第29卷 第8期 计 算 机 工 程 2003年5月
Vol.29 № 8 Computer Engineering May 2003
· 开发研究与设计技术 · 文章编号：1000—3428(2003)08—0195—03 文献标识码：A 中图分类号： TP311

基于UML的XML建模方法

张 志，赵文耘，李 川
(复旦大学计算机与信息技术系软件工程实验室，上海 200433)

摘 要：针对由业务模型直接定义XML Schema存在较多的困难，提出了一种把UML技术用于XML Schema建模的方法。该方法通过领域建模定义了3层模型，即概念层模型、逻辑层模型和实现层模型，分别对应于业务模型的分析、设计及实现阶段，并对各层模型之间的转换方法进行了有效的探讨。

关键词：可扩展标记语言；统一建模语言；模式；建模

UML-based XML Modeling Approach

ZHANG Zhi，ZHAO Wenyun，LI Chuan
(Department of Computer and Information Technology,Fudan University,Shanghai 200433)

【**Abstract**】As there are some difficulties in defining XML Schema according to business model directly, this paper presents a method to apply UML to XML Schema modeling. This method defines three levels of model by domain modeling. These three levels are conceptual level, logical level and physical level. They correspond to the analysis, design, and implementation of the business model individually. And this paper discusses the mapping method between models of different levels effectively.

【**Key words**】XML；UML；Schema；Modeling

XML正在快速成为互联网上数据交换的标准。2001年5月2日，XML Schema正式成为W3C的推荐标准。与传统的DTD相比，Schema在描述XML文档结构方面功能更为强大，提供了更为丰富的数据类型，可以对数据结构提供更多的约束，对名称空间提供了更多的支持，而且XML Schema本身也符合XML的语法。因此XML Schema必将取代DTD而成为定义和验证XML文档的标准。但与此同时，一些人也认为XML Schema过于复杂，他们发现很难直接根据业务模型定义出正确的XML Schema，而且在用XML Schema表示出业务模型后，很难与用户和业务伙伴进行交流。

针对上面这个问题，有必要开发一种针对XML Schema的建模方法。我们可以把XML Schema看作是业务模型的具体实现，而从与用户交流的角度来说，其它一些模型表示方法可能会比XML Schema更有效。而这其中UML是一种比较好的选择。

UML是一种功能强大的图形化建模语言，它可以用来为面向对象系统建模、描述系统架构和业务过程。用UML表示的产品易于理解，便于不同知识背景的客户和系统分析、设计、开发人员的交流，有利于产品的推广，也易于自我扩展。UML的表示能力和直观的可视化形式要强于XML当前的表示能力。在此基础上，我们把UML作为建模语言，把XML作为数据表示语言，定义了一系列从UML到XML的映射方法。

1 XML建模方法

由于难以从业务模型直接定义出XML Schema，因此参考软件系统的设计开发过程，引入了3层设计模式，分别对应于业务模型的需求、分析、设计的过程。因为建模过程是一个由无形到有形，由粗到细的循序渐进的过程，所以把模型分为3个层次，即概念层模型、逻辑层模型和实现层模型。概念层模型是用标准的UML类图来表示的；逻辑层模型也是UML类图，不过在概念层模型的基础上引入了一些构造型；实现层模型就是最终生成的XML Schema。

为了更清晰地给出3层定义的不同，本文采用一个实例说明3层设计模式，描述图书馆中的一次借书记录(Transaction)：一个借书者(教师或学生)一次可借阅若干本书。

1.1 XML Schema的概念层模型

在定义概念层模型阶段，通过精化和组织需求捕获阶段所描述的需求(用UML用例图描述)来对其进行分析，把整个系统划分为易于管理的各个部分，充分考虑复用可能性。

由于教师和学生都可以作为借书者，且他们的部分属性是相同的，因此可以把这部分相同的属性放在一个类person中，把教师和学生作为这个类的子类。把Transaction分为两部分来定义：一部分是核心的transaction类型；另一部分是person类型。在UML中，这两部分定义可以作为两个包 (package)。图1显示了transaction的概念层模型。

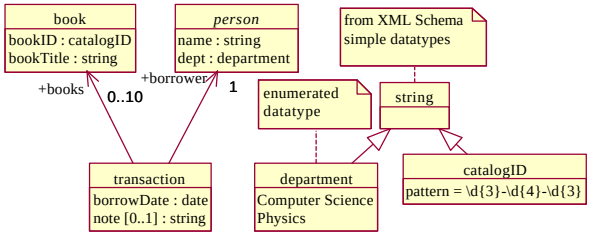

图1 transaction的概念层模型

概念层模型是用标准的UML类图来表示的，并通过加入一些注释来对数据类型、约束条件等进行说明。为了便于映射到XML Schema，约定类图中属性的数据类型使用的均是XML Schema内建的简单数据类型。因为需要的只是数据模型，而不是对象模型，所以只需定义静态类型，而不用考虑类中的操作。

1.2 XML Schema的逻辑层模型

作者简介：张 志(1976－)，男，硕士生，研究方向：软件工程；赵文耘，教授；李 川，硕士生

收稿日期：2002-06-19

通过已经建立的概念层模型，可以与领域专家、用户进行交流，以验证其正确性。本节给出了把该模型转换为逻辑层模型的方法。由于在从UML到XML Schema的映射过程中，可能会面临很多种不同的选择，不同的人按不同的方法生成的XML Schema可能会不一样。例如：UML中属性和关联应该映射为XML Schema中的属性还是元素；UML中类和数据类型的泛化关系怎样映射到XML Schema的定义。

因此，为了解决在从UML到XML Schema的映射过程中的这些问题，使用了UML的扩展机制，分别为构造型(stereotype)、标记值(tagged value)和约束(constraint)3种。这里，主要使用了构造型和标记值。例如，可以用构造型<<simpleType>>来表示这个类应该被映射为用户定义的简单数据类型。可以使用{modelGroup}来定义一个类中的属性及关联在被映射为XML Schema的元素后应该使用哪种模型组，all、choice还是sequence。定义的几种构造型如表1。

表1 构造型

构造型名称	作用于	意义
<<complexType>>	类	映射为Schema的复合类型
<<simpleType>>	类	映射为Schema的简单类型
<<element>>	属性或关联	映射为Schema的元素
<<attribute>>	属性或关联	映射为Schema的属性
<<facet>>	属性	使用Schema的刻面
<<XSDsimpleType>>	类	Schema内建的简单类型

在一个UML模型中，可以为一些所使用的扩展机制设置默认值。例如，可以把{modelGroup}的默认值设为sequence，这样，就不必为每一个类分别设置这个值。在下面的例子中，把UML类属性缺省的映射为XML Schema的元素，需要映射为XML Schema的属性的时候，使用构造型<<attribute>>加以说明。图2显示了transaction的逻辑层模型。

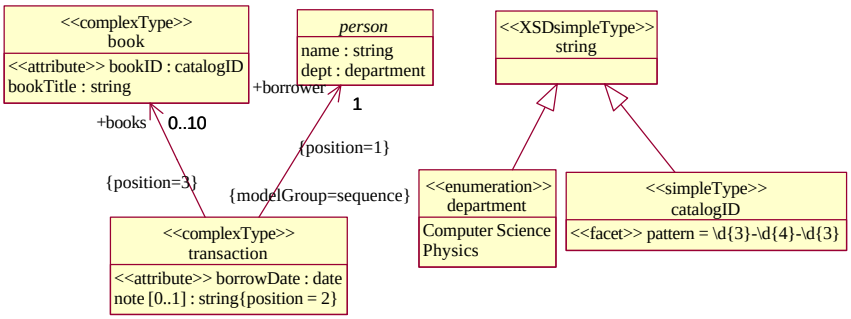

图2 transaction的逻辑层模型

1.3 XML Schema的实现层模型

在得到了逻辑层模型后，就可以根据它来生成最终的实现层模型，也就是XML Schema。下面定义了一种从逻辑层模型到XML Schema的映射方法。

1.3.1 类和属性的映射方法

在UML中一个类定义了一种复杂的数据结构，它应该缺省地映射为XML Schema中的complexType。这一点在图2中已经用构造型<<complexType>>表示出来了。transaction类和它的属性被映射为如下的XML Schema定义：

```
<xsd:complexType name="transactionType">
<xsd:sequence>
<xsd:element name="note" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="borrowDate" type="xsd:date/">
</xsd:complexType>
```

UML中的属性的多重性在XML Schema中用minOccurs和maxOccurs表示。我们为XML Schema中每一个complexType缺省的生成一个顶层元素。transaction的顶层元素如下：

```
<xsd:element name=transaction type=transactionType/>
```

1.3.2 关联的映射方法

图2的transaction类不仅包括属性，它还包括2个关联，borrower和books。每一个关联都有一个角色名以及多重性说明。这些关联被映射为complexType中的元素，与从UML类的属性映射而来的元素放在一起。

```
<xsd:complexType name="transaction">
 <xsd:sequence>
   <xsd:element name="borrower" type="person"/>
…
</xsd:sequence>
<xsd:attribute name="borrowDate" type="xsd:date"/>
</xsd:complexType>
```

为了表示各个元素在这个序列中的顺序，在UML中使用了标记值。

1.3.3 用户定义数据类型的映射方法

在图2中，我们定义了catalogID数据类型。在缺省情况下，用户定义数据类型将被映射为XML Schema中的复合类型。但在这个例子中在这个类型的定义中使用了构造型<<XSDsimpleType>>，所以在映射时将把它们映射为XML Schema中的简单类型：

```
<xsd:simpleType name="catalogID" base="xsd:string">
        <xsd:pattern  value="\d{3}-\d{4}-\d{3}"/>
    </xsd:simplType>
```

为了对catalogID类型加以限制，可以使用XML Schema中的刻面(facet)。为了反映这一点，在图2的catalogID中加入了属性pattern，它将被映射为XML Schema简单类型的刻面Pattern。

1.3.4 泛化的映射方法

面向对象分析和设计中的一个重要概念就是泛化，子类可以从它的父类那里继承属性和关联。在person类型的定义中，teacher是person的子类。根据缺省规则，把person和teacher映射为XML Schema的复合类型：

```
<xsd:element name="person" type="person" abstract="true"/>
<xsd:complexType name="person" abstract="true">
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="dept" type="department"/>
</xsd:complexType>
<xsd:complexType name="teacher">
<complexContent>
<extension base="person">
  <xsd:element name="teacherNo" type="workNo">
  <xsd:element name="teacherTitle" type="title">
```

</extension>
</complexContent>
</xsd:complexType>

在person的顶层元素和复合类型定义中包含了一个属性abstract="true"，这说明它是从一个抽象类映射而来的。复合类型teacher是从person扩展而来，这相当于UML中的继承关系。由于person的属性abstract="true"，因此在borrower元素中不能使用person类型自身，而应该使用从它扩展而来的teacher或student。

1.3.5 枚举类型的映射方法

person中的dept元素引用了简单数据类型定义department。department在UML中是一个枚举类型，我们给它加了一个构造型<<enumeration>>来表明在映射为XML Schema时将生成一个enumeration刻面：

```
<xsd:simpleType name="department" base="xsd:string">
<xsd:enumeration value="Computer Science"/>
<xsd:enumeration value= "Physics"/>
</xsd:simpleType>
```

2 Schema的模块化及复用

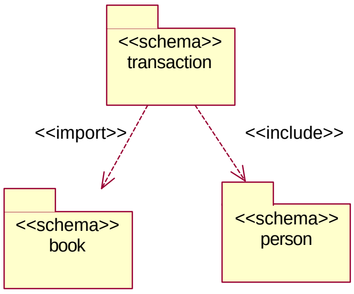

图3 Schema模块依赖关系的UML包图

在为XML Schema建模的过程中使用UML的一个好处是它有利于在XML应用中实现Schema的模块化及复用。在前面的例子中，transaction和person分别在两张UML类图中进行了描述，可以把它们看作两个模块，并且在必要时可以重用这些模块。例如，如果另一个XML Schema中也需要使用person类型，就没有必要重新定义它，只需要简单地引用已有的person模块就可以了。甚至可以自定义person的新的子类。UML中用包来表示各个模块以及它们之间的依赖关系。如果把book的类型声明也作为一个单独的模块，并使用与transaction和person不同的目标名称空间(targetNamespace)，那么可以用图3表示这3个模块之间的关系。

在映射为XML Schema的时候，每一个包都生成一个单独的schema文件。不同包之间的依赖关系在XML Schema中用<include>或<import>来表示。到底是使用<include>还是使用<import>依赖于相关包的目标名称空间是否相同，当它们的目标名称空间相同时使用<include>，否则使用<import>。

3 结语

为了解决由业务模型直接导出XML Schema的困难，我们将UML技术引入到XML Schema的建模过程中，提出了3层设计模式，避免一开始就要考虑底层实现上的问题，便于与领域专家、用户以及开发人员进行交流，在获得正确的领域模型后再生成XML Schema，这样就能提高XML Schema建模的准确性以及效率。在一个大型系统中，使用XML的部分可能仅是其中的一部分，如果这部分和整个系统的其它部分一样都用UML建模，将有助于整个系统的集成。并且用UML定义的模型将有利于XML Schema的模块化和复用。在以后的工作中，可以定义一种从XML Schema到UML的逆向工程的方法，这将有助于目前已经存在的大量XML Schema的复用。

参考文献

1 Fallside D C.XML Schema Part 0: Primer.W3C Recommendation, http://www.w3.org/TR/xmlschema-0/,2001-05-02
2 Rumbaugh J,Jacobson I,Booch G.UML参考手册北京机械工业出版社2001
3 Jacobson I,Booch G,Rumbaugh J.统一软件开发过程北京:机械工业出版社,2002
4 Routledge N,Bird L,Goodchild A.UML and XML Schema.http://www. jrpit.flinders.edu.au/confpapers/,2002
5 Carlson D.Modeling XML Vocabularies with UML(Part I,II,III).http://XMLmodeling.com,2001
6 Holzner S.XML完全探索北京中国青年出版社,2001
7 刘 超,张 莉可视化面向对象建模技术北京北京航空航天大学出版社，1999

☆☆

(上接第155页)

5 安全性设计

系统必须防止非授权物理的或电子的闯入、操作或袭击，保证Intranet和所传信息端对端的完整性。同时保证管理人员能方便有效地进行工作。另外各级管理人员的职责和权限不同，显然安全级别也不一样，如果与Internet相连的话，还必须考虑到来自Internet上的安全性问题。在本系统中主要通过安装防火墙、进行分层安全检查(身份认证)、高级别文件加密和内容检查等手段，实现各层数据库和各级网络的隔离、只有网络管理员才可进行物理访问、只能通过代理访问Internet、多层口令控制和超级密码设置等功能。

6 结束语

在实验室条件下，我们实现了该监控系统的实际运行，通过对4组(每组8根)200m长的电缆模拟设置了短路、断路、接地3种不同的故障，结果表明该系统能显示当前电缆网的运行状态，根据单片机采集到的数据生成状态图表。对于故障电缆正确确定其编号并发出报警信号，测试设备通过Intranet对故障网络实现测量并给出结果，故障定位精度不大于2m。

系统通过将Intranet网络技术应用于电缆监控，减轻了管理人员的日常工作量，提高了信息交流和资源共享的能力，有利于故障后的恢复及校正对策的提出，较好地实现了区域级电缆网的网络化监控。实验搭建的系统运行表明本系统易于建立，使用方便，维护量小，系统运行平稳，可靠性较高，作为电网调度中心运行人员的辅助决策系统是行之有效的。特别是友好的用户界面能为维护人员及时提供电缆运行的有效信息，并由此对电缆未来一段时间的运行状态进行预测，发生故障后在最短的时间里提供维修必须的故障数据，从而有效地提高了电力网的现代化管理水平，在区域级电力网建设中有着一定的应用价值。

参考文献

1 刘松海. Intranet设计及实现. 北京:国防工业出版社, 1999
2 高 林. Intranet建设及应用系统开发电信交换, 2000-02：15-19
3 邵 山.自动化报警系统的开发与应用电力系统自动化,2001-02： 64-65
4 徐丙根电力电缆故障探测技术北京机械工业出版社,1999
5 张丽翠电缆故障集中监控的设计与实现长春邮电学院学报,1999- 04：35-39