December 2004

Toward a Generic Transformation between B2Bi Standards of UML and XML

Jia-Lang Seng
National Chengchi University

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Toward a Generic Transformation between B2Bi Standards of UML and XML

Jia-Lang Seng
National Chengchi University
jljan@nccu.edu.tw

ABSTRACT
The (eXtensible Markup Language) XML is fast becoming the standard data exchange across the Internet. It is suitable to describe complex-structured data. It also provides a validation mechanism to validate vocabulary used in the XML document. In order to exchange XML document, we need to define XML vocabulary by schema language. As a result spread out the issues of XML schema design from here. In the research, UML and XML schema transformation issue have been studied to develop a bi-directional mapping and transformation model. We design the bidirectional mapping and transformation model in two parts. We develop a sequence of production rules from the viewpoints of syntactic and semantic transformation. We develop production rules from the viewpoints of XML constructs. We build the transformation model to make transformation method more explicit, simple and easy to use.

Keywords
XML, UML, XMI, B2Bi, business model, transformation model.

INTRODUCTION
Research Motivation
The eXtensible Markup Language (XML) is fast becoming the standard data exchange across the Internet. It is hierarchical-structured and suitable to describe complex-structured data. It also provides a validation mechanism to validate vocabulary used in the XML document. In the Business to Business integration (B-to-Bi, B2Bi) environment, in order to escape from each making its claims independent of the other and cannot reach semantic interoperability, standard XML vocabularies need to be defined. At the present time, there are two kinds of data definition schema language mostly used to define XML vocabulary. One is Data Type Definition (DTD) and the other is W3C (World Wide Web Consortium) proposed XML Schema.

Using data definition schema language to model data is not easy. Besides hard to understand the syntax, this hierarchical design method would let data modelers unable to comprehensively focus on the data in the analysis time. Furthermore, the results are not suitable to communicate with the end-user. Unified Modeling Language (UML) is a visualized graphical design tool. It is characteristic of semantic expressiveness and has been used for a long time in software engineering domain. It provides several graphical design tools to use in the system analysis, design and implementation phases thus connectivity are improved. Especially, it can easily communicate with project team members and end-users by means of diagrams.

If data modeler could model XML vocabulary with UML, the problem described above can be solved. We can position UML as an XML data modeling tool and XML as an implementation language in the development of software engineering. They individually focus on their improvement in characteristics and domain and advance data exchange transparency and re-usability.

Research Issue
When using UML in data modeling (main class diagram), it needs to transform UML resulting model to XML schema like Java or C++ code. At the present time, there are some works relevant to this. Object Management Group (OMG) proposes XML Metadata Interchange (XMI) specification as a metadata transformation standard. Some software accesses an XMI file, exported from a UML model, to map constructs in class diagram to XML schema constructs. However, the core concept of XMI is not built specially for UML model, so the framework is too enormous and complex to understand by software engineers. This is the first issue. Second, some related works about transformation from UML class diagram to XML schema does not consider their intrinsic difference in construct. They start with UML and think about UML construct but not XML...
construct. The transformation rules should develop from the viewpoint of XML construct and UML construct. Third, XML-related techniques are developed increasingly; some transformation methodology is out-of-date to use. Therefore a methodology that could transform UML data model to XML schema syntactic and semantic completely is needed. Besides, an existing XML schema can be transformed to UML on the other direction at reverse side is another extensively discussed issue. A complete transformation from XML schema to UML is also valuable and needed.

Mapping UML to XML

XML Metadata Interchange (XMI)

XML Metadata Interchange (XMI) is developed by OMG. It defines a mapping between XML and objects. If two models exported from different modeling tools communicate with each other via XMI standard mapping rules, it ensures that objects are shared consistently. This is the goal of XMI.

At the present time, four versions are adopted by OMG. XMI 1.0 and XMI 1.1 are adopted in February 2000. XMI 1.2 and XMI 2 are adopted in November 2001. XMI builds on the XML. As XML related technique was promoted to a recommendation by W3C, XMI involves it in the new version. For example, at the time XMI 1.1 versions standardize, the XML namespace specification was adopted by W3C, XMI involves XML namespace mechanism. Similarly, XML Schema is only a draft version when XMI 1.2 standardizes, therefore XMI 1.2 and previous version only support DTD. Until XML Schema is fully recommended, XMI 2 provides support of XML Schema. XMI 2 specifies how to create XML schema from models.

The XMI specification is not specific to UML. XMI produces a DTD (XMI 1.1) or XML Schema (XMI 2) for any Metamodel that is compliant with the OMG’s Meta Object Facility (MOF) and produces an XML document instance for a model that instantiates that Metamodel. UML is defined using MOF, so the XMI production rules can be applied to map a UML-designed model to XML.

(Gros, Doney and Brodsky, 2002) illustrate the design rationale of XMI 2 using XML Schema and UML:

1. Packages: It is optional to specify an XML namespace for each UML package. If namespaces are specified, target namespaces are created.
2. Classes: XMI specifies that each class in a model creates a complex type declaration and an element declaration.
3. Datatypes: XMI maps a UML datatype to an XML schema simple type, a primitive schema datatype or one of the simple types defined in the schema. XMI creates a simple type for a UML enumeration that restricts the XML schema datatype to the string. The simple type is named with the name of the enumeration. XMI creates an XML element declaration for each UML attribute.
4. Attributes: XMI creates an XML element declaration for each UML attribute. It is possibly an XML attribute declaration as well, depending on the type of the UML attribute.
5. Association ends: XMI creates an XML element declaration and an XML attribute declaration.
6. Inheritance: XMI puts inherited attributes and inherited association ends in the complex type declaration for a class, along with the local declarations.

Three Level Transformation and Algorithm

(Routledge, Bird and Goodchild, 2002) proposed a method for UML and XML Schema mapping. They defined a mapping between the Unified UML class diagrams and XML Schema using the traditional three-level database design approach (i.e. using conceptual, logical and physical design levels). In this approach, the conceptual level is represented using standard UML class notation, annotated with a few additional conceptual constraints, the logical level is represented in UML, using a set of UML stereotypes, and the XML Schema itself represents the physical level. The goal of this three-level design methodology is to allow conceptual level UML class models to be automatically mapped into the logical level, while minimizing redundancy and maximizing connectivity.

The Metamodel in Figure 2-4 shows the relationships between XML Schema concepts such as “element”, “complexType”, “simpleType” and “XSD simpleType”. These XML Schema concepts are represented as stereotyped classes, allowing them to be used in logical level UML class diagrams to represent the corresponding XML Schema concept. Two of the relationships between these concepts, are namely “restricts”, and “extends”. This was done to allow for instance
substitutability between related user-defined types. The relationship “has type” is represented as a stereotyped dependency between an “element” and either a “simpleType” or “complexType”.

**Stereotypes for XML Schema Constructs**

The primary challenge in using something like UML to directly model XML document types is known how application-specific types are to be mapped to XML syntactic constructs: elements, attributes, notations, character data, and etc. Another challenge is the mechanism by which detailed syntactic constraints are specified. Kimber and Heintz (2000) address a straightforward approach. Each XML syntactic construct is represented by a stereotype that is applied to the UML construct used to define the XML syntactic construct.

Content model constraints are specified through a combination of types with the stereotype “<<model-group>>” and content constraint specifications using normal XML model group syntax. Model-group types capture groups of related types that can satisfy a particular point in a content model while content constraint specifications further constrain occurrences of instances of those types within the “content” property of their container. The XML Stereotypes package defines the set of stereotypes that refer to aspects of the metamodel for XML. The XML DTD Stereotypes and transformation algorithm they developed are illustrated.

**GENERIC UML CLASS DIAGRAM CONSTRUCTS AND XML SCHEMA BI-DIRECTIONAL TRANSFORMATION**

In this Chapter, we describe and discuss the research approach and research model. According to the literature review of Chapter two, there have been some related works and discussions of UML and XML transformation. This research focuses on XML data design using class diagrams of Unified Modeling Language. Through the more complete and comprehensive transformation of UML data model, this research produces an XML schema, W3C XML Schema or DTD, which is used to build XML document and validate XML document. Besides this forward engineering, we also develop a reverse engineering model that transforms XML schema to UML model. If bi-directional transformation between UML class diagram generic construct and XML schema is workable, thus users are unable to integrate data modeling, data design and implementation phases of system development and further enhance system development integration and data exchange.

The research architecture is illustrated in Figure 1.

![Figure 1. Class Diagram of UML Data Model and XML Schema Transformation](image)

Due to UML class diagram and XML schema be intrinsically different in models, we present the bi-directional mapping and transformation model in two parts. One is from UML class diagram to XML schema and another is from XML Schema to UML model. We describe the first part in Section 3.2 and 3.3 and the second part in Section 3.5 and 3.6.

Not only syntactic mapping but also semantic mapping is a very important issue in transformation. This is one of the key points of our research. Table 1 below is a comparison between UML class diagram generic constructs and XML schemas generic constructs from the viewpoint of UML. Table 2 below is a comparison between UML generic constructs and DTD generic constructs from the viewpoint of DTD and Table 3 below is a comparison between UML generic constructs and W3C XML Schema generic constructs from the viewpoint of W3C XML Schema.
Table 1. Comparison of UML Constructs and XML Schemas Constructs

<table>
<thead>
<tr>
<th>XML DTD Constructs</th>
<th>UML Constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Supported</td>
</tr>
<tr>
<td>Attribute</td>
<td>Supported</td>
</tr>
<tr>
<td>Attribute ordering</td>
<td>Not supported</td>
</tr>
<tr>
<td>Datatype</td>
<td>Supported</td>
</tr>
<tr>
<td>Nested element</td>
<td>Supported</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Supported</td>
</tr>
<tr>
<td>Relation</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Table 2. Comparison of XML DTD Constructs and UML Constructs

<table>
<thead>
<tr>
<th>W3C XML Schema Constructs</th>
<th>UML Constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Supported</td>
</tr>
<tr>
<td>Attribute</td>
<td>Supported</td>
</tr>
<tr>
<td>Attribute ordering</td>
<td>Not supported</td>
</tr>
<tr>
<td>Datatype</td>
<td>Supported</td>
</tr>
<tr>
<td>Simple type</td>
<td>Supported</td>
</tr>
<tr>
<td>Complex type</td>
<td>Supported</td>
</tr>
<tr>
<td>Nested element</td>
<td>Supported</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Supported</td>
</tr>
<tr>
<td>Relation</td>
<td>Supported</td>
</tr>
<tr>
<td>Namespace</td>
<td>Supported</td>
</tr>
</tbody>
</table>

Table 3. Comparison of XML Schema Constructs and UML Constructs

Transformation from UML Class Diagram to XML Schema

Transformation Structure

In the beginning the first part, we develop how UML class diagram constructs map to XML schema. The transformation structure is illustrated in Figure 2.
As shown in Figure 2, we start with a UML Class diagram in the XML data design phase and then we move on to map it to the XML data structure definition constructs. In this phase, we present the analysis in two parts. One is mapping from UML class diagram to DTDs, another is mapping from UML class diagram to W3C XML Schema. We develop the transformation rules how “datatype”, “class”, “attribute”, “association”, “aggregation”, “composition”, “generalization and “multiplicity” constructs in the UML class diagram map to XML constructs in different XML schema languages.

**UML Class Diagram Step**

Figure 3 is a conceptual UML class diagram. It describes the training courses a company offers. Some are taught in the classroom and some are learned on Internet. Every course must record its list price and pre-requisite lessons. The materials are supplied from many companies. Every course provides a demonstration on Internet. The example is modeled using Rational Rose 2002 in this Chapter. We use this simple example throughout this Chapter to develop transformation rules for DTD from Section 3.2.1 to Section 3.2.6 and for W3C XML Schema from Section 3.3.1 to Section 3.3.7.
The transformation rules we develop are based on OMG’s XMI version 1.2 specifications. We enhance “user-defined type”, “generalization” and “association” transformation rules in our research. They are basic principles how UML generic constructs map to XML DTD constructs. Under these principles, a set of stereotypes is developed to tailor the resulting structure of XML DTD. OMG defines a set of stereotypes with tag values and constraints for transformation, which is called UML profile. Besides OMG, the Rational Software Corporation and other authors also develop UML profiles. In this research, OMG’s UML profile is fundamental to enhance. Not only from the terms of constructs of UML class diagram, XML constructs are considered as well. Table 3-5 is a list of stereotypes created for DTD based on OMG UML profile and Kimber’s work (Kimber, 2000).
Transformation Rules from UML to XML Schema

The transformational rules from UML class diagram to XML Schema we develop are based on OMG’s XMI version 2 specifications. We enhance the semantic transformation part to map “composition”, “association” and “aggregation” and “generalization” constructs. The rules are basic principles how UML generic constructs map to XML Schema constructs. Under these principles, a set of stereotypes is developed to tailor the resulting structure of XML Schema. In this research, OMG’s UML profile is fundamental to enhance. Table 3-6 is a list of stereotypes created for XML Schema based on OMG UML profile.
Figure 5. Transformation Structure from UML Constructs to W3C XML Schema Constructs

<table>
<thead>
<tr>
<th>Stereotypes For XML Schema</th>
<th>UML Constructs</th>
<th>Description for XML Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;XSDschema&gt;&gt;</td>
<td>Package Component</td>
<td>If a UML package is stereotyped with it, all elements of the package are contained within one XML Schema. If a UML component is stereotyped with it, all of its assigned classes are contained in a schema definition.</td>
</tr>
<tr>
<td>&lt;&lt;XSDcomplexType&gt;&gt;</td>
<td>Class</td>
<td>Map to a complexType definition</td>
</tr>
<tr>
<td>&lt;&lt;XSDsimpleType&gt;&gt;</td>
<td>Class</td>
<td>Map to a simpleType definition</td>
</tr>
<tr>
<td>&lt;&lt;enumeration&gt;&gt;</td>
<td>Class</td>
<td>Map to a simpleType with an enumeration definition</td>
</tr>
<tr>
<td>&lt;&lt;XSDsequence&gt;&gt;</td>
<td>Class</td>
<td>Map to a sequence model group contained within a complexType definition</td>
</tr>
<tr>
<td>&lt;&lt;XSDchoice&gt;&gt;</td>
<td>Class</td>
<td>Map to a choice model group contained within a complexType definition</td>
</tr>
<tr>
<td>&lt;&lt;SimpleXLink&gt;&gt;</td>
<td>Class</td>
<td>Map to a complexType definition with simple for simple XLink</td>
</tr>
<tr>
<td>&lt;&lt;ExtendedXLink&gt;&gt;</td>
<td>Association</td>
<td>Map to a new complexType definition with attributes for extend XLink</td>
</tr>
<tr>
<td>XML Schema Element</td>
<td>UML Object</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&lt;&lt;XSIdtoschema&gt;&gt;</td>
<td>Generalization</td>
<td>The child class map to a complexType with a &lt;restriction&gt; child element</td>
</tr>
<tr>
<td>&lt;&lt;XSDelement&gt;&gt;</td>
<td>Attribute AssociationEnd</td>
<td>Map UML construct to an element definition, contained within complexType element definition</td>
</tr>
<tr>
<td>&lt;&lt;XSDattribute&gt;&gt;</td>
<td>Attribute AssociationEnd</td>
<td>Map UML construct to an attribute definition, contained within complexType element definition</td>
</tr>
<tr>
<td>&lt;&lt;XSDtopLevelElement&gt;&gt;</td>
<td>Component</td>
<td>Map to a top level element declaration</td>
</tr>
<tr>
<td>&lt;&lt;XSDLlist&gt;&gt;</td>
<td>N/A</td>
<td>Map UML construct to list definition in XML Schema</td>
</tr>
<tr>
<td>&lt;&lt;XSDunion&gt;&gt;</td>
<td>N/A</td>
<td>Map UML construct to union definition in XML Schema</td>
</tr>
<tr>
<td>&lt;&lt;URI&gt;&gt;</td>
<td>Association</td>
<td>Map to a simple XLink linking to local or remote resource</td>
</tr>
</tbody>
</table>

Table 4. UML Profile for XML Schema (The stereotype prefix “*” is one we develop in the research)

**Transformation from XML Schema to UML Class Diagram**

*Transformation Structure*

In this Section, we develop the transformation rules from XML Schema to UML class diagram. The transformation structure is illustrated in Figure 3-19.
As Figure 6 shows, we start with XML schema including DTD and W3C XML Schema and map UML constructs and XML schema constructs on the other direction at reverse side. In this phase, we also present the analysis in two parts. One is mapping from DTDs to UML class diagram, another is mapping from W3C XML Schema to UML class diagram. We develop transformation rules how “element”, “attribute”, “datatype”, “simple type”, “nested element” and “relation” construct or declaration in the XML schema maps in UML data model. In the Appendix, a complete XML DTD document and W3C XML Schema document are provided. We use these two simple examples to develop transformation rules for DTD from Section 3.5.1 to Section 3.5.5 and for W3C XML Schema from Section 3.6.1 to Section 3.6.5.

**Transformation Rules from XML DTD to UML**

The rules of transformation from DTD to UML class diagram we develop are based on OMG XMI version 2 specifications to enhance. According to these rules, XML DTD document is able to map to a UML class diagram and presents semantics in DTD itself completely into class diagram automatically or manually.
Transformation Rules from XML Schema to UML

In this Section, the transformation rules we develop to map XML Schema to UML are simpler than DTD. XML Schema is an XML-encoded language. The rules of inheritance, multiplicity and type constructs map to UML class diagram is obvious and certain. It is not as complex as DTD transformation rules. In this research, according to the transformation rules from UML to XML Schema, we present how “element”, “attribute”, “simple type”, “nested element” and “relationship” map to UML class diagram constructs.

SYSTEM IMPLEMENTATION

According to the research model described in Chapter 3, a UML class diagram editor with function of bi-directional transformation from UML class diagram to W3C XML Schema and from W3C XML Schema to UML class diagram is implemented. This system shows that the transformation model of this research is able to obtain a semantic equivalent artifact between forward engineering and reverse engineering. In this chapter, the system platform and the system architecture and design is described in the following sections.

System Platform

In our implementation, we choose Microsoft Visual Basic Version 6 as our programming language and Microsoft Windows XP home edition as the operating system. In the functional design of UML class diagram, we adopt the FlowChartX component, a demo version, of MindFusion Group as the graphical sketch foundation. Then, in the functional design of bi-directional transformation, we use DOM API provided in Microsoft XML Parser MSXML4 to access XML and XSD files. This editor is a single site version. It can execute on the platform of Windows XP, 2000 and Windows 98.

System Architecture

The transformation editor and processor architecture is illustrated in Figure 7.
Figure 7. System Architecture
Figure 8. Creating Association and Specifying Association Property
Figure 9. UML Composition Transformation Result
Figure 10. Selecting Transformed XSD File
RESEARCH DISCUSSIONS AND LIMITATIONS

According to the research model and system implementation, we summarize the research results to describe and discuss.

Research Discussions

The data modeler is able to comprehensively focus on the data in the analysis time when modeling XML vocabulary with UML. The data model in UML transforms to XML schema automatically and keep semantic equivalently is our initial motivation.

At the present time, Rose of Rational Software Corporation is the mostly used modeling tool to sketch UML class diagram. Rational Rose 2002 provides an add-in function of transforming UML class diagram to XML DTD and reverse engineered XML DTD to UML class diagram. The function of transforming between class diagram and XML Schema is not provided yet. Besides Rational Rose, software of Ontogenics Corporation called hyperModel implements the bi-directional transformation between UML class diagram and XML Schema. The hyperModel does not provide ability to sketch UML class diagram. It accepts XMI 1.0 (XML Metadata Interchange) file, which exports from Rational Rose or ArgoUML software, then transforms it to XML Schema. Because having an intermediary layer, XML, between class diagram and XML Schema, it is hard to avoid semantics misses and mismatches. The transformation method in hyperModel has some features. We describe in two parts.

1. Transformation from UML class diagram to XML Schema

The transformation rules of “aggregation”, “association” and “composition” constructs are the same. The relation maps to an XML element with a “ref” attribute. Although they are different in semantics in UML, they present the same semantics in XML Schema.
2. Transformation from XML Schema to UML class diagram

(1) When mapping relations, the semantics in UML are unable to recognize definitely. The construct “aggregation”, “association” and “composition” are mismatch. The assistant method for tailoring transforming XML Schema is needed.

(2) When mapping relations from XML Schema to UML, original association role and multiplicity transformed from UML class diagram miss.

In this research, we concentrate to reduce syntactic and semantics miss in the transforming process. The method we adopt is to implement a UML class diagram editor and the full properties of the diagram are stored in an XML file format, it enables us to focus on the generic constructs transformation and transform from class diagram to XML Schema directly. Summarize research model in Chapter 3 and system implementation in Chapter 4, the contributions are described as follows.

1. The transformation rule of aggregation or association is different from the way of that of composition. We create a virtual XML element through using IDREF to present the exact semantics of these two relations. We implement this transformation rule in transforming from class diagram to XML Schema. Also we are able to recognize association or aggregation in the reverse side transformation and get a semantic equivalent mapping result successfully. In addition, the way of association and aggregation transformation produces an XML Schema, which reduces data redundancy possibility and improves data independency on XML implementation.

2. We build up a bi-directional system to transform between UML class diagram and XML Schema. The result proves our research model to be workable. In the system architecture, we use a self-design XML file format to catch all properties of class diagram instead of XMI (XML Metadata Interchange). This XML document structure is easy and elastically. In the transforming process, the processor is easy to parse relations among constructs. It makes our transformation model to be implemented successfully. Furthermore, this XML document structure is extensible to describing the other UML diagrams. Without intermediary of XMI, the resulting XML Schema from this research transformation is more genuine and generic. Related W3C XML techniques can be applied in a natural fashion.

<table>
<thead>
<tr>
<th>Bi-Directional Mapping Constructs</th>
<th>HyperModel</th>
<th>This Research Transformation Editor and Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Full equivalence</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Attribute</td>
<td>Full equivalence</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Full equivalence</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Association</td>
<td>No</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Aggregation</td>
<td>No</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Composition</td>
<td>No</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Association role</td>
<td>No</td>
<td>Full equivalence</td>
</tr>
<tr>
<td>Multiplicity constraints</td>
<td>No</td>
<td>Full equivalence</td>
</tr>
</tbody>
</table>

Table 5. Comparison of Bi-Directional Mapping Result

### Research Limitations

There are several limitations in this research. We describe them as follows.

1. We do not develop transformation rules of multi-inheritance. One part of reason is XML Schema does not support this construct. The other part of reason is we do not design a more perfect rule than other experts.

2. The transformation editor does not provide function of import PTL file format, which is exported from Rational Rose UML model. That means it is unable to read UML model created in Rational Rose. The reason is we want to comprehensively focus on the transformation rules in our bi-directional transformation model. In the future, it needs to add this function to communicate with Rational Rose.

3. The transformation editor does not provide function of tailoring XML Schema. We accept only XML Schema, which results from our transformation editor forward engineering.
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Summary

The objective of this research is to develop the transformation model to make transformation method more generic, genuine, systematic, explicit, simple and easy to use.

1. Enhances the software development integration in XML application analysis and design

   In forward engineering, XML Schema needs to be able to express original semantics in UML data model. In reverse engineering from XML Schema to UML, relation among elements is also the key point to extract. In our research, we design the transformation parser to extract relations among elements, mapping to relations among classes in UML. When UML data model and XML Schema is able to do bi-directional transformation, XML Schema is able to transform to UML data model. Therefore, the re-usability of UML and XML Schema increases.

2. Bi-directional relation transformation

   In our research, the transformation rules of association, aggregation and composition are different. We implement it and do bi-directional transformation successfully.

3. Improve transformation accuracy and reduce transformation cost

   Due to the system in our research exactly improves transformation accuracy; the software engineer does not need to transform UML data model and XML Schema manually. The automatic degree increases. Therefore, the transformation cost, time and money, reduce greatly.

Future Research Directions

The transformation editor and processor implemented in the research are trying to validate the transformation model and get a satisfied result. Due to time and resource restriction, the research still has some improvement space in the future. The editor and processor could extend to sketch the other UML diagrams, like object diagram, sequence diagram, activity diagram or collaboration diagram, and then transform to XML Schema. At the same time, the editor can try to read PTL file exported from Rational Rose UML model and import the model into our editor. Thus the Rational Rose user can integrate this research model and method. We can adopt XSLT skill to translate the self-designed XML storage document to an XML document.

REFERENCES