleType::pattern* and *SimpleType::whitespac*e. The root of the schema is specified by the stereotype <<*Root*>>, applied to the class *NetworkTrain*. The classes defined in this model are also stereotyped <<*Sequence*>>. This stereotype specifies that the corresponding class represents a complex element of the type <xs:sequence> in the XML Schema definition. From this diagram an XMI document is generated and we apply on it an XSL style sheet. This style sheet is composed of rules defining the mapping that we have previous presented[1].

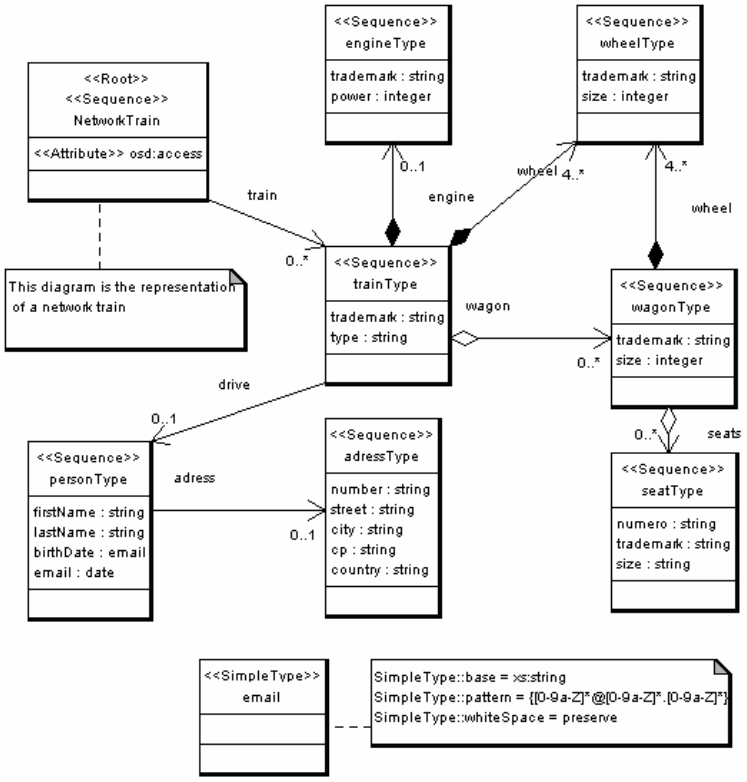

Fig. 2. Exemple de modèle UML.

After the application of our mappings, we get an XML Schema model. Figure 3 presents the transformation of the Networktrain and trainType classes and their relations, into XML Schema elements:

1. We have chosen to not present our style sheet because it is only a technical implementation of our mappings.

```
... <xs:element name="NetworkTrain" osd:access="--">
 <xs:annotation>
  <xs:documentation>This diagram is the representation of a network train
  </xs:documentation>
 </xs:annotation>
<xs:complexType>
 <xs:sequence>
  <xs:element name="train" type="trainType" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
</xs:element>
<!—Train-->
<xs:complexType name="trainType">
 <xs:sequence>
  <xs:element name="trademark" type="xs:string"/>
  <xs:element name="type" type="xs:string"/>
  <xs:element name="wagon" type="wagonType" maxOccurs="unbounded">
   <xs:annotation>
    <xs:appinfo>
     <osd:aggregation/>
    </xs:appinfo>
   </xs:annotation>
  </xs:element>
  <xs:element name="engine" type="engineType">
    <xs:annotation>
     <xs:appinfo>
      <osd:composition/>
     </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="wheel" type="wheelType" minOccurs="4" maxOccurs="unbounded">
    <xs:annotation>
     <xs:appinfo>
      <osd:composition/>
     </xs:appinfo>
    </xs:annotation>
  </xs:element>
  <xs:element name="drive" type="personType"/>
 </xs:sequence>
</xs:complexType>
<!--End train -->
<!—Redefined type-->
<xs:simpleType name="email">
 <xs:restriction base="xs:string">
  <xs:whiteSpace value="preserve"/>
  <xs:pattern value="{[0-9a-Z]*@[0-9a-Z]*.[0-9a-Z]*}"/>
  </xs:restriction>
</xs:simpleType> ...
```

Fig. 3. *Extract of an XML Schema model generated from an UML model with our mapping rules.*

As a comparison, we have defined in HyperModel the same UML class diagram that the one presented in figure 2, and we have then used the functionalities of transformation proposed by HyperModel in order to get an XML Schema model. During our comparative study, we have noticed some losses of information at the end of the transformation processed by HyperModel:

– Loss of documentation. The documentation associated to the *NetworkTrain* class has not been defined during the transformation. Unlike HyperModel, our mappings handle the definition of documentation associated to UML elements such as classes, attributes and relations (association, aggregation, etc.).

– Loss of constraints on redefined types. The constraints on the redefined type *email* defined in the class *<<SimpleType>> email* have not been taken in count during the transformation.

– Loss of semantic about relationships between concepts. Indeed, the relationships such as association, aggregation and composition are confused during the mapping between the UML class diagram and the XML Schema model. The mappings we have defined explicitly express the nature of the relations between concepts through the introduction of annotation in XML Schema models.

5. Conclusion

The need to communicate and to exchange is the result of various evolutions in the model development process. Indeed, the increasing size and complexity of models, the heavy use of networks and the intervention of different specialists in the field in the design imply communication and exchange. We have seen that we can facilitate communication between people by using a common formalism, UML in our case, as an abstraction of any technical language. UML is a powerful and flexible modelling language and XML became a standard for data interchange on the Web. The use of these two technologies addresses interoperability and a standard means of exchanging and defining models. The introduction of UML has been realised by the creation of matches between XML Schema and UML, matches that have enabled to define bidirectional mappings between these two standards. Introducing extensions in a XML Schema model and defining an UML profile, we get an interoperability of level 1 between UML and XML Schema filling some limitations that we have presented in sections 2 and 4. Indeed, our extensions allow us to establish some matches between concepts that have not been taken in count in previous works such as the notions of aggregation, composition, documentation, etc. Managing the notions of aggregation and composition allows to highlight explicitly the semantic links between elements and then bring important information about their life cycle.

The following of our works will be focused on the improvement of our mappings for example considering the definition of OCL (Object Constraint Language) (OMG, 2002) constraints and the development of a tool allowing to define UML and XML schema models in order to make the semi-automation of the mappings and the validation of models by the experts of the domain easier.

6. References

Altovan XMLSpy. http://www.altova.com/xmlspy

Ambler S. « Persistence Modeling in the UML», *Issue of Software Development*. August. http://www.sdmagazine.com , 1999.

Carlson D., *Modeling XML Applications with UML: Practical e-Business Applications*. Addison-Wesley Inc., 2001.

Carlson D., «Semantic Models for XML Schema with UML Tooling», In Proceedings of the 2nd International *Workshop on Semantic Web Enabled Software Engineering*, 2006.

Conrad R., Scheffner, D. and Freytag, J.C., « XML conceptual modeling using UML », Proceedings of the 19th *International Conference on Conceptual Modeling* (ER'2000), 2000.

Iyengar, S., A. Brodsky, XML Metadata Interchange (XMI). Proposal to the OMG Object Analysis & Design Task Force RFP 3: Stream-based Model Interchange Format (SMIF), Object Management Group. http://www.omg.org. (1998).

Kurtev, I., Berg, K.V., Aksit M., UML to XML-Schema Transformation: a Case Study in Managing Alternative Model Transformations in MDA. Forum on specification and Design Languages. (2003)

Levendovszky T., Karsai1 G., Maroti1 M., Ledeczi1 A., Charaf H., « Model Reuse with Metamodel-Based Transformations », *In Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools*, p. 166 -178, 2002.

Menet L., Lamolle M., « Meta-modelling "object": expression of semantic constraints in complex data structure », Proceedings of *ERIMA07'*. p. 104-108, Biarritz, France, 2007.

Menet L., Lamolle M., « Managing Master Data with XML Schema and UML », International Workshop on Advance Information Systems for Enterprises (IWAISE). Constantine, Algeria, 2008. (To appear)

Routledge N., Bird L., Goodchild A., « UML and XML schema », In Proceedings of the 13th *Australasian Database Conference*, pp 157-166, Melbourne, Australie, 2002.

ODMG. The Object Data Standard : ODMG 3.0. Morgan Kauffman Publishers, 1999.

OMG. Response to the UML 2.0 OCL. http://www.omg.org/docs/ad/02-05-09.pdf ,2002.

oXygen XML editor. http://www.oxygenxml.com

W3C. Extendible Markup Language (XML) 1.0. W3C XML Working Group, http://www.w3/org/TR/REC-xml (2000)

W3C. XML-Schema Part 1: Structures 2nd Ed. 2004 http://www.w3.org/TR/xmlschema-1 (2004)

Wu I.C., Hsieh S.H., "An UML-XML-RDB Model Mapping Solution for Facilitating Information Standardization and Sharing in Construction Industry", In Proceedings of the *National Institute of Standards and Technology*. Gaithersburg, Maryland. September, 2002.