第35卷 第22期 计 算 机 工 程 2009年11月
Vol.35 No.22 Computer Engineering November 2009
·软件技术与数据库· 文章编号：1000—3428(2009)22—0084—04 文献标识码：A 中图分类号：TP311

基于XML的UML时序图向Petri网的转换

应一舟，叶丽君，郭义喜
(解放军信息工程大学电子技术学院，郑州 450004)

摘 要：针对统一建模语言(UML)时序图与Petri网间转换的问题，提出基于消息的UML时序图向Petri网转换的映射算法。以XMI, XPDL, XSLT为核心，建立基于可扩展标记语言(XML)的实现该映射算法的3层转换方案。结合XML应用环境要求，建立映射处理流程，并通过实例对方案可行性进行了验证。

关键词：可扩展标记语言；统一建模语言；Petri网；转换

Transformation from UML Sequence Diagram to Petri Net Based on XML

YING Yi-zhou, YE Li-jun, GUO Yi-xi
(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004)

【Abstract】 Aiming at the problem for Unified Modeling Language(UML) sequence diagram transformation to the Petri net, this paper proposes the mapping constructs algorithm. The three-level transformation plan based on XMI, XPDL, and XSLT. It establishes the environment requests for applying XML, the process for UML sequence diagram transformation to the Petri net based on XML. Example shows the correctness of the transformation plan.

【Key words】 eXtensible Markup Language(XML); Unified Modeling Language(UML); Petri net; transformation

1 概述

描述方式对仿真概念建模及后续设计开发都有重要影响。从模型校核与验证(V & V)角度考虑，形式化的模型具有更多的校验手段和技术。因此，将自然语言或半形式化的模型转换为形式化的模型，从而实现对概念模型更详尽具体的校验，成为了研究的热点。

统一建模语言(Unified Modeling Language, UML)具有定义良好、易于表达、普遍适用等特点，是一种应用广泛的可视化建模语言。然而，作为一种半形式化的语言，UML中的时序图等动态图形缺乏严密有效的验证和分析方法，难以对其动态行为进行校验。Petri网具有坚实的数学基础和多种定性或定量分析方法，是形式化的模型描述工具，同时具有易于理解的图形特征和UML时序图相似的结构。因此，Petri网成为UML时序图形式化的合适目标。

本文提出一种基于消息的UML时序图向Petri网转换的映射算法，运用XMI, XPDL, XSLT等技术，在可扩展标记语言(eXtensible Markup Language, XML)平台上，设计转换方案和映射处理流程，通过实例对此方案的可行性进行验证。

2 XMI, XPDL, XSLT介绍

XML是W3C组织开发的一种标记语言，是一种能够交换和表示数据的、独立于平台的、强大而精巧的技术。自1998年2月XML1.0成为W3C的推荐标准以来，XML已经成为不同系统之间数据集成和交换的主流。

本文设计的转换方案以XML为基础，下面介绍使用到的几项核心技术。

2.1 XMI

为了实现不同运行平台上模型的共享，对象管理组织(Object Management Group, OMG)制定了XMI(XML Metadata Interchange, XML元数据交换)标准[1]。XMI集成了4个关键的工业标准：扩展标记语言(XML)，统一建模语言(UML)，元对象设施(MOF)，元模型建模(CWM)和元数据存储仓库标准。XMI制定了描述各种元数据定义的统一标准，具体规范了如何从UML模型生成XML文档[2]。

2.2 XPDL

XML过程定义语言(XML Process Definition Language, XPDL)[3]是工作流管理联盟(WfMC)为实现流程定义在不同工作流管理系统之间的互操作性而制定的标准。XPDL标准给出了过程定义互换中用到的实体及实体间的联系、实体间数据引用的规则。XPDL中最上层实体是一个作为数据容器的包。在包实体中包含了工作流过程定义、工作流活动、转移信息、工作流参与者定义、工作流应用定义、工作流相关数据、系统和环境数据等实体。

2.3 XSLT

可扩展样式单语言变换(eXtensible Stylesheet Language Transformations, XSLT)是可扩展样式单语言(XSL)的副产品，以XSL格式化对象(XSL-FO)的标准方式将格式化应用到XML文档。XSLT可以重新排序、复制、压缩、分类以及增删元素，因此，提供了一种自动将XML文档从一种格式转换成另一种格式的方法。

上述3种技术各有特点，在转换过程的不同阶段起作用，具体比较如表1所示。

作者简介：应一舟(1981－)，男，硕士研究生，主研方向：测评与认证；叶丽君，硕士；郭义喜，副教授

收稿日期：2009-06-10 **E-mail：**boat923@tom.com

表 1　XMI, XPDL, XSLT 技术比较

技术	特点	设计初衷	适用领域
XMI	面向对象	实现对元模型的描述，完成不同平台上模型的共享	描述面向对象的模型
XPDL	面向流程	实现不同工作流管理系统的流程定义及信息共享	描述工作流类模型
XSLT	面向转换	实现不同格式的 XML 文档间的转换	描述 XML 文档转换

3　基于 XML 的三层转换方案

引入 XMI、XPDL 和 XSLT 后，构造基于 XML 的三层转换方案，如图 1 所示。

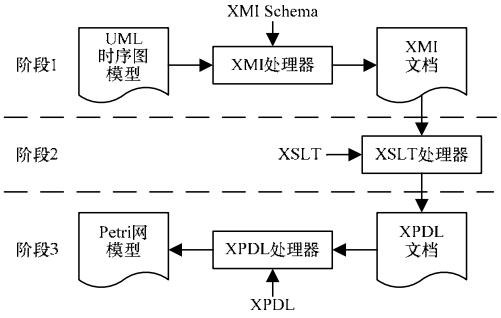

图 1　基于 XML 的三层转换过程

UML 时序图向 Petri 网的转换过程分为以下 3 个阶段：

阶段 1 新建或者从模型库中取出 UML 模型，在 XMI 标准指导下，通过 UML 建模环境提供的 API 对模型中的 UML 时序图进行处理，提取、标注其中的元素、参数、数据、关系、约束等模型信息，依据 XMI 标准进行输出，得到 XMI 文档。

阶段 2 分析阶段一得到的 XMI 文档，考察 XPDL 标准，依据时序图向 Petri 网映射的算法，制定 XMI 同 XPDL 之间转换的 XSLT 样式单，在 XSLT 处理环境下应用该样式单完成转换，得到 XPDL 文档。

阶段 3 根据 XPDL 同 Petri 网间预先定义的映射规则，在相应的 XPDL 处理环境下将阶段 2 得到的 XPDL 文档转换为 Petri 网模型。

转换方案中选用 XMI, XPDL 和 XSLT 技术，有以下优点：

(1)使用 XMI, XPDL，将模型转换端点统一于 XML 平台；使用 XSLT 则有效地将转换演变为此平台上 2 份文档间的映射。

(2)XMI 和 XPDL 是工业标准，得到很多建模工具的支持。利用它们容易实现 UML 和 Petri 网同 XML 之间的转换。

3.1　UML 时序图模型转换为 XMI 文档

由于 XMI 标准本身结合了 UML，该部分的映射规则并不复杂。需要注意的是，对模型的语义没有作用的很多信息将不会被映射到 XMI 文档内。如对象显示的大小、颜色、在图中的位置等[4]。通过研究，得到 UML 时序图同 XMI 之间的部分映射关系如表 2 所示。

表 2　UML 时序图同 XMI 间元素的映射

UML 时序图元素	XMI 文档元素
时序图 Sequence Diagram	Collaboration
活动者 Actor/对象 Object	ClassifierRole
消息 Message	Message

3.2　Petri 网模型转换为 XPDL 文档

用 XPDL 描述 Petri 网，需要了解它们各自元模型的结构，建立起映射关系(表 3)。下面讨论部分映射的建立过程。

表 3　Petri 网同 XPDL 间元素的映射

Petri 网元素	XPDL 文档元素
整个 Petri 网	工作流过程 WorkflowProcess
变迁 T	活动 Activity(IsT=true)
有向弧 F	转换 Transition
库所 P	活动 Activity(IsT=false)
令牌 Token	令牌 Token (Extended Attribute)

整个 Petri 网描述了一个工作流过程，它的象应该是工作流过程；变迁 T 和库所 P 是 Petri 网中的主要对象，虽然有各自不同的属性，却是不同有向弧连接的唯一端点。在 XPDL 中，这类对象定义为活动(activity)。因此，变迁 T 和库所 P 在 XPDL 中的象都是活动(activity)。不同的是，对应变迁的活动的属性 IsT 设置为真(true)，而对应库所的为假(false)；令牌(token)是库所的一个属性，在 XPDL 中，也为活动设置该属性。但是，它只有在属性 IsT 为假时才有意义；有向弧 F 在工作流过程中有相同的结构，XPDL 将其定义为转换(transition)。

3.3　XMI 文档转换为 XPDL 文档

本阶段是转换实现的关键。建立时序图模型同 Petri 网模型之间基于消息的模型映射算法，如图 2 所示。

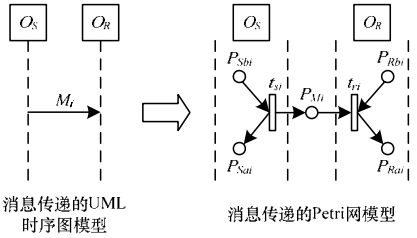

图 2　UML 时序图向 Petri 网的映射

在算法中，将消息的发送与接收映射为变迁，将各对象在发送或接收消息前后的状态映射为库所。

(1)消息传递映射算法

begin

1)对于时序图模型中的每一个对象，设置一个初始库所，定义其为各对象的当前库所，模型中的消息按照其发生的时间顺序定义为 $M_1, M_2, \cdots, M_n$；

2) M_i 为模型的第 i 条消息，设发出消息 M_i 的对象 O_S 的当前库所(即发送消息前的状态)为 p_{Sbi}，接收消息 M_i 的对象 O_R 的当前库所(即接收消息前的状态)为 p_{Rbi}；

For(i=1, i<=n, i=i+1)

{

增加一个表示消息 M_i 的库所 p_{Mi}；

对于对象 O_S，增加一个表示发送动作的变迁 t_{si}，一个表示发送后状态的库所 p_{Sai}；

对于对象 O_R，增加一个表示接收动作的变迁 t_{ri}，一个表示接收后状态的库所 p_{Rai}；

连接库所与变迁： $p_{Sbi} \rightarrow t_{si} \rightarrow p_{Sai}$， $t_{si} \rightarrow p_{Mi} \rightarrow t_{ri}$， $p_{Rbi} \rightarrow t_{ri} \rightarrow p_{Rai}$；

}

end

该算法包含 2 类库所：一类表示传递的消息，另一类表示对象的状态；2 类变迁：一类表示发送消息动作，另一类表示接收消息动作。算法严格根据消息传递的顺序执行映射，一般时序图模型中所包含的要素(对象、消息、消息的发送与接收)都能在其对应的 Petri 网模型中得到体现；而映射之后

的 Petri 网模型中的库所与变迁在相应的时序图中也有其原型含义；库所与变迁的连接严格按照时序图中的语义执行，从而保证完整正确地实现时序图向 Petri 网模型的转换。

(2)映射处理流程

将以上算法应用到 XML 平台，结合 XMI 及 XPDL 标准，得到 XSLT 处理引擎实现 UML 时序图向 Petri 网转换的映射处理流程，如图 3 所示。

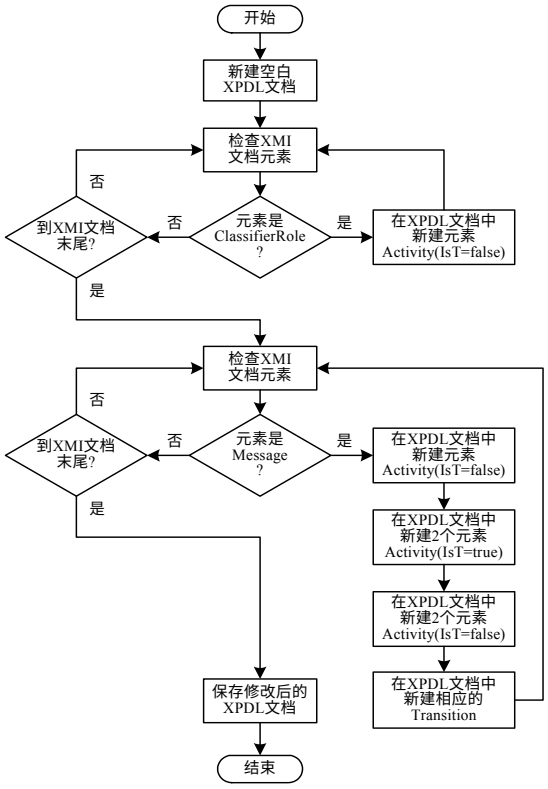

图 3　映射处理流程

整个转换由 XSLT 处理引擎执行。转换开始后，XSLT 处理引擎首先创建一份空白 XPDL 文档，然后根据规则向其中添加各种元素及属性，最后将其输出为转换后的 XPDL 文档。

添加元素主要有 2 个阶段，需要完成对 XMI 文档的 2 次遍历。在第 1 阶段，处理引擎遍历 XMI 文档，找到所有的 ClassifierRole 元素，在 XPDL 文档中为其创建相应的 Activity(IsT=false)元素(其中包括一些重要属性值等信息)。该阶段需要创建针对 ClassifierRole 元素的 XSLT 模板，部分代码如下：

```
<xsl:template match="UML:ClassifierRole">
  <!--新建对象的库所-->
  <xsl:element name="Activity">
      <xsl:attributename="id"><xsl:value-of
select="UML:ClassifierRole/xmi.id"/> </xsl:attribute>
      <xsl:attributename="name"><xsl:value-of
select="UML:ClassifierRole/name"/> </xsl:attribute>
      <xsl:element name="ExtendedAttribute">
          <xsl:attributename="IsT">
<xsl:text>false</xsl:text></xsl:attribute>
          <xsl:attributename="Token">
<xsl:number/></xsl:attribute>
      </xsl:element>
  </xsl:element>
</xsl:template>
```

在第 2 阶段，处理引擎再次遍历 XMI 文档，在检查到每一个 Message 元素时做出相应处理。此阶段需要建立对应 Message 元素的处理模板，部分 XSLT 代码如下：

```
<xsl:template match="UML:Message">
  <!--新建消息的库所-->
  <xsl:element name="Activity">
  …
      <xsl:element name="ExtendedAttribute">
          <xsl:attributename="IsT">
<xsl:text>false</xsl:text></xsl:attribute>
          <xsl:attributename="Token">
<xsl:number/></xsl:attribute>
      </xsl:element>
  </xsl:element>
  <!--新建发送的变迁-->
  <xsl:element name="Activity">
      <xsl:attributename="id"><xsl:value-of
select="UML:Message/xmi.id"/>  <xsl:text>_send</xsl:text>
</xsl:attribute>
      <xsl:attributename="name">
<xsl:text>send</xsl:text></xsl:attribute>
      <xsl:element name="ExtendedAttribute">
          <xsl:attributename="IsT">
<xsl:text>true</xsl:text></xsl:attribute>
          <xsl:attributename="Token">
<xsl:number/></xsl:attribute>
      </xsl:element>
  </xsl:element>
  …
  <!--新建有向弧-->
  <xsl:element name="Transition">
      <xsl:attributename="id"><xsl:value-of
select="UML:Message/xmi.id"/><xsl:text>_Sender_B  </xsl:text>
</xsl:attribute>
      <xsl:attributename="from"><xsl:value-of
select="UML:Message/sender"/> </xsl:attribute>
      <xsl:attributename="to"><xsl:value-of
select="UML:Message/xmi.id"/><xsl:text>_send</xsl:text>
</xsl:attribute>
      <xsl:element name="condition"/>
  </xsl:element>
  …
</xsl:template>
```

第 2 次遍历完成后，XPDL 文档构建完毕。处理引擎将结束工作，输出并保存生成的 XPDL 文档。

4　转换实例

下面是采用以上方案进行转换的一个简单实例。转换前的 UML 时序图如图 4 所示。

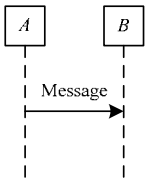

图 4　UML 时序图模型

首先得到此 UML 时序图的 XMI 代码，部分代码如下：

```
<UML:Collaboration  xmi.id="UMLCollaborationInstanceSet.1"
name="CollaborationInstanceSet1">
  …
```

```
                    <UML:Message xmi.id="UMLStimulus.3" name="Message" sender="UMLObject.5" receiver="UMLObject.6" interaction="UMLInteractionInstanceSet.2">
                    </UML:Message>
    …
        <UML:ClassifierRole xmi.id="UMLObject.5" name="A" message2="UMLStimulus.3" isRoot="false" isLeaf="false" isAbstract="false">
        </UML:ClassifierRole>
    …
    </UML:Collaboration>
```

然后通过应用前文构造的 XSLT 样式单，得到该模型的 XPDL 代码，部分代码如下：

```
<WorkflowProcess Id="1" Name="Sample_PetriNT" >
  <ProcessHeader/>
  <Activities>
      <Activity Id="11" Name="A_initial">
          <ExtendedAttributes>
              <ExtendedAttribute IsT="false" Token="1"/>
          </ExtendedAttributes>
      </Activity>
  </Activities>
  …
  <Transitions>
      <Transition Id="18" From="11" To="12">
          <Condition>11.Token > 0</Condition>
      </Transition>
  …
  </Transitions>
</WorkflowProcess>
```

最后根据此 XPDL 代码，得到转换后的 Petri 网模型(见图 5)。该实例说明转换方案是有效可行的。

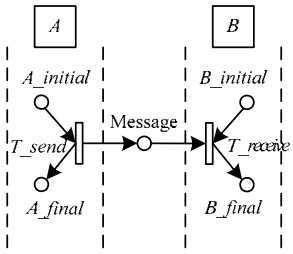

图 5　转换得到的 Petri 网模型

5　结束语

本文分析了 UML 与 XMI、Petri 网与 XPDL 的映射关系，提出了一种基于消息的映射算法，采用 XMI、XPDL 和 XSLT 等核心技术，设计并实现了基于 XML 的三层转换方案和映射处理流程。该项研究为对 UML 动态模型的校核与验证打下了基础，并验证了 XML 技术作为模型转换平台的有效性。

参考文献

[1] OMG. MOF 2.0/XMI Mapping, Version 2.1.1[Z]. (2007-12-01). http://www.omg.org/cgi-bin/apps/doc?formal/07-12-01.

[2] 冯富霞. 基于 XMI 的 UML 模型转换到 XML Schema 的研究[J]. 安徽工程科技学院学报, 2007, 22(2): 44-47.

[3] Workflow Management Coalition(WfMC). Workflow Process Definition Interface — — XML Process Definition Language, Version1.0[Z]. (2002-07-12). http://www.wfmc.org/standards/docs/tc-1025_10 xpdl 102502.pdf.

[4] Marchal B. Working XML: UML, XMI, and Code Generation, Part1[Z]. (2004-04-15). http://www.ibm.com/developerworks/library/x-wxxm23/index.html?S_TACT=105AGX52&S_CMP= cn-a-x.

编辑　索书志

(上接第 83 页)

4.2　数据集大小对算法的影响

对维数为 6 的不同大小的数据集进行测试，已知每个数据集初始状态下的轮廓，在取不同大小的数据集时分别对数据集增加或减少 2 维空间，然后将本文算法的运行时间与 NN 算法的运行时间进行比较，如图 2 所示，其中，空间集大小 d 取 6，数据集大小变化范围为[1 000, 6 000]。NN 算法的性能随数据集的增大而降低，主要是因为不能利用现有轮廓结果，在更新操作时都要重新进行计算，需要处理的数据点太多，所以算法性能随之降低。而本文算法能充分利用现有轮廓查询结果，在不同大小的数据集下，无论是对于空间集增大还是减小的情况，运行时间都明显比 NN 小，性能优于 NN，算法效率受数据集大小的影响不太明显。

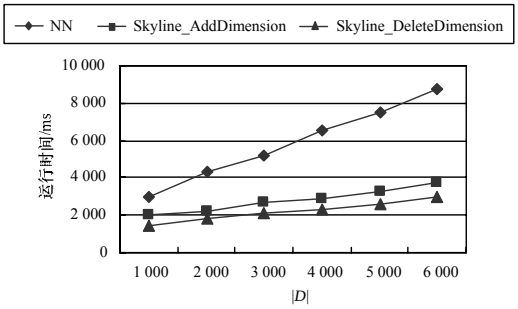

图 2　不同数据集下算法运行时间的比较

5　结束语

针对空间集的动态变化，本文提出一种基于共享策略的轮廓更新算法。当空间集维数增大或减小时，依据共享策略，在原有轮廓的基础上，通过判断部分数据点实现对轮廓的更新，在很大程度上提高了运算效率。实验验证本文算法能有效地完成动态空间集下的轮廓更新，计算量少，并且能保证计算结果正确。

参考文献

[1] Borzonyi S, Kossmann D, Stocker K. The Skyline Operator[C]//Proc. of ICDE'01. Heidelberg, Germany, [s. n.], 2001: 421-430.

[2] Kossmann D, Ramsak F, Rost S. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries[C]//Proc. of VLDB'02. Hong Kong, China: [s. n.], 2002: 275-286.

[3] Jian Pei, Wen Jin, Ester M, et al. Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces[C]//Proc. of VLDB'05. Sydney, Australia: [s. n.], 2005: 253-264.

[4] Zhang Zhenjie, Guo Xinyu, Lu Hua, et al. Discovering Strong Skyline Points in High Dimensional Spaces[C]//Proc. of CIKM'05. Bremen, Germany: [s. n.], 2005: 247-248.

[5] Chan Chee-Yong, Jagadish H V, Tan Kian-Lee, et al. Finding k-Dominant Skylines in High Dimensional Space[C]//Proc. of SIGMOD'06. Chicago, USA: [s, n.], 2006: 503-514.

编辑　张　帆