UML活动图到Petri网的转换方法及实现研究

赵俊峰 周建涛 邢冠男
(内蒙古大学计算机学院 呼和浩特 010021)

摘 要 统一建模语言UML缺乏形式化语义,由其描述的模型难以进行动态的分析和验证。而Petri网在具有丰富而严格语义的同时,又有严谨的数学分析方法。综合运用Petri网和UML能够提高软件描述的全面性、一致性、精确性和完整性。研究了UML活动图向Petri网的转换规则,并依据转换规则实现了模型转换工具APConverter。此工具能有效地将活动图转换为Petri网模型并生成PNML文件,进而更好地对UML模型进行分析和验证。

关键词 UML,活动图,Petri网,PNML,转换规则
中图法分类号 TP301.2 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.029

Research of Translating UML Activity Diagram to Petri Net

ZHAO Jun-feng ZHOU Jian-tao XING Guan-nan
(College of Computer Science, Inner Mongolia University, Hohhot 010021, China)

Abstract Because Unified Modeling Language lacks formal semantics, the dynamic analysis and verification to UML based model are difficult to carry out. However Petri net not only has sufficient and rigid semantics, but also is equipped with precise analysis method. Comprehensive usage of Petri net and UML can efficiently improve the comprehensiveness, consistency, accuracy and completeness of the software model. The translation rules from UML activity diagram to Petri net were proposed. Then a translation tool called APConventer was implemented for translating activity diagram to Petri Net Markup Language. Activity diagram can be translated into Petri net and expressed in PNML effectively by the usage of the tool, so the UML model can be analyzed and verified better.

Keywords UML, Activity diagram, Petri net, PNML, Transformation rule

1 引言

随着面向对象技术的发展,统一建模语言UML越来越受到工业界和应用界的重视和支持。作为一种定义良好、易于表达、功能强大且普遍适用的建模语言,它融入了软件工程领域的新思想、新方法和新技术,支持从需求分析开始的软件开发全过程,并已被用来为不同类型的系统建模。然而由于UML是半形式化的,其语法虽然采用形式化规约,但其语义部分则是用自然语言描述的,缺乏准确的语义,这就使得难以对所建模型进行动态的分析和验证,因此UML的形式化日益成为人们关注的研究领域。

Petri网是一种形式化建模语言,可以描述并发系统的同步、异步、冲突等特征,同时具有图形化的建模元素和精确的分析方法和工具。它是德国科学家C. A. Petri于1962年在其博士论文《用自动机通信》中首次提出的,在工作流、柔性制造、网络协议等复杂系统的建模与分析过程中[1]应用广泛。

深入研究UML模型的形式化语义并借助Petri网进行分析与验证是解决UML模型验证的有效手段。国内外对于UML模型的形式化展开了广泛的研究,在理论上取得了一定的成果,但是对于模型转换工具的实现较少。而对于现实开发中的复杂模型转换而言,自动化的转换工具是必要的,所以对于如何实现高效、可扩展的模型转换工具是需要研究的。本文将从分析UML2.0活动图语义入手,研究UML活动图模型元素到基本Petri网的简单且易于通过算法实现的转换规则,实现基于该规则的高效、可扩展和符合模型交换标准的模型转换工具APConverter。

2 相关工作

目前,关于UML模型到Petri网的转换,国内外已有一些研究成果,大体分为如下两类:

将UML模型转换为高级Petri网,其中包括:文献[2-4]将所有的UML活动模型转换为广义随机Petri网用以在软件开发过程中对模型进行集成,解释如何依据对系统进行描述的一组状态图产生可执行广义随机Petri网以及如何转换活动图到广义随机Petri网;文献[5]定义转换规则将UML规范自动映射为高级Petri网从而为UML增加形式化语义;文献[6]提出了UML状态图和协作图向对象Petri网模型映射的方法;文献[7]提出转换框架实现将UML状态图和协作

到稿日期:2013-04-08 返修日期:2013-05-27 本文受国家自然科学基金资助项目(61262082),国家教育部重点资助项目(212025),内蒙古杰出青年学者科学基金资助项目(2012JQ03),内蒙古自然科学基金资助项目(2011MS0911)资助。
赵俊峰(1976—),男,博士生,讲师,主要研究方向为软件工程与形式化方法,E-mail:cszjf@imu.edu.cn;周建涛(1974—),女,教授,博士生导师,主要研究方向为网络计算与形式化方法,E-mail:cszjtao@imu.edu.cn(通信作者);邢冠男(1984—),女,硕士,主要研究方向为形式化方法。

图转换为着色 Petri 网。文献[8]讨论了 UML 模型向对象 Petri 网的半自动的转换方法。该类转换能够较好地对原模型进行描述，适合于描述复杂模型，但转换规则复杂，不容易实现工具进行自动转换。

将 UML 模型转换为基本 Petri 网，其中包括：文献[9]将 Petri 网作为形式化模型为 UML2 时序图的主要概念提供语义支持，这种方法专注于捕获行为，并将其模拟并可视化；文献[10]对 UML1. X 的时序图进行相应的扩展，使得消息传递机制中的几种特殊关系，如并发、选择、同步等能够通过扩展时序图表达清楚，进而给出扩展时序图转换为 Petri 网时的 9 种转换规则；文献[11]针对 UML2. 0 时序图模型提出将时序图转换为 Petri 网的算法；文献[12]在分析 UML2. 0 时序图特点的基础上，提出基于消息的模型映射算法，并对时序图中的特殊结构(可选、条件、并行、循环) 分别提出其相应的映射算法；郭峰等人提出一种可以准确描述 UML 状态图动态特征的形式化模型 SC_Net，并给出了从 UML 状态图到 SC_Net 的转换步骤[13]；笔者曾在学位论文中研究了活动图的转换规则，并且证明了转换后 Petri 网模型与原模型的等价性[14]。笔者在先期工作中对已有行为模型向 Petri 网的转换机制进行对比分析，提出了改进的 UML2 时序图到 Petri 网的转换规则，实现了时序图向 Petri 网标记语言的转换工具[15]。该类转换规则比较简单，容易实现工具进行自动转换，但生成的模型较复杂。

UML 活动图是状态机的一个变体，用以对业务过程进行建模，描述一个单独用例或用例场景的执行逻辑，或者描述一个业务规则的详细逻辑。但与状态图的目的有一些小的差别，活动图的主要目的是描述动作(执行的工作和活动)及对象状态改变的结果。当状态中的动作被执行(不同于正常的状态图，它无需制定任何事件)时，活动图中的状态(成为动作状态)直接转移到下一个阶段，并且能够描述并行动作[16]。UML 活动图在软件系统的动态特征描述中起重要作用，对活动图的分析验证是保证业务设计合理高效的有效手段。

现有针对活动图的形式化研究同样涵盖上述两种研究类型。其中，文献[17]将在线多角色游戏的 UML 活动图转换为 PEPA 网并进行了分析，但缺少对转换过程的描述。为实现系统形式化验证以及性能评估，文献[18]对模糊 UML 活动图到 Petri 网的转换进行了研究。为实现软件性能评估，文献[3]提出了活动图中模型元素到广义随机 Petri 网的转换规则及算法。这些研究虽然提出了相应的模型转换规则，但均未涉及自动转换工具的实现，只是通过具体实例验证了转换规则的合理性。

3 活动图到 Petri 网的转换规则

活动图是描述操作实现中所完成的工作以及用例场景的动态模型。Petri 网通过变迁点火序列可知模型的可能运行。以此为基础，活动图和 Petri 网之间有一种自然的映射关系。

定义活动图和 Petri 网模型元素的转换规则，将活动图中的每个元素转换为相应的 Petri 网单元，然后再根据活动图中状态之间的迁移关系把转换后的 Petri 网单元连接起来，就可以形成完整的 Petri 网。Petri 网中表示起始状态的库所包含一个 token，触发整个系统的运行。

3.1 起始状态(start)

活动图的起始状态是整个活动的开始，在本转换规则中，映射为包含一个 token 的库所。为了满足系统需要，起始状态转换成一个库所的同时，增加了一个系统的内部变迁以及库所到变迁的有向弧，从而保证该起始状态可以触发整个系统。转换规则如图 1 所示。

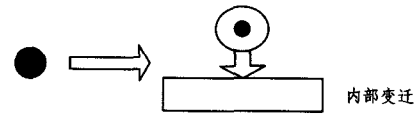

图 1 起始状态(start)到 Petri 网的转换规则

3.2 终止状态(final)

活动图中的终止状态是整个活动流程的结束，是一个静止状态，所以终止状态映射为一个库所。转换规则如图 2 所示。

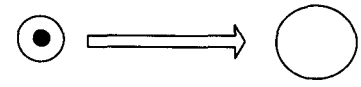

图 2 终止状态(final)到 Petri 网的转换规则

3.3 活动(activity)

活动图中的活动表示一个动作正在执行，正是由于 Petri 网中的变迁表示的也是一个动作的执行，因此在本转换规则中，活动映射为一个变迁。为了保证 Petri 网的构成，增加了一个系统内部的库所以及库所到该变迁的有向弧。转换规则如图 3 所示。

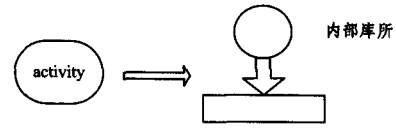

图 3 活动(Activity)到 Petri 网的转换规则

3.4 迁移(transition)

活动图中各个建模元素之间的实线箭头表示迁移。因此，在本转换规则中，迁移被映射为库所和变迁之间的弧。由于在 UML 活动图中，迁移是有向箭头，可以是活动到活动，也可以是状态到活动，因此，在 Petri 网中通过弧的属性体现连接上的不同。转换规则如图 4 所示。

图 4 迁移(Transition)到 Petri 网的转换规则

3.5 动作流决策点(decision point of action flow)

3.5.1 动作流分支(branch)

在活动图中，控制流是指当一个动作结束时，马上进入下一个动作，而一系列的动作和动作间的控制流构成一个动作流，如图 5 所示。

图 5 UML2. 0 活动图动作流

图 5 中的动作流是顺序执行的，然而动作流也可以包含分支(branch)和合并(merge)。在 UML2. 0 中用一个菱形表示动作流的决策点。动作流的一个分支有一个进入控制流和两个或多个离开的控制流。每个离开的控制流上有一个布尔表达式，且这些表达式是互斥的。此外，这些表示应该覆盖了所有的可能性，否则动作流有可能无法继续执行下去。当某个分支上的布尔表达式为真时，该动作流转移到对应分支。

在本转换规则中，将动作流进行决策前的状态映射为库所，然后用弧指向每个代表分支的变迁。转换规则如图 6 所示。

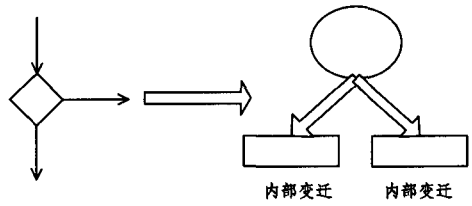

图 6　动作流分支(branch)到 Petri 网的转换规则

3.5.2　动作流合并(merge)

动作流可以产生分支，多个分支也可以在后续的执行过程中进行合并。动作流的一个合并有两个或多个进入控制流和一个离开的控制流。

在本转换规则中，将动作流的合并映射为与合并分支相同数目的库所、一个表示合并的变迁以及库所到合并变迁的弧。转换规则如图 7 所示。

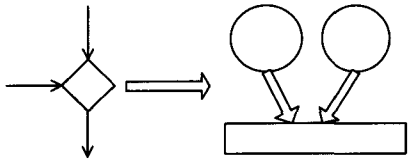

图 7　动作流合并(merge)到 Petri 网的转换规则

3.6　并发控制流(parallel control flow)

活动图中控制流也是可以并发的，在 UML2.0 规范中用同步条表示控制流的并发执行。活动图的同步包含两种类型：分岔(fork)和汇合(join)，如图 8 所示。

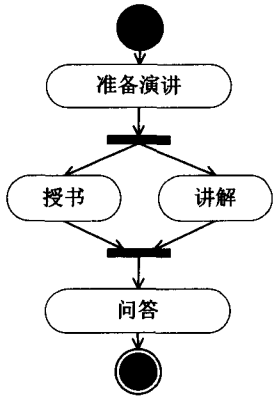

图 8　UML2.0 活动图并发控制流

3.6.1　分叉(fork)

UML2.0 中，活动图的分岔用一条垂直或水平的粗线表示，其含义为：把一条控制流分成两个或多个控制流，且这些控制流可以并发执行。

根据分岔的动作语义，在本转换规则中，将分岔转换为一个库所、一个变迁以及一条由库所指向变迁的弧。转换规则如图 9 所示。

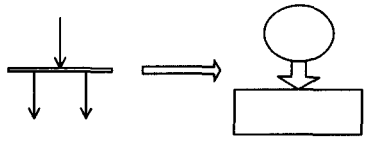

图 9　分岔(fork)到 Petri 网的转换规则

3.6.2　汇合(join)

活动图的一个汇合有两个或多个进入控制流和一个离开控制流。汇合把两个或多个并发的控制流同步起来，执行时表现为：先到达的控制流在此等待，直到所有的进入控制流都到达这里，然后执行离开的控制流。活动图中分岔处生成的控制流数目等于进入汇合的控制流数目。

根据汇合的动作语义，在本转换规则中，将汇合映射为进入控制流数目的库所、一个表示汇合的变迁以及这些库所到汇合变迁的弧。转换规则如图 10 所示。

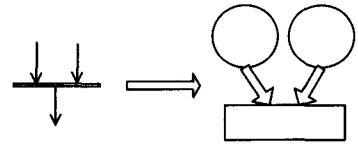

图 10　汇合(join)到 Petri 网的转换规则

3.7　泳道(swimlane)

由于活动图中的泳道表示的是各个活动的所属，实质上并不影响活动图的进行，因此在转换的过程中，可以将活动图中关于泳道的信息表示为活动的属性，在只关注活动运行的情况下，泳道可以忽略。

3.8　对象和对象流(object & object flow)

活动图中的对象表示的是活动之间的输入输出关系，并且对象可以有不同的状态。因为对象是一个静态的结构，所以在本转换规则中，对象被映射为库所。与此同时，同样为了保证 Petri 网的构成，增加了系统内部的变迁以及该库所到变迁的弧。转换规则如图 11 所示。

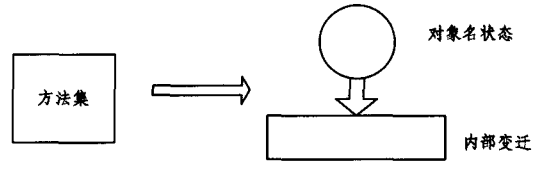

图 11　对象(Object)到 Petri 网的转换规则

活动图中的对象流是连接对象和活动的，在 UML 活动图中用虚线箭头来表示，以区别其他的迁移。在 Petri 网中，把对象流转换为连接弧。与迁移相似，为了区别对象流连接的是“活动到对象”还是“对象到活动”，转换后的连接弧可在属性中体现以上信息。转换规则如图 12 所示。

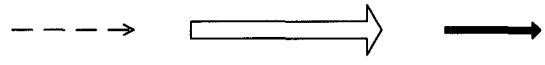

图 12　对象流(Object flow)到 Petri 网的转换规则

4　APConverter 转换工具的设计与实现

当软件分析设计中需对 UML 模型进行分析验证时，其完整过程包括 3 个步骤。首先，在 UML 建模工具中建立 UML 模型。然后，模型转换工具导入生成的 UML 模型文件，工具解析模型文件获取模型元素，依据转换规则对模型元素进行转换，生成 Petri 网模型并导出模型文件。最后，Petri 网工具导入 Petri 网模型文件进行分析验证。具体处理流程如图 13 所示。APConverter 实现该完整过程中的第 2 步，即模型转换。

图 13　模型处理流程

4.1　基于 XMI 的 UML 模型信息提取

XMI(XML-based Metadata Interchange) 使用扩展标记语言(XML),为程序员和其它用户提供元数据信息交换的标准方法,目的在于帮助使用统一建模语言(UML)以及不同语言和开发工具的程序员彼此交换数据模型,从而避免不同工具间模型表示的不可兼容。XMI 作为一种工业标准,得到主要建模工具的支持。APConverter 转换工具将从 XMI 文件提取 UML 模型信息,图 14 为输入源 XMI 文件片段。

```
<?xml version="1.0" encoding="GB2312"?>
<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/spec/UML/2.0" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
  <xmi:Documentation exportedFromDifferentName="false" xmi:Exporter="Visual Paradigm for UML" xmi:ExporterVersion="1.0.0"/>
  <xmi:Extension xmi:Extender="Visual Paradigm for UML">
    <projectProperties>
      <projectProperty name="author" value=""/>
      <projectProperty name="company" value=""/>
      <projectProperty name="description" value=""/>
    </projectProperties>
  </xmi:Extension>
  <uml:Model name="project" xmi:id="vvFatuyGAqDeAK19">
    <xmi:Extension xmi:Extender="Visual Paradigm for UML">
      <vpumlChildModels>
        <vpumlModel id="umJH_WyFS_jGahJH" modelType="UseCasePropConfig">
          <properties>
            <property name="name" type="string" value=""/>
            <property name="modelType" type="string" value="UseCasePropConfig"/>
            <property htmlValue="" name="documentation" type="htmlString" value=""/>
            <property name="pmAuthor" type="string" value="Administrator"/>
            <property name="pmCreateDateTime" type="string" value="1337570267020"/>
            <property name="pmLastModified" type="string" value="1334656119016"/>
          </properties>
          <vpumlChildModelRefs>
            <modelRef value="umJH_WyFS_jGahJa"/>
            <modelRef value="umJH_WyFS_jGahJj"/>
            <modelRef value="exJH_WyFS_jGahJs"/>
          </vpumlChildModelRefs>
          <vpumlChildModels>
            <vpumlModel id="umJH_WyFS_jGahJa" modelType="UseCaseRank">
              <properties>
                <property name="name" type="string" value="Low"/>
                <property name="modelType" type="string" value="UseCaseRank"/>
                <property htmlValue="" name="documentation" type="htmlString" value=""/>
                <property name="pmAuthor" type="string" value="Administrator"/>
```

图 14　XMI 文件片段

4.2　基于 PNML 的 Petri 网模型信息保存

```
<pnml xmlns="http://www.pnml.org/version-2009/grammar/pnml">
  <net id="net" type="http://www.pnml.org/version-2009/grammar/ptnet">
    <page id="toppage"><name><text>page</text></name>
    <place id="P1"/>
      <place id="P2"/>
      <place id="P3"/>
      <place id="P4"/>
      <place id="P5"/>
      <place id="P6"/>
      <place id="P7"/>
      <place id="P8"/>
      <place id="P9"/>
      <place id="P10"/>
      <place id="P11"/>
      <place id="P12"/>
      <place id="P13"/>
      <place id="P14"/>
      <place id="P15"/>
      <place id="P16"/>
      <place id="P17"/>
      <place id="P18"/>
      <place id="P19"/>
      <place id="P20"/>
      <transition id="T1"/>
      <transition id="T2"/>
      <transition id="T3"/>
      <transition id="T4"/>
      <transition id="T5"/>
      <transition id="T6"/>
      <transition id="T7"/>
      <transition id="T8"/>
      <transition id="T9"/>
      <transition id="T10"/>
      <transition id="T11"/>
```

图 15　PNML 文件片段

PNML(Petri Net Markup Language)实现了 Petri 网描述的标准化,实现了 Petri 网建模工具之间的模型交换[19]。PNML Framework 是对 PNML 语法的 Java 实现。所以,APConverter 转换工具以 XMI 文件作为模型输入,通过 PNML Famework 实现目标模型的 PNML 输出。图 15 为转换生成的 PNML 文件片段。

4.3　转换工具的设计

经过分析,确定如图 16 的设计类图。工具设计转换代理类 ConverterAgent,由其设置待转换源模型,调用源模型转换方法实现模型转换。源模型实现 Diagram 接口,而接口定义了转换规则对应的实现方法 convert。目前工具实现了活动图源模型的转换类 ActivityDiagram,将来可以扩展实现其它源模型的转换,只需定义具体实现 Diagram 接口的源模型转换类即可保证工具良好的可扩展性。转换算法实现时首先解析模型文件生成完整模型对象,然后通过遍历该内存对象完成模型转换,从而可以保证转换的高效性。

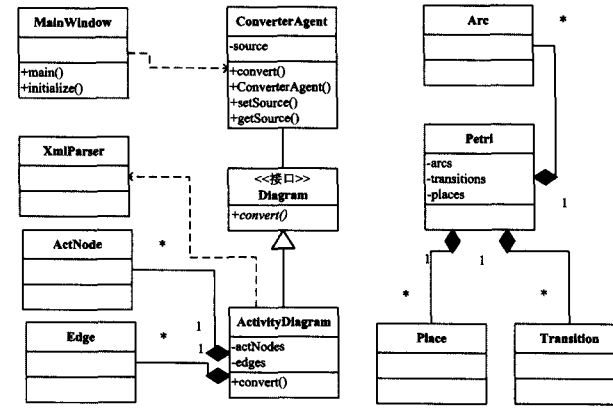

图 16　APConverter 设计类图

4.4　转换工具的验证

转换后得到的 Petri 网模型,保存为 PNML 文件。工具 Tina 支持 PNML 文件的可视化[20],模型的查看和验证工作在辅助工具 Tina 中进行。Tina 提供了 4 种不同的可视化算法,为 Petri 网模型的验证带来了极大的便利。图 17 中活动图描述了电子商务系统中订单的处理过程,其中包括分叉、合并、动作流分支和动作流合并等模型元素。转换得到的 Petri 网模型在 Tina 中的可视化效果如图 18 所示。

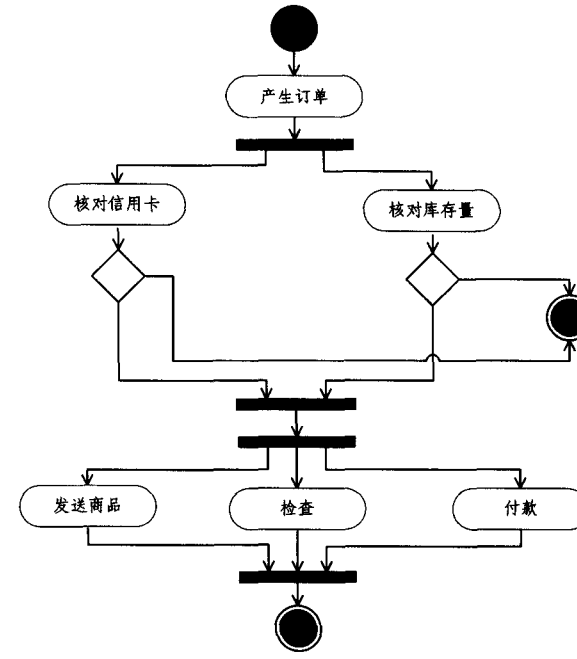

图 17　UML 活动图模型

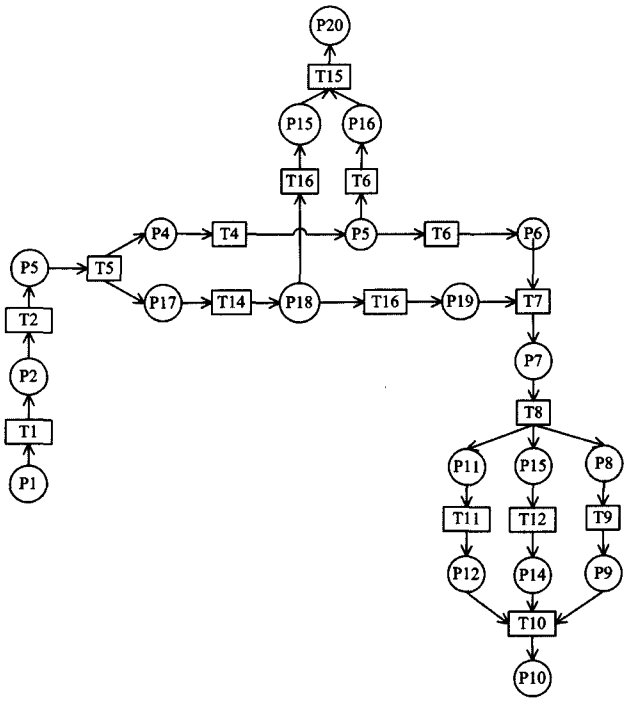

图 18　转换得到的 Petri 网模型

通过分析活动图所描述的行为路径集合与转换为 Petri 网模型的行为路径集合是否等价来判断源模型与转换后模型的一致性。活动图可以看成是一个有向图，通过遍历这个有向图能够获得活动图的行为路径集合。同理，Petri 网也可以看成是一个有向图，遍历转换得到的 Petri 网模型能够得到模型的行为路径集合。经过分析对比，两个模型表示的路径集合等价，从而说明两个模型描述的问题信息等价。目前转换得到的 Petri 网模型比较复杂，下一步工作考虑对模型转换规则进行简化或者对得到的模型进行化简，从而使得模型的验证更为高效。

结束语　本文研究了如何将 UML2.0 的活动图模型转换为 Petri 网模型，以便实现模型的形式化分析与验证，最终提高软件的开发效率，保证软件的质量。

在分析已有研究成果的基础上，提出了活动图到 Petri 网的转换规则，设计并实现了高效、可扩展和符合模型交换标准的自动转换工具 APConverter。通过测试发现，工具能有效地将活动图转化为 Petri 网模型，生成 PNML 文件，实现了转换规则在实际开发中的有效应用，进而更好地对模型进行分析和验证。

本文为 UML2.0 的活动图模型向 Petri 网的转换工具的实现做了有力的探索。进一步的工作可以向两方面扩展：1）将待转换模型扩展到 UML2 中的其他模型，从而更全面地对软件系统的模型进行分析和验证；2）将高级 Petri 网作为转换目标模型，从而更好地简化模型并增强模型的表达能力。最终将研究成果体现为工具实现，使其在特定开发领域具有实际应用价值。

参 考 文 献

[1] 吴哲辉. Petri 网导论[M]. 北京：机械工业出版社，2006

[2] Campos J, Merseguer J. On the Integration of UML and Petri Nets in Software Development[C]// 27th Int. Conf. on Applications and Theory of Petri Nets and Other Models of Concurrency(ICATPN 2006), volume 4024 of Lecture Notes in Computer Science, 2006. Berlin: Springer, 2006: 19-36

[3] López-Grao J P, Merseguer J, Campos J. From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering[C]// 4th Int. Workshop on Software and Performance(WOSP 2004), 2004. New York: ACM Press, 2004: 25-36

[4] Merseguer J, Campos J, Bernardi S, et al. A Compositional Semantics for UML State Machines Aimed at Performance Evaluation [C] // 6th Int. Workshop on Discrete Event Systems (WODES 2002), 2002. NJ: IEEE CS Press, 2002: 295-302

[5] Baresi L, Pezzè M. On Formalizing UML with High-Level Petri Nets[C]// Concurrent Object-Oriented Programming and Petri Nets: Advances in Petri Nets, volume 2001 of Lecture Notes in Computer Science, 2001. Berlin: Springer, 2001: 276-304

[6] Saldhana J A, Shatz S. UML Diagrams to Object Petri Net Models: An Approach for Modeling and An alysis[C]// Proceedings of the International Conference on Software Engineering and Knowledge Engineering(SEKE), 2000. 2000: 103-110

[7] Fernandes J M, Tjell S, Jørgensen J B, et al. Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net[C]// Sixth International Workshop on Scenarios and State Machines(SCESM'07), 2007. NJ: IEEE, 2007: 2-2

[8] Bares L. Some Preliminary Hints on Formalizing UML with Object Petri Nets[C]// Proceedings of the 6th World Conference on Integrated Design and Process Technology, IDPT-2002. Pasadena, USA, 2002: 3-6

[9] Eichner C, Fleischhack H, Meyer R, et al. Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets[M]// SDL 2005: Model Driven. Berlin Heidelberg: Springer, 2005: 133-148

[10] 周长红. UML 图的 petri 网建模[D]. 青岛：山东科技大学，2004

[11] 谢彦辉，姚淑珍，郭峰. 顺序图至 Petri 网转化方法的研究与实现[J]. 计算机工程，2006，32(6)：260-262

[12] 叶丽君，桑海，张明清，等. 基于 UML 的概念模型的 Petr i 网映射算法研究[J]. 计算机仿真，2009，26(3)：112-116

[13] 郭峰，姚淑珍. 基于 Petri 网的 UML 状态图的形式化模型[J]. 北京航空航天大学学报，2007，33(2)：248-250

[14] 邢冠男，周建涛. UML 活动图到 PNML 转换的研究与实现[D]. 呼和浩特：内蒙古大学，2009

[15] 赵俊峰，周建涛. UML 时序图向 PNML 转换的研究与实现[J]. 武汉大学学报：理学版，2011，57(6)：511-516

[16] Arlow J, Neustadt I. UML 和统一过程实用面向对象的分析和设计[M]. 方贵宾，等译. 北京：机械工业出版社，2006

[17] Canevet C, Gilmore S, Hillsone J, et al. Analysing UML 2.0 Activity Diagrams in the Software Engineering Performance Process[J]. WOSP'04, 2004, 29(1): 74-78

[18] 张炜钟，王智学，陈剑. 一种 UML 活动图到模糊 Petri 网的转换算法[J]. 系统仿真学报，2009，20(8)：102-106

[19] Hillah L M, Kindler E, Kordon F, et al. A primer on the Petri Net Markup Language and ISO/IEC 15909-2[J]. Petri Net Newsletter, 2009, 76: 9-28

[20] Berthomieu B, Vernadat F. Time Petri nets analysis with TINA, Quantitative Evaluation of Systems[C]// QEST 2006. Third International Conference on. IEEE, 2006: 123-124