Journal of Theoretical and Applied Information Technology
15th November 2012. Vol. 45 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CONVERTING CONCEPTUAL MODEL XUML TO XML SCHEMA

XUEMIN ZHANG

School of Computer and Information Science, Hubei Engineering University , Xiaogan 432000, Hubei, China

ABSTRACT

As XML has become the standard for representing structured and semi-structured data on the Web, the methods for designing XML schemas is becoming more and more important. At present, a new XML conceptual model - XUML has already been put forward, but lacks the method of converting XML conceptual model to XML logical model . In this paper, an approach is proposed which a conceptual model XUML is converted to the XML Schema. Finally, a specific application example is illustrated, which proves the method the effectiveness and practicality.

Keywords: *XML, XUML, XML Schema*

1. INTRODUCTION

As a universal format, XML (eXtensible Markup Language) realizes data representation and exchange. Many organizations use XML as data storage format, some published and document processing industries also choose XML documents, most of the database system also has support for XML database. But there isn't a proper mechanism to generate or describe XML conceptual model. At present some XML conceptual model has proposed in domestic and foreign, such as Semantic Network[1], AOM[2], ORA-SS[3], X-Entity[4] and C-XML[5], to support XML these models extend semantic network or ER model. Reference [6] defines a number of specialized Profile, extending UML1.x to support WXS. Compared to XML mode in logic level, these conceptual models can capture more semantic and constraint which improve XML schema design, but still not enough to support the XML conceptual modeling. The reference [7] improves reference [6], puts forward a new conceptual model –XUML which succeeded the strongpoint of foregoing models, enhanced in several important ways, but how to convert XML conceptual model to XML logic model has been not studied.

The XML logical model generally uses XML Schema. This paper presents a kind of method of converting XUML to XML Schema aimed at XUML concept model put forward in reference [7], which can achieve conversion from the XML conceptual model to XML logic model.

2. XUML MODEL TO XML SCHEMA MAPPING RULES

The representation method of mapping rule is described in detail in reference [8].

2.1 Attribute Mapping Rules

1) General attribute mapping

The attribute maps the element; an attribute name is as an element name; attribute type is as the element type.

Transformation AttributeToElement (XUML，WXS)

{source

attr：XUML:: Attribute；

target

elt：WXS::Element；

mapping

attr.name <～> elt.name；

attr.type <～> elt.type；

attr. multiplicity.lower <～> elt.minOccurs；

attr. multiplicity.upper <～> elt.maxOccurs；

}

2) The <<attribute>> attribute mapping

The attribute maps the attribute in the XML Schema; XUML attribute name maps the name of the Schema attribute; types for the XUML attribute map the Schema attribute types; other mark value maps behavior attribute and attribute value of the attribute.

2.2 Class Mapping Rules

1) Class <<element>> mapping rules

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

This class maps global element declaration; the class name is as element name or element type name, and as complex type; processing the class attribute according to the attribute mapping rules.

2) Class <<complexType>> mapping rules

The <<complexType>> class has two kinds of cases: the subclass in a no generalization relation (denoted as << class 1 complexType>>) and the subclass in generalization relation (denoted as class 2 <<complexType>>).

(1) Class 1<<complexType>> mapping rules

This class maps complex type definitions; the class name as the complex type name; model group convert according to the modelGroup value of marker value in class; attribute map is same as stated 2.1.

(2) Class 2<<complexType>> mapping:

This class maps derived complex type definitions, complex type as basic types converted from super class, <<extension>> generalization corresponding extended derivative, <<restriction>> generalization corresponding constraints derived; attribute map is same as stated 2.1.

3) Class <<choice>> mapping

This class maps the choice model group, embedded into the model group associated class; attribute map is same as stated 2.1.

4) Class <<sequence>> mapping

This class maps the sequence model group, embedded into the associated class into choice model group, the processing properties.

5) Class <<simpleType>> mapping

The <<simpleType>> class has three cases: the mark value is {ristrict} (denoted as class 1<<simpleType>>), the mark value is {list} (denoted as class 2<<simpleType>>) and the mark valuee is {union} (denoted as class 3<<simpleType>>).

(1) Class 1<<simpleType>> mapping:

This class maps simple type; the class name is as the name of simple type; marker values {base=value} as a base class of simple type; other marker values will be converted to facet of simple type definition.

(2) Class 2<<simpleType>> mapping:

This class maps simple type of list method; the class name is the name of simple type; the value of marker values {itemType=value} is itemType of simple type.

(3) Class 3<<simpleType>> mapping:

This class maps simple type of union method; the class name as a simple type name; the value of marker values {memberTypes=value} is as memberTypes of the simple type.

6) Class <<enumeration>> mapping

This class maps enumeration type in simple type definition; the class name as a simple type name; marker values {base=value} as a base class of simple type; enumeration value maps corresponding to the value of the enumeration facet.

2.3 Linkage Mapping Rules

1) Generalized aggregation mapping

Generalized aggregation connects the whole class and component class (1 or more), when mapped to the XML Schema, the whole class as a parent element, component class as child element nested in one.

2) Relationship mapping

Relationship mapping is mainly embodied in key and keyref definition. There are two kinds of relationship mapping.

(1) The internal relationship. There are two kinds of definitions of key and keyref:

①local definitions of key and keyref within the relationship range

The scope of internal relationship in the range the most recent common ancestor of two relationship class, local definition is the corresponding element definition in the common ancestor. On the field of definition, key corresponds to the main attribute of target class, while keyref corresponding to new attributes of source class. New attributes is same to the main attributes of target class or is named the role name of source end.

② Global definition of key and keyref in the system scope

The system scope, namely the root element, key and keyref are defined under the root element. On the field of definition, as the global definitions will be upgraded the system scope, key have to use composite primary key, namely the key is composted by all ancestors and its own main attributes under identifier class. Similarly, the keyref also uses joint attributes with respect to key.

Compared above two kinds of definition: local definition does not need to add the ancestor's main attributes as a composite (primary/foreign) key, but must identify scope; while global definition is although easy to identify scope, but add the main attributes of levels of the ancestor class as a composite (primary/foreign) key, for the level of the model is deep, composite (primary/foreign) key will become very large.

(2) Internal relationship across range

On the definition scope, the scope must be promoted to be a higher level across the scope. On the definition field, the composite property definition will be used. Target element key will combine main attributes across the scope and its

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

own main properties. Source class element keyref should also apply composite attributes with corresponding to the key.

3) Generalization mapping rules

Generalization relationship is mainly used to reflect the derived definition from complex type, it is divided into two generalizations: <<extension>> and <<restriction>> generalization. At the time of conversion, subclass derives based super class; derivation method depends on the kinds of generalization. The mapping rules see class <<complexType>> mapping rules for more information.

2.4 Related Constraint Mapping Rules

1) Prime attribute mapping

In XUML prime attributes identify a line under, representing the attributes value is only and not empty. In XML Schema there are two ways to describe the prime attribute.

(1) Defining the type as ID attribute can describe the prim attribute

When the attributes use ID type, then its corresponding reference attributes should define as IDREF (S) type.

(2) Define primary attribute through the key constraint mechanism.

Key not only can be applied to the attribute, also applied to the elements. Key can set the composite attributes; and key can clear its range of action. The reference attributes which corresponds to key define as keyref. The detailed definition sees relation mapping.

2) Attributes multiple mappings

The XUML property multiplicities, represents a total closure numerical range. Converted to XML Schema, lower bound of range maps the minOccurs value of the element, upper bound of the range mapping to maxOccurs value of the element, as illustrated in attribute mapping.

3) Relation cardinal numbers mapping

In general aggregation and the relationship, two relation ends may have cardinal numbers. When convert general aggregation, the cardinal number of component class end will be mapped to the minOccurs and maxOccurs values of corresponding elements.

4) Relationship's navigation mapping

Mapping Relationship's navigation to XML Schema mainly embody contains position and reference direction among the elements. Contains position and reference direction has been described in detail in generalized aggregation and relationship.

3. ALGORITHM FROM XUML MODEL TO XML SCHEMA

3.1 Processing Prior To Conversion

1) Keep the name uniqueness

In order to avoid name conflicts of element, attribute or type in subsequent conversion work, it would be better uniquely defines all class names in XUML model before conversion.

2) Note component class order

Components class in generalized aggregation will be converted into of global class. If the model group of global class elements is sequence, the order of sub-element adds notes before convert artificially according to demand.

3) Add mark class

In XUML model, the class expressing theme can appear multiple times, but because the XUML base number of identifier class is only to be one, so the class cannot be a mark class. In order to keep the semantic and conform to XUML syntax, a mark must be added on the class, and form the class to generalized aggregation relationship.

3.2 Algorithm From XUML To XML Schema

XUML model have element content and element type, general idea of conversion XUML to XML Schema is: first converting element type, generating a element type document, named SharedData.xsd; and then converting element content, generating the corresponding element content document, the document includes a element types document, finally get the whole XML Schema document. Conversion process diagram from XUML to XML Schema is shown in figure 1.

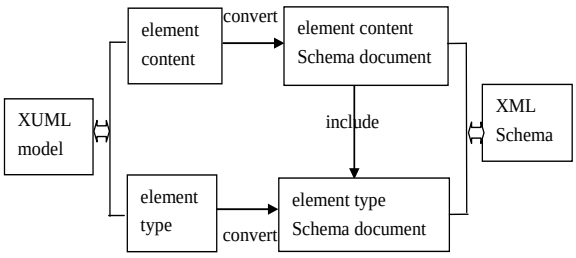

Figure 1: Conversion Process Diagram From XUML To XML Schema

XML Schema embodying the core essence is: nesting elements and types reuse. The XUML element content part main embody the mutual connection between the various XUML classes. these connection primarily are the various relations within generalized aggregation and components. The XUML element type part is mainly composed of various class types, such as: <<simpleType>>

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

class, <<enumeration>> class, <<complexType>> class, etc. <<complexType>> class may exist related generalization links between them. Element type part is for certain the simple or complex types to use many times in element content, of course, the element type part can also be reused by other business component, even further become an application domain vocabulary.

Algorithm of converting XUML to XML Schema is as follows:

Step 1: converting element type part:

1) Firstly, for each class in the element type, judge its classification:

(1) If it was class <<simpleType>>, according to the class <<simpleType>> mapping rule conversion;

(2)If it was class <<enumeration>>, according to the class <<enumeration>> mapping rule conversion;

(3) If it was class <<complexType>>, according to the class <<complexType>> mapping rule conversion.

2) Lastly named document as SharedData.xsd.

Step2: Converting element content, the generated Schema document called element content document, named by the identifier class. From identifier class start conversion level by level, the following describe the conversation process through a recursive algorithm:

1) To create the mark C corresponds to root element E, type E is complex type CT;

2) To define recursively globally complex type CT:

(1) If C does not reference type of the element type, the C's attributes according to the attribute mapping rules to generate CT's sub-elements;

(2) If association A in C exists, and as the source class in association, a new attribute in C must be added.

i. if A is internal association in the range, the number of new attribute is 1;

ii. If A is internal association across a range, the number of new attribute is multiple.

In two cases, additional attributes are generated a child element in accordance with the (1)

(3) If C has multiple independent binary generalized aggregation, converting to the internal association of scope, processing according to the (2);

(4) If C only one generalized aggregation, wherein the component class:

i. If it was class <<choice>> or class <<sequence>>, according to class <<choice>> mapping or class <<sequence>> mapping rule to convert;

ii. If it was class <<complexType>>, it can be regarded as the root of subtree, processing as (1);

(5)If C refers types of element type, CT will extend the derived definition, sub-elements of extension according to (2), (3) to generate.

3) name document as identifier class

Step 3: In element content, the relation between classes primarily through key and keyref to present. Because relation has multiple conditions, so the definition of key and keyref is not the same. Here separate generation algorithm for key and keyref are given:

For each class in element content:

(1) If it has main attribute, but has no relation, according to the mapping rules of main attribute to define key in the system scope;

(2) If it has no relation, for each relation to generate the corresponding key and keyref:

i. If the range is internal relation, processing according to mapping rule of internal relation;

ii. if internal association across a range, then processing according to mapping rules across a range of internal association.

Step 4: key and keyref generated on Step 3 must put in element content of document generated by Step 2, and elements content document must include SharedData.xsd document generated on Step 1.

Through the above 4 step conversion, a XUML model can be generated a XML Schema document.

4. APPLICATION EXAMPLES

An extended XUML model is shown in Figure 2.

XML Schema document generated by the above algorithm as follows.

SharedData.xsd document generated from element type is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

Class<<complexType>> of element type is converted to complex type "PaperType", such as:

```
<xs:complexType name="PaperType">
    <xs:sequence>
        <xs:element
name="Title" type="xs:string"/>
```

Journal of Theoretical and Applied Information Technology
15th November 2012. Vol. 45 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

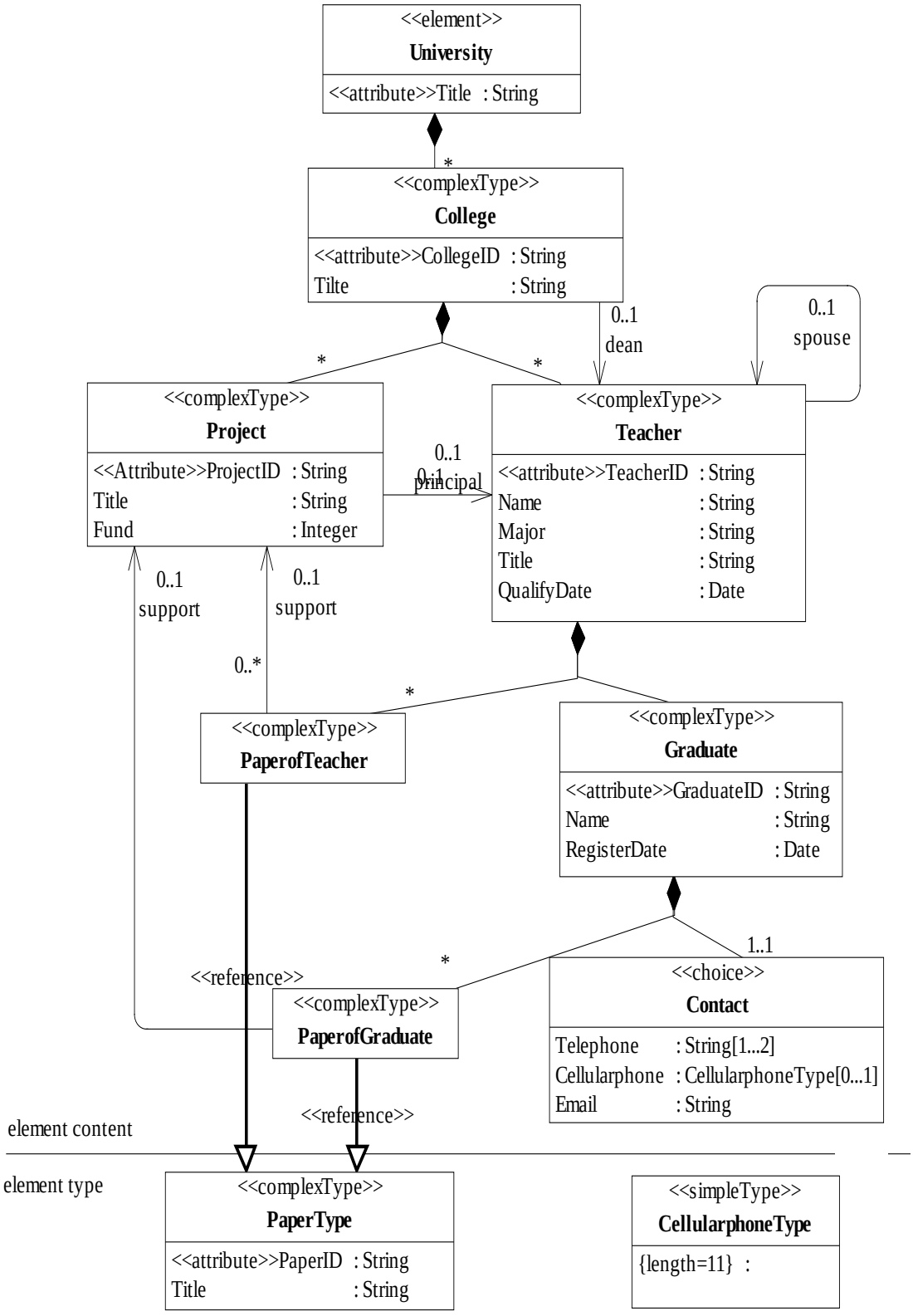

Figure 2: An Extended XUML Model

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
<xs:element name="publishDate" type="xs:date"/>
                </xs:sequence>
                <xs:attribute name="PaperID"
type="xs:string" use="required"/>
        </xs:complexType>
```

class<<simpleType>>> of element type convert to simple type "CellularphoneType", as follows:

```
        <xs:simpleType
name="CellularphoneType">
                <xs:restriction base="xs:string">
                        <xs:length value="11"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Element content generated to University.xsd document is as follows：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

Element content generates document must include SharedData.xsd, as follows:

```
        <xs:include
schemaLocation="SharedData.xsd"/>
    <!-- iderfier class convert to a global element
with complexType -->
        <xs:element name="University"
type="University"/>
        <xs:complexType name="University">
                <xs:sequence>
                        <xs:element
name="College"type="College"
maxOccurs="unbounded">
```

The key "Teacherkey" and keyref "Teacherkeyref", "Projectkeyref" of local scope defined is as follows:

```
                                <xs:key
name="Teacherkey">

        <xs:selector xpath="Teacher"/>

        <xs:field xpath="@TeacherID"/>
                                </xs:key>
                                <xs:key
name="Projectkey">

        <xs:selector xpath="Project"/>

        <xs:field xpath="@ProjectID"/>
                                </xs:key>
                                <xs:keyref
name="Teacherkeyref" refer="Teacherkey">

        <xs:selector xpath="Teacher"/>
    <xs:field xpath="SpouseID"/>
                                </xs:keyref>
                                <xs:keyref
name="Projectkeyref" refer="Teacherkey">

        <xs:selector xpath="Project"/>

        <xs:field xpath="PrincipalID"/>
                                </xs:keyref>
                                <xs:keyref
name="Collegekeyref" refer="Teacherkey">

        <xs:selector xpath="."/>

        <xs:field xpath="DeanID"/>
                                </xs:keyref>
                                <xs:keyref
name="PaperofTeacherkeyref" refer="Projectkey">

        <xs:selector
xpath="Teacher/PaperofTeacher"/>

        <xs:field xpath="ProjectID"/>
        </xs:keyref>
                <xs:keyref
name="PaperofGraduatekeyref"
refer="Projectkey">
        <xs:selector
xpath="Teacher/Graduate/PaperofTeacher"/><xs:fi
eld xpath="ProjectID"/>
        </xs:keyref>
        </xs:element>
                </xs:sequence>
      <xs:attribute name="Title" type="xs:string"
use="required"/>
        </xs:complexType>
```

Each class <<complexType>> converts to complex type definition, such as complexType "College", "Project", "Teacher", "Graduate" etc..

```
        <xs:complexType name="College">
                <xs:sequence>
                        <xs:element
name="Title" type="xs:string"/>
                        <xs:element
name="DeanID" type="xs:string" minOccurs="0"/>
                        <xs:element
name="Project"                  type="Project"
maxOccurs="unbounded"/>
                        <xs:element
name="Teacher"                  type="Teacher"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="CollegeID"
type="xs:string" use="required"/>
        </xs:complexType>
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
        <xs:complexType name="Project">
                <xs:sequence>
                        <xs:element
name="Title" type="xs:string"/>
                        <xs:element
name="Fund" type="xs:integer"/>
                        <xs:element
name="PrincipalID"              type="xs:string"
minOccurs="0"/>
                </xs:sequence>
                <xs:attribute   name="ProjectID"
type="xs:string" use="required"/>
        </xs:complexType>
        <xs:complexType name="Teacher">
                <xs:sequence>
                        <xs:element
name="Name" type="xs:string"/>
                        <xs:element
name="Title" type="xs:string"/>
                        <xs:element
name="Major" type="xs:string"/>
                        <xs:element
name="QualityDate" type="xs:date"/>
                        <xs:element
name="SpouseID"              type="xs:string"
minOccurs="0"/>
                                <xs:element
name="PaperofTeacher" type="NewPaperType"
  maxOccurs="unbounded"/>
                        <xs:element
name="Graduate" type="Graduate" minOccurs="0"
  maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute   name="TeacherID"
type="xs:string" use="required"/>
        </xs:complexType>
        <xs:complexType name="Graduate">
                <xs:sequence>
                        <xs:element
name="Name" type="xs:string"/>
                        <xs:element
name="RegisterDate" type="xs:date"/>
```

Here shows choice module group nested in sequence module group.

```
  <xs:choice>
                        <xs:element
name="Telephone"
type="xs:string"maxOccurs="2"/>
                        <xs:element
name="Cellularphone" type="CellularphoneType"
  minOccurs="0"/>
                        <xs:element
name="Email" type="xs:string"/>
                </xs:choice>
                <xs:element
name="PaperofGraduate" type="NewPaperType"
  minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="GraduateID"
type="xs:string" use="required"/>
        </xs:complexType>
```

The following is definition of complex types "NewPaperType" derived refer to the type of element type "PaperType" section.

```
        <xs:complexType              name="
NewPaperType">
        <xs:complexContent>
                        <xs:extension
base="PaperType">
        <xs:sequence>

        <xs:element          name="ProjectID"
type="xs:string"
  minOccurs="0"/>
        </xs:sequence>
         </xs:extension>
        </xs:complexContent>
        </xs:complexType>
  </xs:schema>
```

5. CONCLUSION

Based on the concept of XML model – XUML, this paper defines the mapping rules from extended XUML to Schema, puts forward a kind of algorithm from XUML to XML Schema, the algorithm can effectively realize the conversion of XUML to XML Schema. The next step of research is processing XMI document corresponding to XUML provided by XML Schema Infoset Modle API based the Eclipse platform to generate XML Schema document.

ACKNOWLEDGEMENTS

This work is supported by grants from the Chinese National Natural Science Foundation (No. 60873193) and Natural Science Foundation of Hubei Province of China (No.2011CDC029), Key project in Hubei Provincial Department of Education (No.D20122606), Humanities and social sciences research youth foundation of Ministry of Education of China(No.12YJC630006).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFRENCES:

[1] Ling Feng, Elizabeth Chang, Tharam S. Dillon, "A Semantic Network-Based Design Methodology for XML Documents", ACM Transactions on Information Systems, Vol. 20, No. 4, 2002, pp. 390-421.
[2] Daum Blin, Asset Oriented Modeling(AOM). http://www.aomodeling.org, 2005.
[3] Gillian Dobbie, Wu Xiaoying, Tok Wang Ling, ORA-SS: An Object-Relationship-Attribute Model for Semistructured Data, [Technical Report TR21/00]. National University of Sigapore, 2000.
[4] Bernadette Farias Lósio, Ana Carolina Salgado, Luciano do Rêgo GalvÐo. "Conceptual Modeling of XML Schemas", Proceedings of WIDM03, 2003, pp. 102-105.
[5] David W. Embley, Stephen W. Liddle, Reema A1-Kamha ,"Enterprise Modeling with Conceptual XML", Proceedings of ER 2004 , pp.150-158.
[6] Carlson Din, Modeling XML applications with UML, Wesley Press ,2001.
[7] Hongxing Liu,Yansheng Lu ,Ming Chen, An XML Conceptual Model:XUML, Computer Science, Vol. 34, No. 1, 2007, pp.88-91.
[8] Anneke Kleppe, Jos Warmer,Wim Bast, Parsing of MDA, People's Posts & Telecom Press, 2004.