基于 Petri 网的工作流建模与正确性分析*)

周福明 吴 斌 顾 庆 陈道蓄
(南京大学软件新技术国家重点实验室 南京210093)

摘 要 目前用于工作流建模和分析的工具很多,Petri 网以其坚实的数学基础、直观的图形表示受到大家的青睐。本文介绍了如何使用 WF-Net 建立工作流模型,并根据 Aalst 给出的 WF-Net 正确性定义提出了一个算法,用来检查该 WF-Net 的正确性。

关键词 工作流模型,Petri 网,WF-Net,基本 WF-net,正确性检查

Modeling and Correctness Checking of Petri-Net-Based Workflow

ZHOU Fu-Ming WU Bin GU Qing CHEN Dao-Xu
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing210093)

Abstract There are a lot of modeling and analysis tools available for workflow, but Petri nets is more preferable because of its solid mathematical foundation and graphical nature. This paper introduces the notion of WF-Net and the method modeling workflow with WF-Net. At last, this paper introduces the definition of correctness offered by Aalst, and presents an algorithm based on it to check the correctness of the WF-Net.

Keywords Workflow model, Petri nets, WF-Net, Basic WF-net, Correctness checking

1 引言

工作流的概念起源于生产组织和办公自动化领域。它是对日常工作中具有固定程序的活动而提出的一个概念[3]。随着计算机网络技术和分布式数据库技术的迅速发展,多机协同工作技术的日趋成熟,工作流技术的应用范围日益广阔。至今,工作流技术已成功地应用到图书馆、医院等社会生活的各个方面。工作流技术对软件开发组织也有特别的支持,工作流技术可用于项目管理、业务建模、需求分析、设计、实现以及测试。

近年来,对工作流模型描述和分析方法的研究得到普遍重视。用于工作流建模和分析的工具也越来越多,使用 Petri 网建立工作流模型也被越来越多的人接受。选择 Petri 网建立工作流模型有以下原因[1]:

·形式化的语义。Petri 网描述的工作流过程有着清楚、精确的含义。

·直观的图形表示。Petri 网是一个图形化的语言,易于学习,也易于与最终用户交流。

·表达能力强。Petri 网支持工作流建模所需的所有原语。

·Petri 网有着坚实的数学基础。

本文简要介绍了工作流和 Petri 网的基本概念,描述了使用 Petri 网建立工作流模型(工作流网 WF-Net)的方法,并给出了一个实例:软件测试过程的 Petri 网表示。最后,本文根据 Aalst 提出的关于 WF-Net 正确性的定义,给出了一个算法,用来验证 WF-Net 的正确性。

2 工作流

工作流管理联盟给出的工作流定义[3]是:工作流是一类能够完全或者部分自动执行的经营过程,根据一系列过程规则,文档、信息或者任务能够在不同的执行者之间传递、执行。工作流中两个最基本的元素是活动和活动之间的连接关系。活动对应于经营过程中的任务,主要是反映经营过程中的执行动作或操作。活动之间的连接关系代表了经营过程的规则和业务流程。以下我们给出一个工作流的例子。

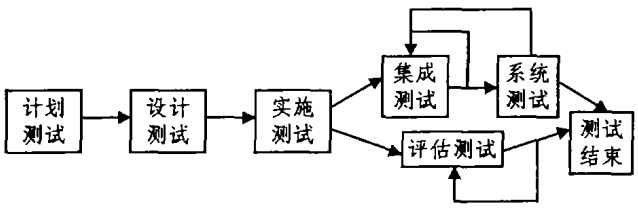

图1 软件测试工作流

上面例子描述的是软件的测试过程。例子中的方框表示测试过程中的任务,它们对应于工作流中的活动。方框之间的连接弧表示各个任务之间的关联。

有很多方法可以用来进行工作流模型的定义与描述。目前较为广泛接受的建模语言有 CIMOSA 的经营过程描述语言、工作流管理联盟 WfMC 定义的工作流描述语言、Keller 等人提出的 EPCM 模型等,这些工作流描述语言的描述形式与程序设计语言中的语义结构的定义方式类似。当所需建模分析的经营过程比较复杂,并存在并发、冲突等情形时,就需要采用形式化程度更高、描述能力更强的方法,如 Petri 网方法。

3 Petri 网

1962年联邦德国的 Carl Adam Petri 在他的博士论文《用自动机通信》中首次使用网状结构模拟通信系统,这种系统模型就是 Petri 网的雏形。现在 Petri 网已经发展成为一个庞大的研究领域。

*)本文得到国家863项目支持,项目编号:2001AA113090。周福明 硕士研究生,研究方向:工作流系统,软件过程管理。吴 斌 硕士研究生,研究方向:工作流,软件过程管理。顾 庆 博士,副教授,研究方向:分布式计算。陈道蓄 博士生导师,研究方向:分布式计算与并行处理。

定义1(Petri 网) 三元组$(P,T;F)$称为 Petri 网的充分必要条件是[5]:1)$P\cap T=\varnothing$;2)$P\cup T\neq\varnothing$;3)$F\subseteq P\times T\cup T\times P$。其中 P 是库所的有限集,T 是变迁的有限集,F 是流关系。

Petri 网可用一个有向图来表示,其中库所只能与变迁相连,变迁只能与库所相连。图2给出了一个 Petri 网的一个例子[6]。图中,库所用圆圈表示,变迁用矩形表示,元素$(x,y)\in F$ 表示有一条有向弧从 x 指向 y。例如$(p_1,t_1)\in F$ 表示为一条连接 P_1与 t_1的有向弧。该网的 P,T,F 分别为:

$P=\{P_1,P_2,P_3,P_4,P_5\}$

$T=\{t_1,t_2,t_3,t_4,t_5\}$

$F=\{(P_1,t_1),(t_1,P_2),(t_1,P_3),(t_1,P_4),(P_3,t_2),(t_2,P_3),(P_2,t_2),(P_2,t_3),(P_4,t_2),(P_4,t_4),(t_3,P_4),(t_3,P_5),(t_4,P_5),(P_5,t_5)\}$

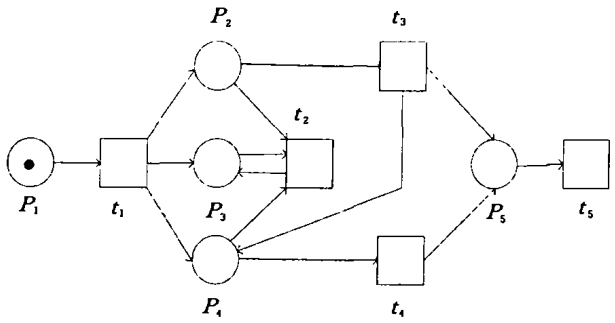

图2 一个 Petri 网例子

定义2(前集和后集) 设 $x\in P\cup T$,x 的前集(pre-set)或输入集和 x 的后集(post-set)或输出集分别如下:

$$\cdot x=\{y\mid(y,x)\in F\}$$

$$x^{\cdot}=\{z\mid(x,z)\in F\}$$

变迁的前集和后集元素都是库所,库所的前集和后集都是变迁,前、后集可以是空集。例如图2中 t_1的前集是$\{P_1\}$,t_1的后集是$\{p_1,p_2,p_3\}$,p_3的前集是$\{t_1,t_2\}$,p_1的前集是空集,形式化地表示如下所示:

$\cdot t_1=\{P_1\}$ $t_1^{\cdot}=\{P_2,P_3,P_4\}$

$\cdot P_3=\{t_1,t_2\}$ $\cdot P_1=\varnothing$

定义3(状态) 也称作标识,表示令牌在库所中的分布情况。可以使用一个向量来表示 Petri 网的状态。向量的维数表示该 Petri 网中所含的库所数。

例如(1,2,1,0)表示该 Petri 网中一共包含四个库所,库所 P_1中有一个令牌,P_2中有两个,P_3中有一个,P_4中没有。为了对两个状态进行比较,我们引入偏序关系(≤)。对任意两个状态 M_1,M_2,$M_1\leqslant M_2$当且仅当$\forall i\in[1..n]$ $M_1[i]\leqslant M_2[i]$。

定义4(变迁的触发) 1)变迁 t 是可触发的,当且仅当 t 的前集每个库所中都至少含有一个令牌;2)当变迁 t 触发时,t 消费掉前集每个库所中的一个令牌,并在后集每个库所中产生一个令牌。

例如,在图2所示的例子中,初始状态为 $M_0=(1,0,0,0,0)$ $\cdot t_1=\{P_1\}$且 P_1中含有一个令牌,所以 t_1可以触发,P_1中的令牌将被消费掉。又因为 $t_1^{\cdot}=\{P_2,P_3,P_4\}$,所以 t_1触发以后,将在 P_2,P_3,P_4中各产生一个令牌,状态变为 $M_1=(0,1,1,1,0)$。整个变迁发生过程可以表示为 $M_0\xrightarrow{t_1}M_1$。方便起见,当状态 M 下有变迁 t 可以触发时,称 M 是可执行的。

定义5(强连通) 一个 Petri 网是强连通的,当且仅当对任意一对节点(库所或变迁)x 和 y,存在一条从 x 到 y 的路径。

4 工作流的 Petri 网表示

描述工作流的 Petri 网称为工作流网(WF-net)。用 Petri 网对工作流建模比较直观:工作流中的任务用 Petri 网中的变迁表示,条件用库所表示,工作流实例用令牌表示,令牌的流动表示工作流的执行。

定义6(WF-net[1]) Petri 网 $PN=(P,T;F)$是工作流网(WF-net),当且仅当:

1) PN 有两个特殊的库所:i 和 o。i 是初始库所:$\cdot i=\varnothing$,o 是终止库所,$o^{\cdot}=\varnothing$。

2) 如果我们在此 PN 中加入变迁 t^*,t^* 连接 o 到 i(即$\cdot t^*=\{o\}$,$t^{*\cdot}=\{i\}$),那么此 Petri 网成为强连通的。

工作流管理联盟定义了四种基本的工作流路由结构[4]:

①顺序路由。表示两个任务之间存在时序依赖关系,在前一个任务完成之前,后一个任务不能执行。

②并行路由。表示并行的两个或多个任务之间可以任意顺序执行。

③选择路由。表示在流程的某一点,依执行结果选择一条路径继续执行。

④循环路由。表示某一个任务可能需要执行多次。

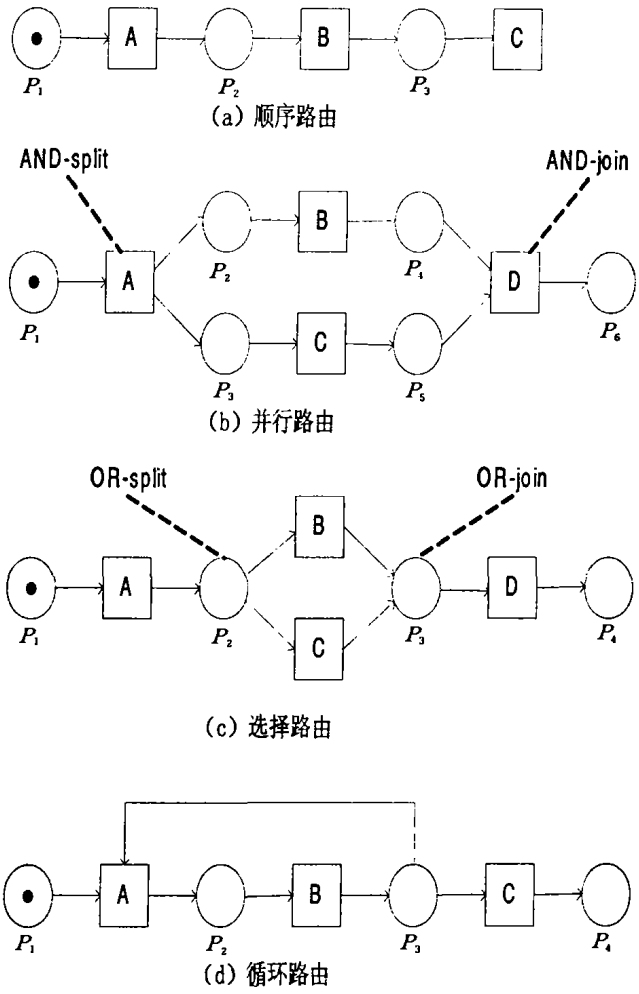

图3 四种基本路由结构的 Petri 网表示

这四种结构是工作流执行的基本结构,所有工作流的执行结构可以由这四种基本结构组合而成。以上四种基本结构可以用图3所示的 WF-Net 描述。在这个描述中,我们引入四种构造模块,分别是与分叉(AND-split)、与合并(AND-join)、或分叉(OR-split)、或合并(OR-join)。与分叉和与合并用来表

示一个并行执行的过程，或分叉和或合并用来表示一个选择执行过程。

使用 WF-Net 对前面我们提到的软件测试过程建模，可以得到图4所示的 Petri 网。

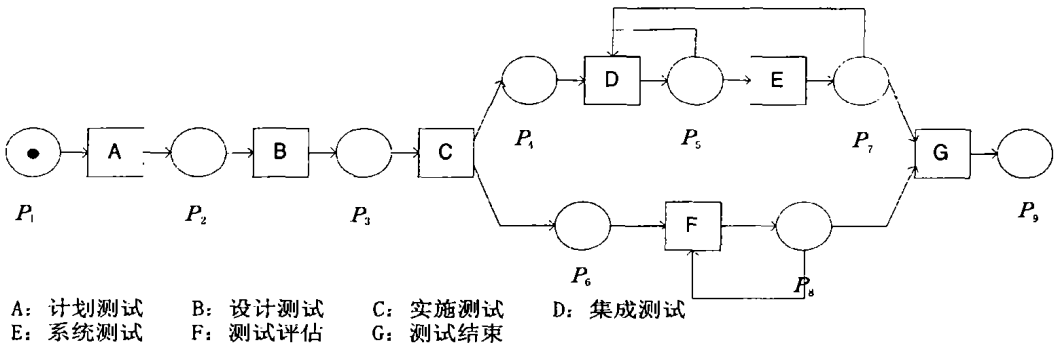

图4　软件测试过程 Petri 网表示

5　WF-Net 的正确性验证

目前可供选择的工作流管理系统很多，但是大多数没有提供正确性验证机制。工作流过程定义之后，不经过彻底的检查即投入运行，这常常导致到运行时才发现定义错误，而此时的修改需要极高昂的费用，还会引起用户的不满。本节介绍一个基于 WF-Net 的算法，用来检验工作流定义的正确性。

定义7（正确性[1]）　Petri 网表示的工作流过程 $PN=(P,T;F)$ 是正确的，当且仅当：

1）对任意 i 可达的状态 M，存在一个从状态 M 到状态 o 的触发序列。形式化表示为：

$$\forall_M(i\xrightarrow{*}M)\Rightarrow(M\xrightarrow{*}o)$$

2）状态 o 是唯一结束状态，当一个工作流实例运行结束时，只有库所 o 中有一个令牌，其它库所为空。形式化表示为：

$$\forall_M(i\xrightarrow{*}M\wedge M\geqslant o)\Rightarrow(M=o)$$

3）PN 中没有死变迁，即任意一个变迁都可被执行。形式化表示为：

$$\forall_{t\in T}\exists_{M,M'}i\xrightarrow{*}M\xrightarrow{*}M'$$

正确性定义表明：所有路径的归宿都是最终态，并且在遍历从 i 到 o 的所有路径时，每个变迁都触发过，每个库所都经历过。下面我们根据 Aalst 的定义给出一个算法，用来检查一个 Petri 网表示的工作流是否满足定义7规定的正确性。

在给出算法之前，我们引入向量加和向量减的概念。

定义8（向量加（+）和向量减（−））　已知向量 X,Y,A,B，令 $X=A+B,Y=A-B$，则：

$$X_i=\begin{cases}1 & \text{其他}\\ 0 & A_i=0\wedge B_i=0\end{cases}$$

$$Y_i=\begin{cases}1 & A_i=1\wedge B_i=0\\ 0 & \text{其他}\end{cases}$$

5.1　算法

算法思路：

a. 采用广度优先算法，从初始状态集合 $S=\{M_0\}$ 出发。

b. 运行 S 中的状态，得到新状态集合 S'，已经出现过的状态不在 S' 中。同时记录触发过的变迁、经历过的库所。

c. 把 S' 赋值给 S，重复 b，直到 S 中的元素都没有变迁可以发生。

d. 如果 S 中的元素都是形如(0,0,…,1)的终止态，且所有变迁都触发过，所有库所都经历过，则该 Petri 网表示的工作流是正确的，否则存在错误。

我们先说明一下在算法中用到的标记：RULE 是规则集，用来存放规则，每一个规则对应一个变迁。规则 rule, 对应变迁 T_i，形如(M_k,M_j)，其中 M_k,M_j 均为 n 维向量。状态 M 下，变迁 T_i 触发到达状态 M' 可形式化表示为：

1）$M_k\leqslant M$；2）$M'=M-M_k+M_j$

M_0是初始状态（n 维向量）。*History* 是状态集，用于存放已经出现过的工作流状态，最初 *History* 中只有初始态 M_0。*Now* 是状态集，用于存放现在可以执行的状态，最初 *Now* 中只有一个元素 M_0。*New* 是状态集，用于存放执行 *Now* 中元素时产生的新状态，最初 *New* 为空。*Transitions* 是变迁集，用于存放触发过的变迁，最初 *Transitions* 为空。*Places* 是一 n 维向量，用于存放经历过的库所，*Places* [i]表示第 i 个分量，*Places* [i]=0表明库所 i 还没有经历过，*Places* [i]=1表明库所 i 已经经历过了，最初 *Places* 为(1,0,…,0)。

算法：

(1)初始化：$M_0=(1,0,\cdots,0)$；*History*={ M_0}；*Now*={ M_0}；*New*=∅；*Transitions*=∅；*Places*=(1,0,…,0)。

(2)∀ $M\in$ *Now* 有两种可能：

a)此状态下有 k 个变迁可以实施$(t_1,t_2,\cdots,t_k)$。分别触发，得到状态 $M_1,M_2,\cdots,M_k$。

若 $M_i\notin history(i\in[1,k])$，则

History=*History*+{M_i}

New=*New*+{M_i}。

Transitions=*Transitions*+{$t_1,t_2,\cdots,t_k$}

Places=*Places*+(M_i-M)。（此处为向量加减）

b)此状态下没有变迁可以实施

返回，取下个元素　若 $M=(0,0,\cdots,1)$

报错，并退出程序　若 $M\neq(0,0,\cdots,1)$

直到 *Now* 中元素都已访问。

(3)*Now*=*New*　*New*=∅。

(4)重复(2)直到 *Now* 为空(即上一轮没有产生新状态)。

(5)算法结束。

如果程序运行过程中报错，并退出程序，表明正确性检查没有通过；如果程序正常运行结束，则检查向量 *Places* 和集合 *Transitions*，如果 *Places*=(1,1,…,1)& *Transitions*=T，表明该 Petri 网表示的工作流是正确的。

5.2　实例分析

以前面我们建模的软件测试过程为例，运行该程序：

初始状态：*Now*={M_0(1,0,0,0,0,0,0,0,0,0)}；*History*=

$\{M_0\}$;$Transitions=\varnothing$;$Places=(1,0,0,0,0,0,0,0,0,0)$。

步骤1:只有变迁 A 可以触发,得到状态 $M_1(0,1,0,0,0,0,0,0,0,0)$,其中 M_1 是新状态:

$Now=\{M_1\}$

$History=\{M_0,M_1\}$

$Transitions=\{A\}$

$Places=(1,1,0,0,0,0,0,0,0,0)$。

步骤2:只有变迁 B 可以触发,得到状态 $M_2(0,0,1,0,0,0,0,0,0,0)$,其中 M_2 是新状态:

$Now=\{M_2\}$

$History=\{M_0,M_1,M_2\}$

$Transitions=\{A,B\}$

$Places=(1,1,1,0,0,0,0,0,0,0)$。

步骤3:只有变迁 C 可以触发,得到状态 $M_3(0,0,0,1,0,1,0,0,0,0)$,其中 M_3 是新状态:

$Now=\{M_3\}$

$History=\{M_0,M_1,M_2,M_3\}$

$Transitions=\{A,B,C\}$

$Places=(1,1,1,1,0,1,0,0,0,0)$。

步骤4:有变迁 D 和 F 可以触发,得到状态 $M_4(0,0,0,0,1,1,0,0,0,0)$,状态 $M_5(0,0,0,1,0,0,0,1,0,0)$,其中 M_4,M_5 是新状态:

$Now=\{M_4,M_5\}$

$History=\{M_0,M_1,M_2,M_3,M_4,M_5\}$

$Transitions=\{A,B,C,D,F\}$

$Places=(1,1,1,1,1,1,0,1,0,0)$。

步骤5:有变迁 D、E、F(在状态 M_4 下触发),变迁 D 和 F(在状态 M_5 下触发)可以触发,分别得到状态 M_4,$M_6(0,0,0,0,0,1,1,0,0,0)$,$M_7(0,0,0,0,1,0,0,1,0,0)$,M_7,M_5,其中 M_6,M_7 是新状态:

$Now=\{M_6,M_7\}$

$History=\{M_0,M_1,M_2,M_3,M_4,M_5,M_6,M_7\}$

$Transitions=\{A,B,C,D,E,F\}$

$Places=(1,1,1,1,1,1,1,1,1,0)$。

步骤6:有变迁 D,F(在状态 M_6 下触发),变迁 D,E,F(在状态 M_7 下触发)可以触发,分别得到状态 M_4,$M_8(0,0,0,0,0,0,1,1,1,0)$,M_7,M_8,M_9,其中 M_8 是新状态:

$Now=\{M_8\}$

$History=\{M_0,M_1,M_2,M_3,M_4,M_5,M_6,M_7,M_8\}$

$Transitions=\{A,B,C,D,E,F\}$

$Places=(1,1,1,1,1,1,1,1,1,0)$。

步骤7:有变迁 D,F,G 可以触发,分别得到状态 M_7,M_8,$M_9(0,0,0,0,0,0,0,0,1)$,其中 M_9 是新状态:

$Now=\{M_9\}$

$History=\{M_0,M_1,M_2,M_3,M_4,M_5,M_6,M_7,M_8,M_9\}$

$Transitions=\{A,B,C,D,E,F,G\}$

$Places=(1,1,1,1,1,1,1,1,1,1)$。

步骤8:没有变迁可以发生,结束。

由以上程序运行可知,该程序顺利退出,且 $Transitions=T$ & $Places=(1,1,1,1,1,1,1,1,1,1)$,所以该 Petri 网表示的软件测试过程满足正确性。

5.3 算法分析

算法时间与可能出现的状态成正比。此算法原则上可以对任意 WF-net 进行检查,但是对任意的 WF-net 而言,可能状态数与库所成指数级增长。下面我们来考察一类特殊的 WF-net,基本 WF-net。

定义9(基本 WF-net) 基本 WF-net 由顺序、选择、并行、循环结构构成,选择与并行可以相互包含,但不能交叉。

定理1 算法对基本 WF-net PN 可以在多项式时间内完成正确性检查。

证明:对结构作归纳

1)顺序结构。假设 PN 是顺序结构的,有库所 n 个,则可能状态为 n 种,算法在库所个数的多项式时间内结束。

2)选择结构。假设 PN 是选择结构的,有2个分支子网(多个分支的情况可以化归到两个分支)。分支1有 n_1 个库所,p_1 种状态,p_1 是 n_1 的多项式;分支2有 n_2 个库所,p_2 种状态,p_2 是 n_2 的多项式。则 PN 一共有 $n=n_1+n_2$ 个库所,$p=p_1+p_2$ 种状态。显然 p 是 $n(n_1+n_2)$ 的多项式,引入选择结构后,算法仍可以在多项式时间内结束。

3)并行结构。假设 PN 是并行结构的,有2个分支子网(多个分支的情况可以化归到两个分支)。分支1有 n_1 个库所,p_1 种状态,p_1 是 n_1 的多项式;分支2有 n_2 个库所,p_2 种状态,p_2 是 n_2 的多项式。则 PN 一共有 $n=n_1+n_2$ 个库所,$p=p_1\times p_2$ 种状态。显然 p 是 $n(n_1+n_2)$ 的多项式,引入并行结构后,算法仍可以在多项式时间内结束。

4)循环结构。循环结构并不引入新的状态,所以,引入循环结构后,算法还是在多项式时间内结束。

证明结束。

总结 Petri 网自提出以来,已经发展成为一个庞大的研究领域。它的形式化的语义、直观的图形表示使得它成为工作流建模的良好工具。本文根据 Aalst 的定义提出了一个算法,用来检查一个 WF-Net 的正确性。但是仍然存在以下问题:

1) 算法只检查了 WF-Net 语法上的正确性,并没有对语义上的正确性进行检查。

2) 对于大型的业务过程来说,其规模极其庞大,其建模也十分复杂,验证尤其困难。

3) 正确性只是一个基本要求,对一个工作流模型来说,它的效率和可维护性也是十分重要的。

今后的工作将在这几个方面展开。

参 考 文 献

1 van der Aalst W M P. The Application of Petri nets to Workflow Management [J]. The Journal of Circuits, Systems and Computers, 1998, 8(1): 21~66

2 van der Aalst W M P. A class of Petri net for modeling and analyzing business processes: [Computing Science Reports 95/26]. Eindhoven University of Technology, Eindhoven, 1995

3 范玉顺. 工作流管理技术基础——实现企业业务过程重组、过程管理与过程自动化的核心技术. 北京:清华大学出版社,施普林格出版社,2001

4 Workflow Management Coalition. The workflow reference model. WFMC TC00-1003, 1994

5 袁崇义. Petri 网原理. 北京:电子工业出版社,1998

6 Reisig W. Petri nets: an introduction. volume 4 of Monographs in theoretical computer science: an EATCS series. Springer-Verlag, Berlin, 1985

7 Kruchten P. Rational 统一过程引论(第二版 影印版). 北京:中国电力出版社,2003