Research and Development of a XML Modeling Tool

Deng Jia, Liu Hongxing
School of Computer Science and Technology
Wuhan University of Technology,
Wuhan, China
jiadeng@whut.edu.cn, liuhongxing@whut.edu.cn

***Abstract*—The XML schema is used to describe the structure and constraint of XML instance document. The designers of XML are required to proficiently use the schema definition language to write XML schema. In the development of large and medium XML schema, direct writing XML schema is difficult, which entails appropriate design tools to improve the development efficiency. A XML modeling tool dedicated to development of XML schema is designed and implemented based on MetaEdit+ meta-modeling tool. The designer uses this tool to first establish the XML conceptual model, and then transform the XML conceptual model to XML Schema. An instance is used to introduce the application method of this tool. Verification shows that the automatically generated XML Schema document is syntactically correct.**

Keywords*：*XML Conceptual Modeling; XML Modeling Tools; MetaEdit+ Meta-Modeling

I. INTRODUCTION

Currently many efforts have been put into the research on XML conceptual modeling, such as X-Entiry [1], X-CM [2] ,XSEM [3] and XUML [4]etc., which preliminarily formed several categories of XML conceptual models. There are mainly two types in terms of content and form: the extended ER model and extended UML class diagram, yet special XML modeling tool has not been formed. Such meta-modeling tools as MetaEdit+ [5], EMP [6], VMSDK [7] etc. that emerged in recent years provide a good environment for the development of modeling tool. The meta-modeling tool can be used to design modeling language and implement the modeling tool supporting this modeling language [8]. With GOPPRR as meta-modeling language, MetaEdit+ meta-modeling tool is simple and easy to use as it provides flexible domain specific language definition function, integrates manifold graphical user interface tools to customize representation methods, and its meta-model editor adopts dialogue box. This tool provides MERL script language to define generator and model mapping rule to implement the model transformation algorithm. The paper adopts MetaEdit+ to realize XML modeling tool and uses GOPPRR to define XML conceptual modeling language, which is favorable for the users to visually establish a conceptual model. Besides, MERL is used to define the generator to realize self-automatic transformation from conceptual model to logic model, thereby improving the development efficiency of XML schema.

II. MDA BASED XML DESGIN

MDA puts forward four layers of architecture, which are respectively meta-meta model layer (M3), meta-model layer (M2), model layer (M1), and instance layer (M0). The relation between the layers can be regarded as follows: each layer is the instance of its upper layer and the abstraction of its next layer. With the MDA idea as reference, as shown in Fig.1, the XML design and tool research can be concentrated on two layers and in two stages.

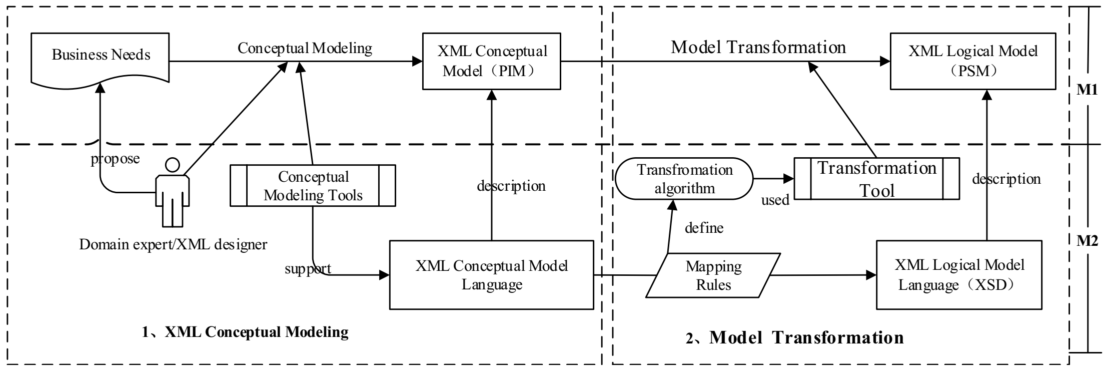


Figure 1: MDA Based XML Design

(1) Two layers: M1 layer, M2 layer

The model-driven XML design is mainly concentrated in M1 layer and M2 layer in MDA system architecture. In M1 layer, the design method of relational data base is drawn as an analogy: Firstly, the designer establishes XML conceptual models according to the actual business demands, then uses tools to automatically generate XML logic model. The establishment of models and transformation between models are defined and realized by M2 layer in fact. In M2 layer, to establish conceptual models and transform the models need support by corresponding tools. The XML conceptual modeling tool supports XML conceptual modeling language to define and describe XML conceptual model. The model transformation tool supports mapping rule and transformation algorithm in M2 layer. The XML logic model is generated via transformation with XML conceptual model as source objective model.

(2) Two stages: XML conceptual modeling and model transformation

XML conceptual modeling: This is the process of establishing XML conceptual model. A XML conceptual modeling language is defined, and a modeling tool supporting this modeling language is established, to visually establish XML conceptual model.

Model transformation: The XML conceptual model is transformed into the XML logic model. The paper adopts XML Schema as the XML logic model. The XML conceptual model is a platform independence conceptual model (PIM), and is on the conceptual layer; the XML Schema is a platform specific logic model (PSM) (specific to XML language, and is described using XML Schema Definition Language), and is on the logic layer. The XML conceptual model and XML Schema are both on different platforms and in different stages. Before model transformation, it is needed to define the mapping rules between XML conceptual model and XML Schema to transform the model via transformation algorithm with the guidance of mapping rules.

The model-driven XML design method separates the business logic and concrete implementation platform from two layers and two stages. The XML conceptual modeling tool and model transformation tool can help the developers to design XML Schema, as the developers no longer need substantive manual coding to develop XML, but can establish XML conceptual model visually, then transform the model to generate XML Schema document. From establishment of conceptual model to generation of XML Schema document, the developers do not need an in-depth knowledge of XML Schema details, the XML Schema generated via inter-model transformation is the result of automatic transformation to a certain extent. This shows that the model-drive design method can effectively improve the development efficiency of XML.

III. Development of XML Modeling Tool

To apply the model-driven XML design method entails appropriate conceptual modeling tools and model transformation tools as support. The conceptual modeling tools help the XML schema designers to design XML conceptual model, while the model transformation tools transform the conceptual model into XML Schema document. In terms of the four-layer model of MDA, the conceptual model belongs to M1 layer, conceptual modeling language belongs to M2 layer and is used to describe the conceptual model of M1 layer, the conceptual modeling tools belong to M2 layer. To define and realize conceptual modeling language and modeling tools of M2 layer, meta-modeling language and meta-modeling tools of M3 layer are needed as support.

A. *XML conceptual modeling language*

XML conceptual modeling language is used to express the modeling language of XML conceptual model (Adopt XUML [4] model), and is established using the GOPPRR meta-meta model belonging to meta-modeling language of M3 layer in the MetaEdit+ meta-modeling tool (i.e. metatypes of Graph, Object, Port, Property, Role Relationship). Following presents, a description of metatypes of Object, Property, Role, Relationship defined in this paper.

1)Create object and property meta-model: The Object Tool and Property Tool are used to create object and property. The object and property description are as shown in Table I.

TABLE I. DEFINITION AND DESCRIPTION OF OBJECT AND PROPERTY

Object	Property	Type	Widget Type	Instructions
Class	Name[Obj]	string	Input Field	The name of Class, which must be unique
	Marks	string	Fixed List	Mark value, used to distinguish representation types of Class
	SubClass	Collection	Item type = Object	Used to define multiple property values of this Class
	SetFacet	Collection	Item type = Object	Used to define constraint of multiple property values of this Class
	Description	Text		Brief introduction of Class, can help the model creator to understand
SubClass	Name[sub]	string	Input Field	The name of subclass
	Type	string	Editable List	Subclass type can be only built-in type or simpleType
	InitialValue	string	Input Field	Initial value of subclass
	MinOccurs	Number		Use to set the minimal occurrence times of subclass
	MaxOccurs	string	Editable List	Use to set the maximal occurrence times of subclass
	IsAttribute	Boolean		Used to set whether the subclass is the main attribute
SetFacet	CTmark	string	Input Field	Corresponding to restriction |list | union value
	Base	string	Input Field	Use to set the name of data type based on
	Facet	Collection	Item type = Object	Set constraint for deriving new types via constraining
Facet	FacetList	string	Fixed List	Set 11 constraints in Schema
	Values	string	Input Field	Set the value of corresponding constraint

There are four Object types of Class, SubClass and SetFacet, Facet. Wherein, the last three types simultaneously serve as the Property type of Object type set to describe the details of Class. The Property of SubClass and SetFacet is the Object type set of Class. SubClass defines multiple Property values and corresponds to the Property or subelement in Schema. SetFacet corresponds to the simple data type in Schema. The Property of Facet is the Object type set of SetFacet to derive new simple data types via constraining.

2)Create Relationship and Role meta-model: The Relationship and Role are created using Relationship Tool and Role Tool. The correspondence between Relationship and Role is as shown in Table II.

TABLE II. THE CORRESPONDENCE BETWEEN RELATIONSHIP AND ROLE

Relationship Name	Role From Name	Role To Name
Association	Association From	Association To
Reference	Reference From	Reference To
Aggregation	Aggregation From	Aggregation To

A Relationship corresponds to two Roles and jointly expresses a connection relationship, and expression can be formalized into: .A~Role$_A$>Relationship~Role$_B$.B

.A and .B represent the instances of two different Class building blocks in the meta-model. ~ Role$_A$ and ~ Role$_B$ respectively represent different Roles of the two instances .A and .B in the relationship. >Relationship represents the relationship classification of Objects A and B. If .A and.B are the same Class instance, this means they have a relationship relation in themselves.

Association Relationship is used to describe the data-relation association between the building blocks of meta-model, and is mainly embodied in the reference relationship of data items between elements in systems such as XML. It uses key and keyref for constraining, which is similar to foreign key constraint in database. It is a one-way connection relationship.

Reference Relationship is used to describe the data type reference relation between building blocks of meta-model, and is mainly embodied in the relation of reference of data type by the elements in systems such as XML. For the simpleTpye and complexType that need multiple references, the base Class can be defined before using reference relationship for extension definition or direct reference.

Aggregation: The generalized aggregation relationship is used to describe the inclusion relation between building blocks of meta-model, and is mainly embodied in the inclusion semantic meaning between elements in system such as XML.

The Class, Relationship and Role are expressed in graph visualized, i.e. defining the graphic symbols of modeling elements in the model diagram, as shown in table III. The visualization of property is bound in Class.

TABLE III. THE GRAPHIC SYMBOLS OF MODELING ELEMENTS

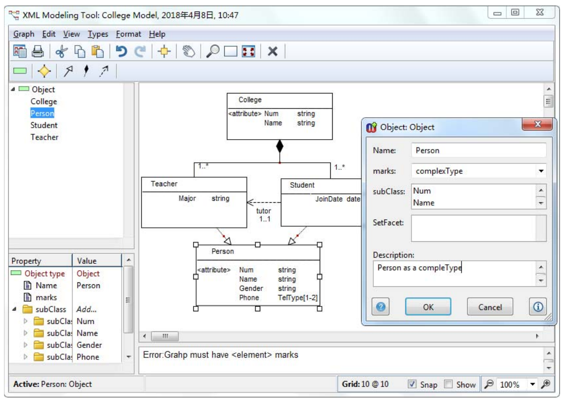
Modeling element	Class	Reference	Aggregation	Association
Graphical representatio	▭	┈┈┈▷	——◆	- - - ->

B. *XML conceptual modeling tool*

To establish XML conceptual model for the language that supports XML conceptual modeling needs the modeling tools supporting this language. The XML conceptual modeling language is set to Graph of MetaEdit+ via "Bindings", and multiple defined generators are written into this Graph, then the XML modeling tool is implemented, and saved in two formats of .met or .mxm. In using this tool, the Import function can be used to import the files in the above two formats to the MetaEdit+ tool to import XML conceptual modeling too. To create an instance of a graph can create a XML conceptual model, and to create instances of multiple Graphs can create multiple XML conceptual models. Fig.2 shows the structural information model of an institute's personnel expressed by this tool.

C. *XML model transformation tool*

In MetaEdit+, the model transformation tool is implemented by defining manifold generators. The definition of generator is implemented via writing of MERL code by generator editor. The model transformation generator in this paper is the main generator named as "ModelTransformation", which contains six subgenerators. The structural diagram is as shown in Fig.3.



Property	Value
Object type	Object
Name	Person
marks	
subClass	Add...
subCla:	Num
subCla:	Name
subCla:	Gender
subCla:	Phone

Name: Person
marks: complexType
subClass: Num Name
SetFacet:
Description: Person as a compleType

Error:Grahp must have <element> marks

Figure 2: The XML conceptual model with "institute" as module

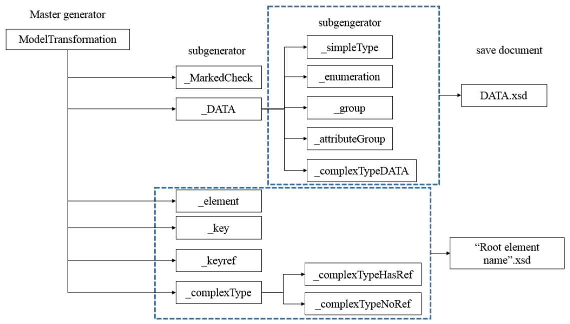


Figure 3： The structural diagram of "ModelTransfromation" main generator

The subgenerator "_MarkedCheck" is used to detect the effectiveness of model to guarantee the transformed XML Schema document is effective. Subgenerator "_DATA" is used to transform data object and save them as "DATA.xsd". It also includes five subgenerators, whose functions are to realize transformation according to the difference of mark value. The subgenerator "_element" is used to realize transformation of root element. The subgenerators "_key" and "_keyref" are used to define the scope and uniqueness of main property between associated objects. The subgenerator "_complexType" is used to transform content object. It also includes two subgenerators, whose largest difference is whether they extend or constrain the object of referenced data type, the corresponding mapping rules and conversion algorithms are different either. The content transformed by the subgenerators "_element", "_key", "_keyref" and "_complextype" will be written into the .xsd document named after the name of corresponding Class in "_element".

The above describes the basic corresponding transformation functions of multiple generators. The subgenerator "_complexTypeHasRef" realizes the mark Class of <complexType>. This Class can possess manifold relationships. When it possesses the association relationship, the main property of the associated end needs to be taken as an " foreign key" to be transformed into own subelement. When it possesses the generalized aggregation relationship, it is needed to take the Class of part end as one of its subelement to form a parent and child element relation. When it possesses the reference relation, it is needed to take the referenced Class as the base Class of own data type to define constrained derivation or extended derivation. Following will describe a relatively key transformation algorithm in this paper. The transformation algorithm for subgenerator "_complexTypeHasRef" is as follows:

_complexTypeHasRef()

```
foreach .Class            //Ergodic Class
 if :marks='complexType' then        //Judgment mark
   if Reference!='' //Whether there is a reference relationship
     base=name //The Class name of reference serves as the base Class
   end
   if Association!=''   //Whether there is an association relationship
      //Add the main property of the associated end as a new element
      element=Object().subclass().attribute()
   end
   if Aggregation!=''   //Whether there is an aggregation relationship
      //Add the Class name of the part end as a new element
      element=Object().name
   end
   subClass()     //Property of self Class and element output
 end
```

The generator is imported into the XML conceptual modeling tool to generate XML modeling tool. Now the tool is provided with two major functions of conceptual modeling and model transformation.

IV. Application Case Analysis on the Tool

The establishment of conceptual model is based on the actual business demands. This case is based on an institute's personnel structural information to create a conceptual model chart of the basic information of the institute, as shown in Fig.2.

The marks of the conceptual model are defined. Reference [9] presents the UML extension definition, based on which, the marks of XML conceptual model is defined, and the "Marks" generator is used to automatically mark the model. One mark corresponds to one concept in PSM model. The application of one mark by one Class in PIM model just designates the transformation mode of this Class. The marked XML conceptual model is as shown in Fig.4. After running the "ModelTransformation" generator, the marked conceptual model generates two XML Schema documents of DADA.xsd and College.xsd, as shown in Fig.5

V. Conclusion

To make up for the drawbacks of special XML modeling tools available in the market to facilitate the exchange and communication between the domain experts and XML designers and improve the development efficiency of XML, the paper defines a XML conceptual modeling language and implements a graphical modeling tool supporting this language based on the MetaEdit+ meta-modeling tool to help the developers to establish XML conceptual model and transform models. Lastly, the whole process of model-driven XML schema design method is introduced via a case. The research findings of the paper are favorable for the developers to improve the efficiency and effect of XML design and development, and can help the domain experts to establish the XML Schema document of own domains via visualized modeling tools.

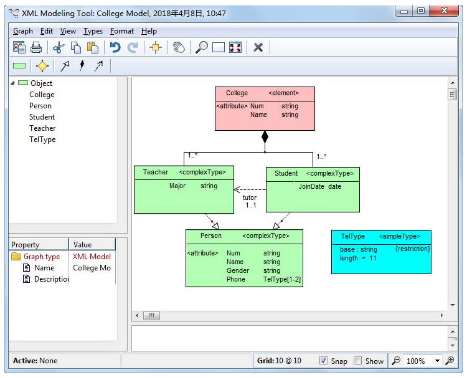
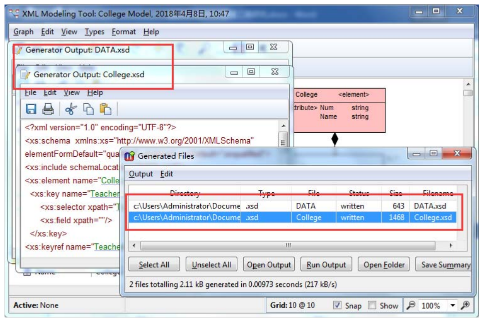


Figure 4: Marked XML conceptual model



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qua
<xs:include schemaLocat
<xs:element name="Colle
  <xs:key name="Teacher
     <xs:selector xpath="T
     <xs:field xpath=""/>
  </xs:key>
<xs:keyref name="Teache
```

Directory	Type	File	Status	Size	Filename
c:\Users\Administrator\Docume	.xsd	DATA	written	643	DATA.xsd
c:\Users\Administrator\Docume	.xsd	College	written	1468	College.xsd

2 files totalling 2.11 kB generated in 0.00973 seconds (217 kB/s)

Figure 5: Generated XML Schema document

References

[1] Lóscio B F, S algado A C , Galvão L R. Conceptual Modeling of XML S chemas. Proceeding s of W IDM03, ACM Press , 2003.pp. 102-105

[2] Chin S M, Haw S C, Lee C S. X-CM: A Conceptual Modeling for XML Databases ICCCM: Proceedings of International Conference on Computer Communication and Management, 2011.pp.59-63

[3] Nečaský M. Conceptual Modeling for XML, volume 99 of Dissertations in Database and Information Systems Series. IOS Press/AKA Verlag, January 2009. pp.527-549

[4] Liu H X, Lu Y S, Chen M. An XML conceptual modeling:XUML. Computer Science, Vol. 34, No. 1, January 2007, pp.88-91.

[5] MetaCase. MetaEdit+ Verion5.0 User's Guide. http://www.metacase.com/support/50/manuals/meplus/Mp.html Accessed 28 September 2017.

[6] Gronback R. Eclipse Modeling Project Addison-Wesley Professional, 2009.

[7] Cook S, Jones G, Kent S, et al. Domain-specific Development with Visual Studio DSL Tools. Addison-Wesley Professional, 2007.

[8] LIU H,MA Z Y, SHAO W Z.Progress of Research on Metamodeling. Journal of Software, Vol.19, No.6, June 2008, pp.1317−1327

[9] Carlson Din, Modeling XML applications with UML, Wesley Press ,2001.