第37卷 第14期 计 算 机 工 程 2011年7月
Vol.37 No.14 Computer Engineering July 2011

·软件技术与数据库· 文章编号：1000—3428(2011)14—0047—03 文献标识码：A 中图分类号：TP391

# UML活动图模型正确性诊断方法

周春燕，李绪蓉，周 良

(南京航空航天大学信息科学与技术学院，南京 210016)

**摘 要：**UML活动图的正确性决定了业务流程的正确执行，为此，提出一种UML活动图模型的正确性诊断方法。将模型分解后，对子模型进行模型验证，并对验证错误的子模型进行模型诊断，得到诊断结果。在质量管理过程实例中的应用结果表明，该方法能减小模型空间，减少诊断次数，准确锁定错误，有效地对模型的正确性进行诊断。

**关键词：**活动图；模型正确性；模型分解；模型验证；模型诊断；

# Correctness Diagnosis Method of UML Activity Graph Model

ZHOU Chun-yan, LI Xu-rong, ZHOU Liang

(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

**【Abstract】**The correctness of the activity graph decides that the correctness of execution of business flow, so this paper proposes an method for diagnosis of correctness of UML activity graph model. It verificates the model. Model diagnosis of sub-models which are wrong and receiving the diagnosis results. The application of example of quality management process, demonstrates this approach can reduce the space of model, decrease diagnosis frequency, lock error accurately, diagnose the model correctness effectively.

**【Key words】**activity graph; model correctness; model decomposition; model verification; model diagnosis

**DOI:** 10.3969/j.issn.1000-3428.2011.14.014

## 1 概述

活动图是提供独立于目标系统的高层次流程描述，多数非常复杂并包含了大量的活动和相关的协作约束，它的正确性决定了系统中实际业务过程执行的正确性。因此，在将其运用于目标系统前，必须及时发现并更正错误。

活动图类似于工作流图，结合工作流模型的验证和诊断方法，本文提出先验证后诊断的UML活动图模型正确性诊断方法，采用图分解技术来加快诊断速度。同时结合实例进行方法应用，验证其有效性。

## 2 基于活动图的模型表示

活动图描述了需要进行的活动以及执行这些活动的顺序。活动图有起点、终点、活动、判断、对象、同步条、泳道、控制流等基本元素。鉴于工作流图模型[1]，活动图模型定义为一个八元组：$G=(N, E, T, ST, EN, AC, JI, SP)$，其中：

(1)$N$是节点，是一个七元组，$N=(type, dout, din, postNodes, preNodes, inEdges, outEdges)$。设$n\in N$，则：$n.type\in T$；$dout$是$n$的出度；$din$是$n$的入度；

$n.outEdges=\{e|e\in E\cap e.fromNode=n\}$

$n.inEdges=\{e|e\in E\cap e.toNode=n\}$

$postNodes=\{n'|n'\in N\cap e\in E\cap e.fromNode=n\cap e.toNode=n'\}$

$reNodes=\{n'|n'\in N\cap e\in E\cap e.fromNode=n'\cap e.toNode=n\}$

$N=ST\cup EN\cup AC\cup JI\cup SP$

(2)$E$是有向边，$E\subseteq N\times N$，是一个二元组$E=(fromNode, toNode)$。其中，$fromNode\in N$；$toNode\in N$。

(3)$T$是节点类型集合，$T=$\{start, end, activity, AND Splits, AND Joins, XOR Splits, XOR Joins\}，如图1从左到右依次所示。

(4)$ST\in N$，是模型的开始节点，$ST.type$=start，$ST.dout$=1，并且$ST.din$=0。

(5)$EN\in N$，是模型的结束节点，$EN.type$=end，$EN.dout$=0，并且$EN.din$=1。

(6)$AC\subseteq N$，是模型的活动节点，$AC.type$=activity，$AC.dout$=1，并且$AC.din$=1。

(7)$JI\subseteq N$，是模型的Joins节点，$JI.type\in$\{AND Joins, XOR Joins\}，$JI.din\geqslant 1$，并且$JI.dout$=1。

(8)$SP\subseteq N$，是模型的Splits节点，$SP.type\in$\{AND Splits, XOR Splits\}，$SP.din$=1，并且$SP.dout\geqslant 1$。

(9)如果$n\in SP$，节点$n$被访问，当$n.type$=XOR Splits时，则其中一条出口路径随机被访问；当$n.type$=AND Splits时，则所有出口路径都被访问。

(10)如果$n\in JI$，当$n.type$=XOR Joins时，当且仅当一条入口路径被访问时，节点$n$才会被访问；当$n.type$=AND Joins时，当其所有入口路径都被访问时，节点$n$才会被访问。

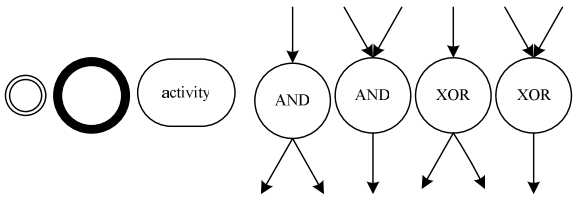



**图1 节点类型**

在进行对活动图的诊断前，需要将活动图转换为标准模型。首先去除活动图中不必要的元素，如对象和泳道；然后保留起点、终点、活动、判断、同步条和控制流。最后进行

**作者简介：**周春燕(1986－)，女，硕士研究生，主研方向：信息系统集成；李绪蓉、周 良，副教授、博士

**收稿日期：**2010-12-10 **E-mail：**dolphin86abc@yahoo.com.cn

以下的转换：(1)起点有多条出口路径，则在起点和多条路径中间添加 XOR Splits；(2)有多个终点，则添加一个 XOR Joins，将多个终点的入口路径指向这个 XOR Joins，XOR Joins 指向一个终点，其余的移除；(3)活动有多条入口路径和多条出口路径，则在入口路径与活动中间添加 AND Joins，在出口路径与活动中间添加 AND Splits；(4)判断有多条入口路径，则在判断与入口路径中间添加 XOR Joins；(5)同步条有多条入口路径，则在同步条与入口路径中间添加 AND Joins。

## 3 模型正确性诊断方法

模型的正确性不仅包括语法的正确性，还包括语义的正确性。模型具有正确性当且仅当具有[2]：

(1)活性：从开始节点 *ST* 开始可到达的每个节点 *n*，可以到达终止节点 *EN*。

(2)可靠性：终止节点 *EN* 必须是从开始节点 *ST* 开始的可达的唯一的终止节点。

模型的正确性检查，主要检测两大类型的冲突：

(1)死锁：如果 $n\in JI$ 并且 *n.type*=AND Joins，只有 *n* 的一部分入口路径被访问，则会导致模型不具有活性，在 *n* 节点上发生死锁。

(2)乏同步：如果 $n\in JI$ 并且 *n.type*=XOR Joins，*n* 的多个入口路径被访问，则会导致模型不具有可靠性，在 *n* 节点上发生乏同步。

### 3.1 模型分解

模型诊断所需的时间空间与模型大小有直接关系。采用业务过程模型的分解技术[3]，将模型分解成多个子模型分别进行分析，锁定错误在哪个子模型中，提高检测和诊断的速度。

设模型 $G=(N, E, T, ST, EN, AC, JI, SP)$，分解的子模型 $F=(N', E', T', ST', EN', AC', JI', SP')$ 是 *G* 的非空子模型，是单入口单出口型的，其中，$N'\subseteq N$，$E'=E\cap(N'\times N')$，还存在一些边 $e, e'\in E$，$E\cap((N\backslash N')\times N')=\{e\}$，$E\cap(N'\times(N\backslash N'))=\{e'\}$，*e* 为入边，*e*′为出边。设 $n\in N$，注：设 *set* 是一个集合，*firstElement*(*set*)是集合的第一个元素；*nextElement*(*set*)是集合的当前的下一个元素；$n.postNodes^*$ 是 *n* 的所有连续后继节点；$n.preNodes^*$ 是 *n* 的所有连续前继节点。

算法描述如下：

(1)遍历寻找节点 *n*，满足 $n\in SP$，则转步骤(2)。

(2)寻找 *n* 的每条出口路径的汇合节点 $join=\{j|n'\in n.postNodes\cap\forall n'(j\in n'.postNodes^*)\cap j\in JI\}$。如果 *join* 不为空，设 *j*=*firstElement*(*join*)，则转步骤(3)。

(3)得到子模型 $F=(N', E', T', ST', EN', AC', JI', SP')$，$N'=\{n'|(n'=n\cup n'\in n.postNodes^*)\cap n'\notin j.postNodes^*\}$，$E'=E\cap(N'\times N')$，$T'=T$，$ST'=n$，$EN'=j$，$AC'=\{n'|n'\in AC\cap n'\in N'\}$，$JI'=\{n'|n'\in JI\cap n'\in N'\}$，$SP'=\{n'|n'\in SP\cap n'\in N'\}$。

(4)检查 *F* 所有节点，除开始节点外的前继节点，是否都包括在模型中，如果 $\exists n''(n'\in N'\wedge n'\neq start\wedge n''\in n'.preNodes\wedge n''\notin N')$，则说明子模型结构不正确；如果 *nextElement*(*join*)不为空，则 *j*=*nextElement*(*join*)，转步骤(3)，否则 *n*=*firstElement*(*j.postNodes*)，转步骤(1)；如果正确，则转步骤(5)。

(5)将 *G* 中的 *F* 替换成活动节点，对 *F* 进行验证，具体步骤见 3.2 节，如果验证子模型是不正确的，则对 *F* 进行诊断，具体步骤见 3.3 节，*n*=*firstElement*(*j.postNodes*)，转步骤(1)。

### 3.2 模型验证

采用 Sadiq 和 Orlowska 的图化简方法[4]来判断模型是否正确。其思想是移除所有绝对正确的结构块，保留冲突。运用化简规则将结构正确的模型逐步化简，而结构不正确的则不能化简。化简过程比诊断过程简易，将子模型先化简验证后诊断，可减少诊断次数，提高效率。有 5 种化简规则：终端化简，顺序化简，邻近化简，闭合化简和交叠化简。设 $G=(N, E, T, ST, EN, AC, JI, SP)$，$n\in N$。

(1)终端化简：如果 *n.din*+*n.dout*=1，移除 *n*。

(2)顺序化简：如果 *n.din*=1 并且 *n.dout*=1，移除 *n*。

(3)邻近化简：如果 $n\in SP$ 并且 $n'\in n.preNodes$，*n*′*.type*=*n.type*，则令 $e\in n.outEdges$，$e'\in n.inEdges$，*e.fromNode*=*n*′，移除 *n* 和 *e*′。如果 $n\in JI$ 并且 $n'\in n.postNodes$，*n*′*.type*=*n.type*，则令 $e\in n.outEdges$，$e'\in n.inEdges$，*e*′*.toNode*=*n*′，移除 *n* 和 *e*。

(4)闭合化简：设 $e\in E$，如果 $\exists e'(e'\in E\wedge e.fromNode=e'.fromNode\wedge e.toNode=e'.toNode\wedge e.fromNode.type=e.toNode.type\wedge e\neq e')$，移除 *e*′。

(5)交叠化简：设 $n\in N$，如果 $\exists n'(n'\in N\wedge n.postNode=n'.postNode\wedge n.preNode=n'.preNode\wedge n.type=n'.type\wedge n\neq n')$，移除 *n*′。

### 3.3 模型诊断

本文借鉴文献[5]提出的基于 Integer Programming 诊断工作流的方法，在模型诊断中将模型翻译为包含一系列的 0/1 线性约束的 IP 模型，其中每个节点 *n* 都有一个 IP 变量 *IP*(*n*)，赋值为一个整形值 0 或 1。赋值为 1 的节点和边在执行路径内，为 0 的不在执行路径内。利用 IP 规范中的约束集来对每条执行路径的每个 join 进行测试，为不同的 join 添加其不会导致错误的条件约束，如果执行路径不满足某个 join 的特定约束集，说明在此 join 节点上有错误。反之，如果满足，说明此节点是正确的。

算法描述如下：

(1)为模型 $G=(N, E, T, ST, EN, AC, JI, SP)$ 定义 IP 规范约束集，设 $n\in N$：

IP1：*n.type*∈{start}：*IP*(*n*)=1；

IP2：*n.type*∈{activity, AND Splits, XOR Splits, end}：*IP*(*n.preNodes*)−*IP*(*n*)=0；

IP3：*n.type*∈{activity, AND Joins, XOR Joins, start}：*IP*(*n.postNodes*)−*IP*(*n*)=0；

IP4：*n.type*∈{AND Splits}：$\sum_{n'}^{n'\in n.postNode} IP(n')-n.dout\times IP(n)=0$；

IP5：*n.type*∈{XOR Splits}：$\sum_{n'}^{n'\in n.postNode} IP(n')-IP(n)=0$。

(2)遍历每条执行路径，分别检验路径上的每个 join 节点。AND Joins 节点，添加节点的约束 IP6：

IP6：$\sum_{n'}^{n'\in n.preNode} IP(n')=n.din\times IP(n)$；

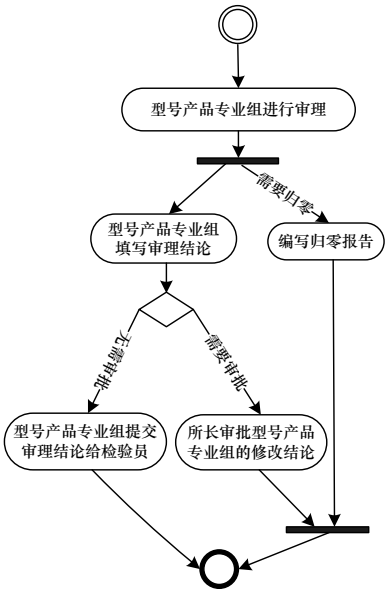
XOR Joins 节点，添加节点的约束 IP7：

IP7：$\sum_{n'}^{n'\in n.preNode} IP(n')=IP(n)$

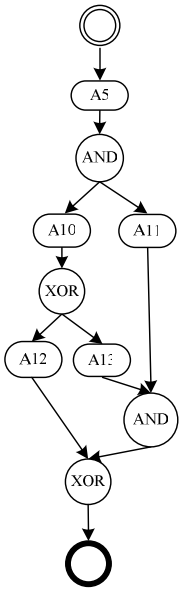
检验执行路径是否符合约束集，若符合，则此 join 节点正确；若不符合，则说明此 join 节点会导致冲突；AND Joins 节点会导致死锁；XOR Joins 节点会导致乏同步。

## 4 应用验证

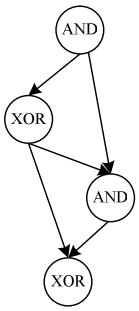
为验证本文所提出诊断方法的有效性，在此以某企业内部的质量管理过程为例。该过程是对生产中的不合格品进行审理，经过审理小组、常设机构和型号产品专业组的审理。具体如图 2 所示。




(a)型号产品专业组的审理活动图




(b)型号产品专业组的审理模型




(c)验证结果

**图 2　诊断过程**

诊断过程是将原始活动图预处理为标准模型，限于篇幅，选取部分活动图，型号产品专业组的审理活动图如图 2(a)所示，转换为模型如图 2(b)所示。将模型分解后，对子模型进行验证，型号产品专业组的审理模型是总模型的子模型，对其进行验证时，验证失败，如图 2(c)所示。随后进行诊断，得出诊断信息：A13 与 A11 的相交点 AND Joins 处存在冲突，可能引发死锁；A12 的下一个 XOR Joins 处存在冲突，可能引发乏同步。即当型号产品专业组填写审理结论后，选择不需要审批时，编写归零报告就不能进行下去，导致图失去可靠性，后面的同步条不能到达，导致图失去活性。

## 5　结束语

本文提出一种 UML 活动图模型的正确性诊断方法，并以某企业的内部质量管理过程为例对该方法进行应用验证，结果表明了该方法的有效性和实用性。但本文只涉及到 AND 和 XOR 2 种连接类型，实际中可能还有 OR 和(M/N)AND 等多种类型，因此，下一步工作将对模型的多样化进行研究，并将该方法运用到更复杂和大型的系统建模中，使其得到进一步完善。

## 参考文献

[1] 褚红伟, 赵银亮, 葛　玮. 分布工作流过程模型及其实现[J]. 计算机工程, 2009, 35(14): 55-57.

[2] van der Aalst W M P. Verification of Workflow Nets[C]//Proc. of the 18th International Conference on Application and Theory of Petri Nets. Toulouse, France: [s. n.], 1997.

[3] Vanhatalo J, Völzer H, Leymann F. Faster and More Focused Control-flow Analysis for Business Process Models Through SESE Decomposition[C]//Proc. of ICSOC'07. Vienna, Austria: [s. n.], 2007.

[4] Sadiq W, Orlowska M E. Analyzing Process Models Using Graph Reduction Techniques[J]. Information Systems, 2000, 25(2): 117-134.

[5] Eshuis R, Kumar A. An Integer Programming Based Approach for Verification and Diagnosis of Workflows[J]. Data & Knowledge Engineering, 2010, 69(8): 816-835.

编辑　金胡考

(上接第 46 页)

的语义异构问题。目前该框架已经成功应用在参数估计系统中，在不影响数据集成效率以及正确性的同时，更好地保证了数据集成过程数据的一致性。

本文的不足之处是没有为实例数据设计映射发现策略，因为电力系统数据的多样性以及复杂性，所以暂时还不能提取出实例数据的一般特征。下一步工作将继续深入研究映射匹配算法，为实例数据制定映射匹配策略，实现不需人工干预自动完成异构数据源的集成。

## 参考文献

[1] Doan A, Madhavan J, Domingos Petal. Learning to Match Ontologies on the Semantic Web[J]. The VLDB Journal, 2003, 12(4): 303-319.

[2] 史　斌, 方丽英. 基于本体的概念语义相似度度量[J]. 计算机工程, 2009, 35(19): 83-85.

[3] 屈振新, 唐胜群. 信息集成中冲突的语义解决方案[J]. 计算机科学, 2010, 37(1): 167-169.

[4] 李　熙. 基于 WordNet 的本体映射研究[D]. 长沙: 中南大学, 2008.

[5] Andrew M, Egenhofer M J. Determining Semantic Similarity Among Entity Classes from Different Ontologies[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(2): 442-456.

[6] 李艳霞. 面向信息集成的语义异构消除技术研究[D]. 兰州: 西北师范大学, 2009.

编辑　金胡考