第37卷 第15期 计 算 机 工 程 2011年8月
Vol.37 No.15 Computer Engineering August 2011

·软件技术与数据库· 文章编号：1000—3428(2011)15—0027—03 文献标识码：A 中图分类号：TP311

基于虚拟执行的动态工作流演进算法

张 杰，吕 红，周立军，王丽娜

(海军航空工程学院基础实验部，山东 烟台 264001)

摘 要：为实现工作流管理系统中工作流演进的控制，解决流程实例动态调整的问题，给出工作流模型相关定义的形式化表示及正确性标准，提出基于数据流相关信息历史记录的虚拟执行算法，适用于模型结构变化及模型存在循环结构等情况。该算法能判断流程实例能否迁移，通过回退和撤销的策略保证工作流实例快速迁移。具体应用实例证明了迁移算法的正确性与可行性。

关键词：工作流；工作流模型；工作流演进；历史记录；虚拟执行

Dynamic Workflow Evolution Algorithm Based on Virtual Execution

ZHANG Jie, LV Hong, ZHOU Li-jun, WANG Li-na

(Department of Basic Experiment, Naval Aeronautical Engineering Institute, Yantai 264001, China)

【Abstract】In order to achieve workflow system for the control of workflow evolution and to solve the problem of dynamic adjustment of the workflow model, this paper presents formal specification and correctness criteria of workflow model briefly. Based on the execution history of process instance, an efficient virtual execution algorithm involving flow change is proposed, which is also applied to loop structure. The algorithm can determine whether an instance can be smoothly migrated, through the means of rollback and aborting, it guarantees that the work done is valid at maximum and workflow instances are migrated rapidly and automatically. A case is introduced to validate the correctness and applicability of the algorithm.

【Key words】workflow; workflow model; workflow evolution; history record; virtual execution

DOI: 10.3969/j.issn.1000-3428.2011.15.007

1 概述

在工作流管理系统中，业务过程优化、重组导致工作流模型变更，必然影响基于原模型正在运行的工作流实例，导致工作流版本管理混乱、工作流实例运行脱节、工作流异常等一系列问题，因此，必须及时有效地调整工作流实例。

工作流演进已经取得众多研究成果，提出的方法可分为基于模型结构一致性原则和基于历史事件(记录)一致性原则[1]，基于模型结构一致性的方法比较典型的是 Aalst 采用 Petri 网对模型建模，只需计算 1 次动态变更区域即可对所有实例做出能否迁移的判断，其缺点是迁移的标准仅以实例的当前状态是否合法为依据，没有参考流程实例的历史执行记录及相关数据信息。历史事件一致性原则主要基于虚拟执行的迁移规则，以文献[2]中的公理(下文称为公理 1)为迁移准则，文献[3]给出一种基于虚拟执行的算法，没有考虑活动历史事件的时间顺序关系和活动节点修改特点[4]，该算法使用集合作为数据存储单元，不适用于循环结构的工作流模型。本文提出一种基于虚拟执行的动态工作流演进算法，研究如何将工作流实例调整(迁移)，使其按新的流程定义模型运行。

2 工作流相关定义

在参考 WFMC 工作流定义及文献[2-4,5-6]的基础上，首先给出工作流相关定义。

2.1 工作流模型 WF Schema

工作流模型 WF Schema 是实际业务过程抽象出来的活动的集合，描述了各个活动之间的偏序关系，可用六元组表示：$w=\{ID,N,NT,D,\sigma,\omega\}$。其中，$ID$ 是模型的唯一标识；N 为业务过程的活动集合；NT 是活动的类型集合 $NT(N)=\{START, END, TASK, XOR\_SPLIT, XOR\_JOIN, AND\_SPLIT, AND\_JOIN\}$，各个类型的活动图形化表示如图 1 所示；$D$ 为模型相关数据元素的集合；σ 是各个活动之间的顺序关系，一个模型的 σ 使用数据流向关系[2](Data Flow Relation, DFR)表示，如果任意 2 个活动存在边，则用 1 表示，否则为 0；ω 是模型条件的集合；$\omega(N)$为 N 转移的条件。

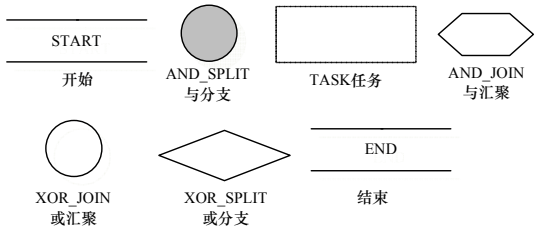


图 1 活动类型图形化表示

2.2 活动

活动是完成一定目标功能的最小单元，活动对应的数据操作称为数据流，按照活动是否操作数据单元，活动类型又可分为执行类型活动和非执行类型活动，执行类型包括任务活动和条件活动，执行类型活动 N 可能对应 n 个数据元素操作 $DU_{N_i}=\{DF|df=(d,para,dp)\}$。其中，$d \in D$；$dp \in \{read, write\}$；$(d,para,read)$表示输入参数 $para$ 的值从数据元素 d 读入；

作者简介：张 杰(1983－)，男，助理工程师，主研方向：工作流模型；吕 红，讲师；周立军、王丽娜，助理工程师

收稿日期：2011-01-07 **E-mail：**lazytr@sina.com

(*d*,*para*,*write*)表示当活动被执行完后，将 *para* 的值写入数据元素 *d*，为便于算法讨论，增加一种空操作类型活动 *NullNode*，该活动不对任何数据单元进行操作。

2.3 工作流实例

工作流实例是一个工作流过程的一次执行，形式化表示为：*I*=(*ID*, *W*,*NS*,*DV*,*H*)，其中，*ID* 是工作流实例的唯一标识；*W* 是工作流实例执行所基于的工作流模型；函数 $NS(N_i)$={*NOTACTIVITED*,*ACTIVITED*,*RUNING*,*COMPLETED*}标记了活动 N_i 的执行状态，*NOTACTIVATED* 表示活动没有被激活，是活动的初始化状态，*ACTIVATED* 表示活动已经激活，活动执行的前提条件已经满足，*RUNNING* 表示任务正在执行，*COMPLETED* 表示任务已完成；*H* 是 *I* 的历史执行记录，*H* 记录了活动对工作流相关数据读写的历史记录；*DV* 即为实例 *I* 相关数据单元的集合。本文给出一种适用于模型结构变化和循环结构的虚拟执行算法。

3 工作流模型的正确性分析

工作流模型的正确性和可靠性是工作流系统健壮性的基础，构造有效的模型可以使工作流实例避免异常和死锁现象，模型验证分析分为定性分析和定量分析，前者侧重于所定义工作流在逻辑上的正确性，消除异常结构，如死锁，后者主要考察工作流性能，如平均完成时间、服务水平和能力利用，由于工作流模型的复杂性，因此目前还没有有效的算法可以对工作流模型的正确性进行定性分析，为了便于正确讨论，首先给出本文涉及工作流可达状态定义。

定义 任意工作流实例活动 $NT(NX) \in$ {*ACTIVITED*, *RNNING*}的集合称为该工作流模型 *W* 的一个可达状态 *S*。所有可达状态的集合称为工作流模型的可达状态集。

工作流可达性能够表达被建模过程的行为，通常使用可达图描述工作流的可达性，可得到工作流程中某活动是否能被执行到及其执行次序。本文在总结文献[7-8]的基础上，给出模型结构和相关数据正确性的一些准则，该准则是保证工作流模型正确性的必要条件。

3.1 模型结构正确性

模型结构正确性可从 4 个方面进行分析。

(1)存在唯一 N_x 与 N_y 满足 $NT(N_x)=START$，$NT(N_y)=END$：

$$(\sum_{j=0}^{K} DFR_{[j][x]}=0 \wedge \neg\exists_{N_i}(\sum_{j=0}^{K} DFR_{[j][i]}=0 \wedge (i\neq x))) \wedge (\sum_{j=0}^{K} DFR_{[y][j]}=0 \wedge \neg\exists_{N_i}(\sum_{j=0}^{K} DFR_{[i][j]}=0 \wedge (i\neq y)))$$

其中，*K* 为模型活动个数；$\sum_{j=0}^{K} DFR_{[j][x]}$ 为活动 N_x 输入连接弧的度数(入度)；$\sum_{j=0}^{K} DFR_{[y][j]}$ 为活动 N_y 输出连接弧的度数(出度)。

(2)任意活动 N_x 存在于开始活动 N_{start} 与结束活动 N_{end} 之间的一条路径上：

$$\forall_{N_x}(\exists_{p_0(N_{\text{start}},\cdots,N_x)}(N_{\text{start}} \xrightarrow{p_0} N_x) \wedge \exists_{p_1(N_x,\cdots,N_{\text{end}})}(N_x \xrightarrow{p_1} N_{\text{end}}))$$

(3)任意可达状态 S_i 都可以到达结束活动 S_{end}：

$$\forall_{S_i}(S_{\text{start}} \xrightarrow{*} S_i) \Rightarrow S_i \xrightarrow{*} S_{\text{end}}$$

(4)若结束活动的状态为 *COMPLETED*，实例执行记录的所有活动的状态为 *COMPLETED*：

$$\exists_{N_{\text{end}}}(NS(N_{\text{end}})=COMPLETED) \Rightarrow \forall_{N_x \in W}(NS(N_x)=COMPLETED)$$

3.2 相关数据正确性

(1)N_x 读取数据 *d* 之前，*d* 应该已被赋值(初始化或 N_x 前驱写入)，即 $val(d)\neq UNDEFINED$：

$$\forall_d \exists_{N_x}((N_x,d,read)\in DU_{N_x}) \Rightarrow \exists_{N_y}(N_y \in Pred(W,N_x) \wedge (N_y,d,write)\in DU_{N_y}) \vee Inited(d)$$

(2) N_x 与 N_y 对数据 *d* 进行写操作，则 N_x 和 N_y 满足：

$$N_x \in Pred^*(W,N_y) \text{ 或 } N_y \in \Pr ed^*(W,N_x)$$

$$\exists_{N_x,N_y}((N_x,d,write)\in UD_{N_x} \wedge (N_y,d,write)\in UD_{N_y}) \Rightarrow N_x \in \Pr ed^*(N_y) \vee N_y \in Pred^*(N_x)$$

4 虚拟执行规则

虚拟执行规则即通过设计虚拟执行算法，根据实例执行的历史信息数据，在新工作流模型重新虚拟调度，公理 1 给出了工作流实例迁移的判断标准。

公理 1 一个基于工作流模型 *W* 的工作流实例 *I* 可以迁移到新的正确的工作流模型 *W'*，当且仅当实例 *I* 在 *W'* 可以产生与 *W* 相同的历史执行记录。

该公理给出了工作流实例可以迁移的充要条件，如何判别 *W'* 会产生一个相同的历史执行记录是实现迁移的关键，本文提出的虚拟执行算法是根据实例 *I* 在原模型 *W* 产生的历史执行记录，通过虚拟执行方式检测能否在新工作流模型 *W'* 上产生，假定 N_x 为实例 *I* 在虚拟执行中当前执行活动点(又称虚拟点)，判断工作流实例能否迁移，工作流迁移具体准则以及虚拟执行结束条件如下：

(1)若 N_x 不属于新工作流模型 *W'* 下一步待选活动列表 *L*。

(2)若 N_x 属于 *XOR_SPLIT* 类型活动，任取 $N_i \in Succ(W',N_x)$，相关数据都无法满足条件 $\omega(N_i)$。

(3)若 N_x 属于 *AND_JOIN* 类型活动，$Count(N_x) < \sum_{j=0}^{k} DFR_{[j][x]}$，表明 N_x 无法满足同步转移前置条件。

(4)根据实例历史执行记录，按照新的工作流模型 *W'* 能够完成虚拟执行操作，判定可以迁移。

本文提到历史执行记录包含模型定义的 7 种活动类型，除任务类型活动，其他类型活动不对相关数据单元做任何修改，但记录实例路由信息，在实例虚拟执行时路由信息可降低算法复杂度，当任务活动之间存在数据相关性时，可以更好地保证任务活动的前后顺序关系。本文没有采用文献[9]提出的简化循环算法处理历史记录，因为如出现文献[1]中列举的被修改部分位于循环结构的情况，则会导致实例满足迁移规则，但不符合公理的要求。

5 迁移算法

根据公理的虚拟执行标准，结合上述虚拟执行终止条件，给出一种检测实例能否迁移的算法。

首先初始化算法需要的数据：

(1)初始化数组 *HD*，记录实例 *I* 历史执行活动。

(2)初始化列表 *L*，记录虚拟执行下一步待选取的活动列表，初始化时加入开始活动。

(3)初始化工作流实例 *I* 相关数据 *TD*。

(4)初始化 $CP \leftarrow 0$ 与记录实例 *I* 虚拟执行当前的位置，原工作流模型和新工作流模型标记为 *W* 和 *W'*。

算法实现如下：

```
While(CP<HD.length){ Switch (NT(HD[CP]))
{
Case START:{ CP++;L 中删除元素 HD[CP];
将 Succ(W',HD[CP]) 插入到活动 L 列表中;
```

}

Case AND_SPLIT:{在 L 中遍历，若不存在 HD[CP];

GOTO QUTE_FLAG;

CP++;L 中删除元素 HD[CP];

将 Succ(W',HD[CP]) 插入到活动 *L* 列表中;

}

Case TASK:{在 L 中遍历，若不存在 HD[CP];

GOTO QUTE_FLAG; CP++;

if($\exists$(H[CP],write,d,val) $\in DU_{HD[CP]}$){d:=val;};

L 中删除元素 HD[CP];

将 Succ(W',HD[CP]) 插入到活动 L 列表中；}

Case AND_JOIN:{在L中遍历,若不存在HD[CP]; GOTO QUTE_FLAG; //同步控制

CP++;计算 DFR 中活动 HD[CP]列所有值为“1”的个数，表示为 $\sum_{j=0}^{k} DFR_{[j][HD[CP]]}$;

if(Count($N_{HD[CP]}$) < $\sum_{j=0}^{k} DFR_{[j][HD[CP]]}$){GOTO QUTE_FLAG; }

else{L 中删除元素 HD[CP]；将 Succ(W',HD[CP]) 插入到活动 L 列表中；

}}

Case XOR_SPLIT:{在 L 中遍历，若不存在 HD[CP];

GOTO QUTE_FLAG;

CP++;L 中删除元素 HD[CP];

遍历 N_x $\in$ Succ(W',HD[CP]) 根据其对应的 $DU_{HD[CP]}$ 检测是否满足条件 $\omega(N_x)$;

将唯一满足条件的 N_x 插入到活动 L 列表中;

}

Case XOR_AND:{在 L 中遍历，若不存在 HD[CP];

GOTO QUTE_FLAG;

CP++;L 中删除元素 HD[CP];将 Succ(W',HD[CP]) 插入活动 L 列表中;}

Case END:{符合迁移标准;

//END 通常迁移实例不会处于完成状态}

}

}

QUTE_FLAG:

If(CP<HD.length){For(int j=CP;CP<=HD.length;j++)

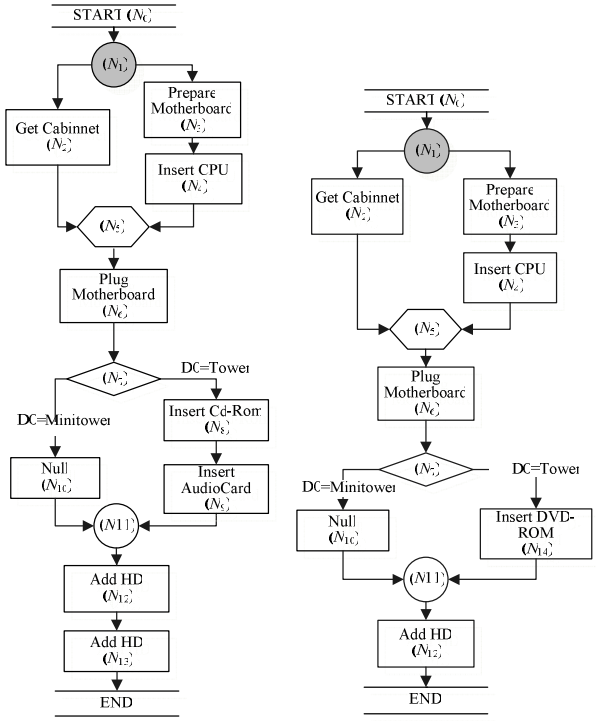
{

在历史执行记录中删除 HD[j]，根据其对应的 $DU_{HD[j]}$ 取消对相关数据单元的操作;}}

通过计算 W' 模型的 DFR 实现 AND_JOIN 类活动 N_i 的同步控制，当待选活动列表中 N_i 的个数 $Count(N_i) < \sum_{j=0}^{k} DFR_{[j][i]}$ 时，表明活动 N_i 无法激活；当 $Count(N_i) = \sum_{j=0}^{k} DFR_{[j][i]}$ 时，表明活动 N_i 满足前置条件可以激活；按照 W' 模型虚拟执行实例的历史记录，标记 *QUTE_FLAG* 处，若 *CP<HD.length*，表明当前实例的执行历史已经与新的工作流模型产生了不一致情况，需要回退该实例 $N_{HD[CP-1]}$ 活动处，以使实例与新的工作流模型保持一致。

6 应用实例

本文以文献[1]中简化的 PC 自动装配流程模型为例验证上述迁移算法的可行性，原工作流模型 W 和新工作流模型 W' 如图 2(a)和图 2(b)所示。其中，任务类型活动 N_6 和 N_{10} 转移的条件分别为数据单元 d_0 的值为 *Tower* 和 *Minitower*，即 $\omega(N_{10})$={ d_0 ="*Tower*"}，$\omega(N_{10})$={ d_0 ="*Minitower*"}，任务活动 N_6 对数据单元 d_0 进行(d_0 ,*para*,*write*)操作(*para* $\in$ {*Tower*, *Minitower*})。



(a)原工作流模型 W (b)新工作流模型 W'

图 2 PC 组装工作流模型

I 是基于原工作流模型 W 的实例，历史执行记录 $H=\{N_0,N_1,N_3,N_2,N_4,N_5,N_6,N_7,N_{10},N_{11},N_{12}\}$，各个活动对数据单元操作为 N_6 :(d_0,"Minitower",write)和 N_{10} :(d_0,"*Minitower*", *read*)，新工作流模型 W' 对应的 *DFR* 如表 1，各个活动对数据单元操作 *DU* 为 N_6 :(d_0,"*Minitower*",*write*)， N_7 :(d_0, "*Minitower*", *read*)，转移条件 ω (N_{10})={d_0="*Minitower*"}，ω (N_{14})={ d_0="Tower"}。

表 1 工作流模型 W' 对应的 DFR

F/T	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	E
s	1	0	0	0	0	0	0	0	0	0	0	0
N_1	0	1	1	0	0	0	0	0	0	0	0	0
N_2	0	0	0	0	1	0	0	0	0	0	0	0
N_3	0	0	0	1	0	0	0	0	0	0	0	0
N_4	0	0	0	0	1	0	0	0	0	0	0	0
N_5	0	0	0	0	0	1	0	0	0	0	0	0
N_6	0	0	0	0	0	0	1	0	0	0	0	0
N_7	0	0	0	0	0	0	0	1	1	0	0	0
N_8	0	0	0	0	0	0	0	0	0	1	0	0
N_9	0	0	0	0	0	0	0	0	0	1	0	0
N_{10}	0	0	0	0	0	0	0	0	0	0	1	0
N_{12}	0	0	0	0	0	0	0	0	0	0	0	1

对于 I 读取执行历史记录 H 初始化 HD，执行上述算法从模型 W' 的起始活动 *START* 开始虚拟执行，对后置转移进行评估并将后继活动 N_x 纳入虚拟列表 L 中，再从 HD 中取得活动作为虚拟点，在虚拟点执行前和执行过程中，对转移前置条件做相应的判断，以此类推，其虚拟过程如表 2 所示。

表 2 虚拟执行过程

序列	虚拟点	后继集合	满足条件	待选活动
1	*S*	N_1	N_1	N_1
2	N_1	N_2, N_3	N_2, N_3	N_2, N_3
3	N_3	N_4	N_4	N_2, N_4
4	N_2	N_5	N_5	N_5, N_4
5	N_4	N_5	N_5	N_5, N_5, N_5
6	N_5	N_6	N_6	N_6
7	N_6	N_7	N_7	N_7
8	N_7	N_{10}, N_{14}	N_{10}	N_{10}
9	N_{10}	N_{11}	N_{11}	N_{11}
10	N_{11}	N_{12}	N_{12}	N_{12}
11	N_{12}			

(下转第 33 页)