XML Conceptual Modeling with XUML

HongXing Liu
HuaZhong University of Science and Technology
P. R. China
86-27-86581376
liuhx@public.wh.hb.cn

YanSheng Lu
HuaZhong University of Science and Technology
P. R. China
86-27-87556601
lys@mail.hust.edu.cn

Qing Yang
Wuhan University of Technology
P. R. China
86-27-86568023
qingyang@public.wh.hb.cn

ABSTRACT

As XML has become the standard format for representing structured and semi-structured data on the Web, the methods for designing XML schemas is becoming more and more important. XML schemas represent the logical models of the documents. In order to design or integrate XML schemas, it is necessary to first design the conceptual structures with a proper conceptual model. We thus specify a XML conceptual model, XUML, which has following characteristics comparing with the existing XML conceptual models: 1) expressing the containment semantics more explicitly; 2) supporting the concept of Business Components; 3) specifying the data dependencies in multiple contexts. XUML is defined based on the UML2 standard, so it will be more friendly and practical for those who have had knowledge and experience on UML. Based on XUML, we further present a framework of the methodology which is dedicated to the design of XML documents and XML databases. In this paper, the focus is put on XUML.

Categories and Subject Descriptors

H.2.1 [**Database Management**]: Logical Design—*Data models; normal forms;* D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Computer-aided software engineering*

General Terms

Design, Languages

Keywords

XML, Conceptual Model, XML Schema, UML2

1. INTRODUCTION

XML has become the de-facto standard for representing and exchanging data among various applications and databases over the Internet. The Document Type Definition (DTD) or W3C XML Schema definition language (WXS) can be used to define the schema which describes the syntax and structure of XML documents. However, it represents the logical model rather than the conceptual model, so is hard to express the semantics that underlies these documents. To design XML documents or XML database, it is necessary to describe and model real-world data semantics and their complex interrelationships by using suitable conceptual models [1].

ICSE'06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

Several XML conceptual models, such as Semantic Network [1], AOM [2] and ORA-SS [3], have been presented recently. X-Entity [4] and C-XML [5] support XML by extending the ER model. [6] defines special profiles based on UML1.x to accommodate WXS. These models and modeling methods can capture richer semantics and constraints than what XML schemas could; therefore improve the design of XML.

In this paper, we present a XML conceptual model, XUML, which inherits the merits of those mentioned models, and has major improvement in following aspects: 1) can express the containment semantics more explicitly; 2) support the concept of Business Components; 3) can specify the data dependencies in multiple contexts. These improvements make XUML more expressive, precise and understandable.

The main purpose of XUML is to support the design of XML documents and XML databases, and support the information integration of XML. These are very important in large-scale electronic commerce environments and the content management systems. XUML is based on the UML2 standard so it will be more friendly and practical for those who have had knowledge and experience on UML.

The rest of the paper is organized as follows. Section 2 introduces the XUML and demonstrates its characteristics and advantages. Section 3 describes a XUML-based XML design methodology, and outlines a CASE environment which will support XUML modeling. Section 4 presents some concluding remarks.

2. XUML

The characteristics of XUML can be outlined as follows: 1) XUML is firstly a conceptual model, so its primary purpose is to describe the conceptual structures and semantics of the application domains. 2) XUML is based on UML2; XUML metamodel is defined by using the standard profile mechanisms, including stereotypes, tagged values and constraints. 3) It can support all the data types defined in WXS. 4) It can model three kinds of orderings which are important in XML 5) It introduces two key constructs, i.e., the generic Aggregation and the Business Component, which make the conceptual modeling of XML more rigorous and accurate than the existing XML conceptual models. 6) It can specify the data dependencies in multiple contexts. 5) and 6) are the most important characteristics of XUML, so they are explained with more details.

2.1 Generic Aggregation Relationships

UML (1.x, 2.0) requires an aggregation relationship to be binary only [7]. But in many cases, especially in XML, a relationship is defined among more than two participants, the invariant that

defines the relationship refer to all its participants rather than just to two of them, so several binary relationships are not equivalent to one non-binary. In Figure 1, there are three binary aggregations for the same whole (Order), but each is independent. While in Figure 2, there is only one non-binary generic Aggregation, the "whole" is related to a "collection of parts". Based on [8], a generic Aggregation is defined in XUML as a first-class relationship type.

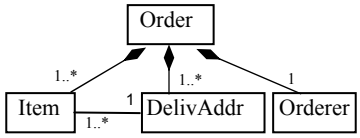

Figure 1. Three binary aggregations

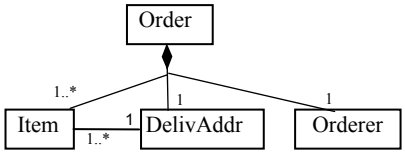

Figure 2. A non-binary aggregation

Definition 1: For an **(generic) Aggregation**, the properties of one of the participants ("whole") are determined, in part, by the properties of the other participants ("parts."). An aggregate type corresponds to one or more part types, and an aggregate instance corresponds to zero or more instances of each part type.

The Aggregation is more general than the usual binary aggregation defined in base UML, so we call it "generic". It is also different from the (symmetric) n-ary association in base UML because it is asymmetric.

A binary Aggregation uses the traditional UML aggregation diamond notation. When the Aggregation is non-binary, a new notation is used as shown in Figure 2. Note that this notation is different from the tree form of notation in base UML which is for multiple binary aggregations to the same whole.

The Aggregations can be shared (non-hierarchical) or composite (hierarchical). We take the composite Aggregation as the default (the diamond is black), because it describes the most important relationship in XML, i.e., the hierarchical relationship between an element and its child elements.

The Aggregation in Figure 2 has just two levels. But the Aggregation can be nested and form a multilevel Aggregation hierarchy. Aggregation is transitive.

2.2 Business Components and Structured Classes

A component is a modular part of a system; now it is a logical concept in UML2. A **business component** (BC) denotes a unit that is an object or a group of objects and exists in its own right. A BC can be a **business object** (BO) that plays a role in the business process, or a **business document** (BD) that is exchanged as a message between BOs in the context of a business process [2].

Traditional conceptual modeling approaches, such as ER, tend to model the real world as a collection of basic, minimal entities (normal form entities) and the relationships among them, so a normalized relational database schema can be easily produced from the ER model. But the concept of BC is often blurry in traditional conceptual models. While in XML an instance documents usually records and describes a BC, and the schema of the document specifies the BC's logic structures. So the BC concept is more important in XML applications than in usual databases. The concept of BC is introduced into XUML and treated as the first-class modeling constructs.

We can model a BC by using the generic Aggregation hierarchy (see Figure 2) or the Structured Class which is a new construct introduced in UML2 [7]. A **structured class** is a classifier with internal structure. It contains a set of parts connected by connectors. A **part** has a type and a multiplicity within its container. All of the objects in a single structured object are implicitly related by their containment in the same object. A **connector** is a contextual relationship between two parts in structured class. It defines a relationship between objects serving as parts of the same structured object. A connector between two parts of a structured class is different from an association between two classes that are associated by composition to the same class. Note that the connector between *Item* and *DelivAddr* in Figure 2 is different from the association between *Item* and *DelivAddr* in Figure 1.

Figure 3 shows an example of structured class, which expresses the same meaning as in Figure 2. We can use either the generic Aggregations or the structured classes.

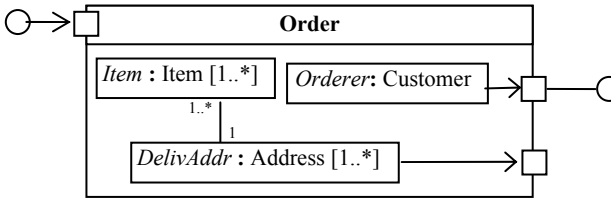

Figure 3. A structured class, its parts and ports

A structured class may be tightly encapsulated by forcing all interactions between the external environment and the internal parts to pass through ports. A **port** is an interaction point with a well-defined interface; so it is useful for defining the queries and operations on XML documents, or defining the references to other BCs. There is one **provided interface** and two **required interfaces** in Figure 3.

2.3 Data Dependencies in Context

The data dependencies, especially the functional dependencies, are most important semantics. In traditional data modeling, the data dependencies are analyzed and specified in the context of a "flatten" entity or class. Because of the hierarchical structure, the data dependencies in a XML BC are much more complicated, so need to be specified in multi-levels contexts.

A structured class is a container, a namespace, and can be nested. So it is also a very suitable context to specify the data dependencies in XML.

Example1: In Figure 1, the business concept of Order is divided into four concepts, and which are all at the same (conceptual) level. So the business rule *that an Item (of goods) be delivered to one address* can be specified as a functional dependency as:

Item (orderID1, itemID) → *DelivAddr (orderID2, deliverAddr), where orderID1 = orderID2.*

While in Figure 2 or Figure 3, because the Order has been modeled as a structured class and it has built a definite context, the functional dependency now can be simply specified as:

Order: (Item → DelivAddr)

3. A XML DESIGN METHODOLOGY BASED ON XUML

There are two kinds of XML schemas: 1) single XML document schema; 2) XML database schema, i.e., a collection of XML schemas; these schemas relate to each other and specify the schema of a XML database. The XUML-based design methods support the modeling of these two kinds of schemas.

3.1 Design Process

The proposed methodology is based on two design levels: a **conceptual level** and a **logical level**, as shown in Figure 4. Here ECM, XCM, and XLM stands respectively for Enterprise Conceptual Model, XML Conceptual Model, and XML Logical Model.

We first present a way to semantically model XML using XUML. A XCM can be designed from scratch, or transformed from an existed ECM which is a UML model. We then examine the mapping from the XCM to the corresponding XLM (XML schema or XDB schema), which is specified by using WXS and mainly concerned with detailed XML element/attribute declarations and simple/complex type definitions.

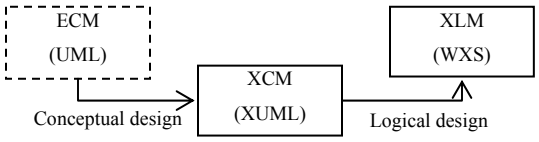

Figure 4. The two-level design process

3.2 Conceptual Design

3.2.1 From UML to XUML

In our methodology, we consider a system as a System-Components-Objects hierarchy. Usually an independent BC corresponds to an independent document schema, while interrelated BCs correspond to the schema of XML database. So we take the modeling of BCs as the key task.

The basic principle: tight cohesion within a component, loose coupling between components.

In a UML class diagram, a BC involves one class or several classes with associations. The designers designate a class, called identifying class, as the root of the BC, and then use the **UML2XUML** arithmetic to generate a XUML model diagram.

Arithmetic: UML2XUML

Input: a BC_{uml} model (one class or several classes with associations, a root class)

Output: a BC_{xuml} model, it is a hierarchical view of the BC_{uml}.

Example2: the class diagram (Figure 5) is the conceptual model of a book store. One possible corresponding XUML diagram is as Figure 6. Three BCs are identified in Figure 6. Look at the BC rooted by *Order*, the main inner relationship is represented by the generic Aggregation which naturally describes the nested hierarchy of the BC; beside the generic Aggregation there is an inner association between *Item* and *DelivAddr*. The class *orderer* is built as an agent of *Customer* in another BC. The reference relationships between BCs should be loose, and they will be implemented as XPointer or XLink late in XML.

3.2.2 XUML Normalization

To get a XML schema with good properties, one possible approach is to use the normal form theory such as the XNF [9] in logical design phase. But XNF is not intuitional so is hard to analyze and apply. Our approach to the good schemas is to normalize the structure in conceptual design phase. It is easier relatively for a schema designer.

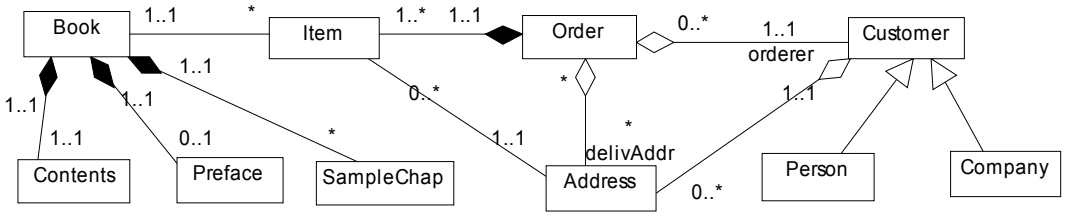

Figure 5. The UML model of a book store

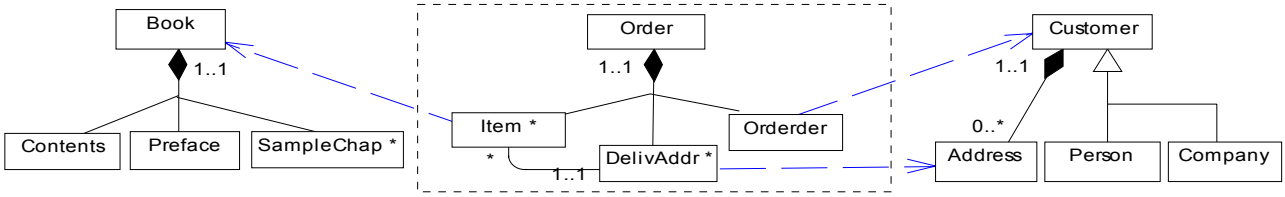

Figure 6. A possible XUML model corresponding to Figure 5

The characters of XUML Normal Forms include:

1) Relationships from the parent to children are 1:1 or 1:n;

2) All the classes in a component are in PNF [2];

3) There are no connection traps [10]. In Figure 7(a), if an employee does not belong to any department, then his information cannot be stored. Similarly in Figure 7(b), if a department has no employee, then its information will be lost. These are examples of the traps.

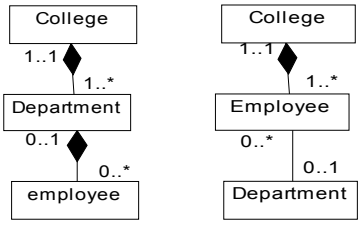

Figure 7. Connection traps

3.3 Logical Design

When a XUML conceptual model has been designed well, the logical design is relatively simple. It can be divided into two steps:

1) After choosing a XML schema language (here the WXS), the designers can mark some special objects in the XUML model using stereotype and tagged values, while most of the objects are marked automatically by the default rules.

2) The **XUML2XSD** Arithmetic is applied to automatically transform the XUML model to a XML schema specification (XSD), according to a set of mapping rules.

Arithmetic: XUML2XSD
Input: a BC_{xuml} , with optional marks;
Output: A XML schema specification.

3.4 CASE Environment

We are planning to design and implement a set of tools, which constitute a Computer Aided Software Engineering (CASE) Environment together with other necessary tools. The environment is based on the XUML metamodel.

The main modules of the environment are as follows:

- XUML Editor, it will be implemented based on an existing UML tool which supports UML profile well;
- UML2XUML transformer;
- XUML2XSD transformer.

The Eclipse techniques [11] will be incorporated as the base supporting platform which can integrate all the modules.

4. CONCLUSION

This paper presented XUML, a UML2-based conceptual model for XML. It provides clearer description for XML schemas by hiding implementation details and focusing on semantically relevant concepts. The most distinctive characteristic of XUML is that it supports the generic (asymmetric) Aggregation relationship representation. This characteristic is very important for XML conceptual modeling. And to the best of our knowledge, it has not been demonstrated by other XML conceptual model. In XUML models, the containment semantics and Business Components could be explicitly expressed, and the data dependencies in multiple contexts could be specified. The XUML metamodel is an extension to UML2 metamodel, so inherits its power. Especially, the new model elements in UML2, such as structured class, parts, connectors and ports, are incorporated into XUML. Our previous studies show that XUML is a suitable conceptual model for XML.

We intend to extend the research in three directions. Firstly, we are specifying the XUML meta-model formally; secondly, we are planning to implement a XUML-based CASE tools; lastly, we will further develop and evaluate the XUML-Based methodology.

5. ACKNOWLEDGMENTS

This research is supported by the Natural Science Foundation of Hubei, China, (No. 2004ABA040).

6. REFERENCES

[1] Ling Feng, Elizabeth Chang, Tharam Dillon. A Semantic Network-Based Design Methodology for XML Documents. *ACM Transactions on Information Systems*, Vol. 20, No. 4, October 2002, 390–421.

[2] Berthold Daum. Asset Oriented Modeling (AOM). http://www.aomodeling.org. 2005.

[3] Dobbie G, Wu X, Ling T W, Lee M L. *ORA-SS: An Object-Relationship-Attribute Model for Semistructured Data*, Technical Report TR21/00. National University of Singapore, 2000.

[4] Bernadette Farias Lóscio, Ana Carolina Salgado, Luciano do Rêgo Galvão. Conceptual Modeling of XML Schemas[C]. *Proceedings of the fifth ACM international workshop on Web information and data management (WIDM03)*, 102-105, ACM Press, 2003.

[5] Embley D W, Liddle S W, Reema Al-Kamha. Enterprise Modeling with Conceptual XML. *Proceedings of ER 2004*: 150-165, LNCS 3288, Springer-Verlag, 2004.

[6] David Carlson. *Modeling XML Applications with UML*. Addison-Wesley Professional, 2001.

[7] Rumbaugh J, Jacobson I, and Booch G. *The Unified Modeling Language Reference Manual* (2nd Edition). Addison-Wesley, Reading, MA, 2004.

[8] OMG. *UML Profile for Relationships Specification*. http://www.omg.org/cgi-bin/doc?formal/2004-02-07.

[9] Marcelo Arenas, Leonid Libkin. A Normal Form for XML Documents. *Proceedings of PODS2002*: 85-96. June 3-5, Madison, Wisconsin, USA. ACM 2002

[10] D. R. Howe. *Data Analysis for Data Base Design*. Edward Arnold Ltd, London, 1983.

[11] Eclipse. *Eclipse Platform Technical Overview.* http://www.eclipse.org/whitepapers/eclipse-overview.pdf, 2003-2-12.