一种基于工作流类型版本和实例迁移的动态修改方法

崔立真　王海洋
（山东大学计算机科学与技术学院　济南250100）

摘　要　工作流管理系统中一项重要而未解决的问题是工作流模式的动态修改，比如，在保证模式正确的前提下，创建、删除和修改工作流类型。在修改工作流类型时，如果工作流实例是活动的，则这个问题就变得尤其严重，因为任何工作流实例必须符合其工作流类型的定义。本文中提出了一种基于工作流类型版本和工作流实例迁移的动态工作流模式修改方法。

关键词　工作流管理，动态工作流模式修改，工作流类型版本，工作流实例迁移

Dynamic Workflow Schema Evolution Based on Workflow Type Versioning and Workflow Migration

CUI Li-Zhen　WANG Hai-Yang
(Dept. of Computer Science and Technology, Shandong University, Jinan 250100)

Abstract　An important yet open problem in workflow management is the evolution of workflow schemas, i.e., the creation, deletion and modification of workflow type in such a way that the schema remains correct. This problem is aggravated when instance of modified workflow types are active at the time of modification, because any workflow instance has to conform to the definition of its type. This presents a framework for dynamic workflow schema evolution that is based on workflow type versioning and workflow migration.

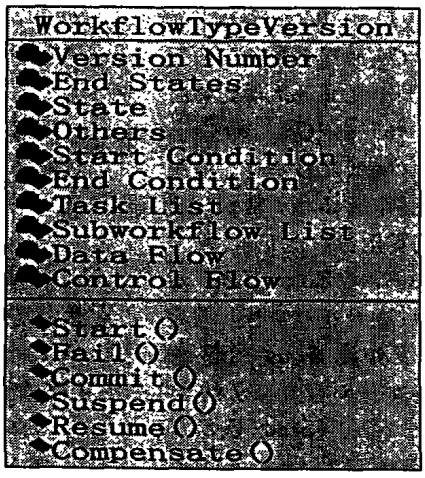
Keywords　Workflow management, Dynamic workflow schema evolution, Workflow type versioning, Workflow instance migration

1　引言

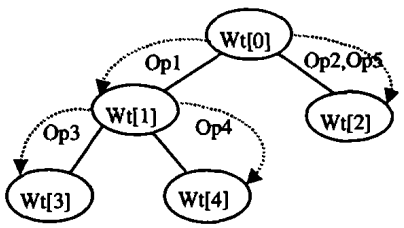
本文中提出了一种基于工作流类型版本和工作流实例迁移的动态工作流模式修改方法，它利用工作流类型版本化和工作流实例的迁移达到模式修改的目的。文[4]中的工作流迁移是直接对工作流实例进行修改操作，同文[4]相比，本文的面向对象的工作流概念模型更加简单，工作流的迁移条件都加以形式化描述，并提供了相应的算法。本文的另一创造性在于将工作流的恢复引入了工作流的模式修改。

2　支持动态修改的工作流概念模型

定义1（工作流类型版本[4]，WFT version）　工作流类型版本或者是一复合工作流类型版本或者是活动类型版本，有全局唯一的命名。在本文中我们采用如下面向对象的概念模型表示。



定义2（工作流类型版本树[4]，WFTVT）　工作流类型版本树定义了工作流类型之间的相互关系。如左图所示工作类型树，其节点为工作流类型，虚线表示应用的操作，实线表示节点之间的关系，wt [1]通过对 wt [0]进行操作 Op_1而得到，wt [2]通过对 wt [0]进行操作 Op_2、Op_5而得到，wt [3]通过对 wt [1]进行操作 Op_3而得到，wt [4]通过对 wt [1]进行操作 Op_4而得到。

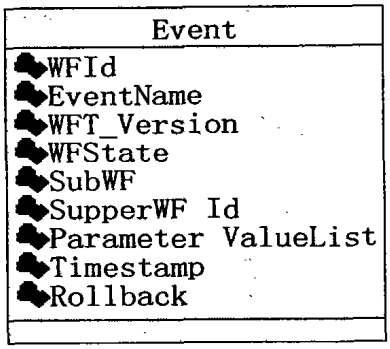


定义3（工作流类型[4]，WFT）　工作流类型或任务类型，由全局唯一的名字和对应的工作流类型版本树组成。

3　工作流模式的动态修改

3.1　工作流执行日志

工作流执行日志记录了工作流所有执行状态的变化，工作流执行状态的变化可以看作是一个个事件，因此，工作流执行日志就是事件日志。工作流的正确执行依赖于工作流执行日志必须满足工作流执行约束。当工作流改变其执行状态，实际上是事件的产生而导致的。当在某一时刻工作流由起始状态进入起执行状态，实际上是由 $Start_w$ 事件引起的，而结束状态则是由 End_w 事件引起的。我们形式地描述事件的结构如下。



工作流的执行日志 L 可以看作是一个偏序集 $L=(E,<)$，E 为工作流 W 引发的事件的集合，偏序关系$<$是 E 上基于时间戳的反自反的、传递的二元关系。即$\forall e1、e2\in E$，$e1.timestamp<e2.timestamp\Leftrightarrow(e1<e2)$。

为了描述问题的方便，我们定义以下运算符：

·E(关系运算)：表示 L 中满足该关系运算的事件集合。

·$L'(E',<')$：表示 L 的子集，其中 $E'\subseteq E\wedge\forall e1,e2\in E',e1<e2\Rightarrow e1<'e2$。

·$E[e1]=\{\forall e\in E|\neg(e<e1)\}$：表示不在 $e1$ 之后发生的所有事件的集合。

·$(E_{ct},<_{ct})$：表示到时刻 ct 为止所记录的日志，其中 E_{ct} $\{\forall e\in E|timestamp(e)\leqslant ct\}$。

3.2 工作流执行约束

下面是形式化的工作流执行过程中的事件约束，公式成立的前提为公式中事件为有效事件，即事件的回滚标志(Rollback)为假。

3.2.1 *工作流 w 的 $Start_w$ 事件约束*

·$Start_w\in E_{ct}\Rightarrow\neg(Start'_w<_{ct}Start_w)$

·$Start_w.Supperwf\_id<>Null\wedge Start_w\in E_{ct}\Rightarrow w.start\_condtion=True\wedge Supperwf\_id.state=Executing$

·$Start_w.Supperwf\_id<>Null\wedge Start_w\in E_{ct}\Rightarrow Start._{superwf}<_{ct}Start_w\wedge End_{superwf}\notin E[Start_w]$

3.2.2 *End_w 事件约束*

·$End_w\in E_{ct}\Rightarrow\neg(End'_w<_{ct}End_w)\wedge Start_w<_{ct}End_w\wedge E.end\_condition=true$

·$End_w.Supperwf\_id<>Null\wedge End_w\in E_{ct}\Rightarrow\neg(End_{super\_wf}<_{ct}End_w)$

3.2.3 *其它约束* 工作流及其子工作流必须严格按照其定义执行。

3.3 模式修改操作原语

1)创建新的工作流类型版本。Create workflow type version 包括如下操作：

编号	操作	说明
Op_1	add a new parameter	增加新参数
Op_2	delete a parameter	删除参数
Op_3	add a new end state	增加新的结束状态
Op_4	delete a end state	删除一种结束状态
Op_5	add a new subworkflow declaration	增加一个新子工作流定义
Op_6	delete a new subworkflow declaration	删除一个子工作流定义
Op_7	modify the start condion of a subworkflow	修改子工作流的起始条件
Op_8	add a new data flow definition	增加一个新的数据流
Op_9	delete a data flow definition	删除一个数据流
Op_{10}	modify an end condition	修改工作流的结束条件

2)删除工作流类型版本 Drop workflow type versio；

3)创建一个新的工作流类型 Create workflow type；

4)删除一个工作流类型 Drop workflow type。

工作流的模式修改就是利用上面工作流模式修改操作原语进行处理的过程。一个新的工作流类型版本的产生是首先继承现有某一工作流类型版本，然后通过操作 $Op_1\sim Op_{10}$进行修改而得到的；在删除工作流类型版本时只能删除处于版本树底端(即叶子节点)的工作流类型版本；创建一个新的工作类型的同时也创建了该工作流类型的第一个版本；删除一个工作流类型将删除所有该工作流类型的信息包括该工作流类型的所有实例。

4 工作流迁移

当创建了一个新的工作流类型版本后根据需要将满足条件的工作流实例迁移到新的版本，这个过程称为工作流的迁移。工作流实例是否能够迁移取决于工作流实例当前所处的状态，由于任何一个非活跃状态的工作流实例都可以满足迁移条件，我们只考虑处于活动状态的工作流实例。一般地，只要在到时刻 ct 为止，如果工作流实例 w，满足新工作流类型版本的需求，那么工作流就满足迁移条件。一个简单的方法是检查工作流的所有执行日志是否满足迁移后的工作流类型版本，但这种方法在大量复合工作流需要迁移时，效率很低；我们考虑一种更为有效的方法，通过分析工作流模式的修改操作来有效地判断工作流是否能够迁移，并对不满足迁移条件的工作流实例首先进行恢复，然后再进行迁移。

4.1 迁移条件

假定$\forall wt[x]、wt[y]$是工作流类型 wt 的版本，$MC(W,OP,wt[s],wt[d],ct)$表示工作流实例 w 从 $wt[x]$迁移到 $wt[y]$的满足条件判别式，OP 是模式修改操作的有序集合，表示 $wt[y]$是按照操作再 OP 的循序的依次对 $wt[x]$修改而得到的工作流类型版本，ct 是工作流 w 迁移的时刻。我们对工作流模式修改操作的迁移条件形式地描述如下：

1)操作 Op_1的迁移条件 假定 $wt[y]$是由 $wt[x]$增加参数 p 得到，则

$\forall ct\Rightarrow MC(w,\{Op_1\},wt[x],wt[y],ct)\equiv$True

2)操作 Op_2的迁移条件 假定 $wt[y]$是由 $wt[x]$删除参数 p 得到，则

$\forall ct\Rightarrow MC(w,\{Op_2\},wt[x],wt[y],ct)\equiv$True

3)操作 Op_3的迁移条件 假定 $wt[y]$是由 $wt[x]$增加结束状态 End_z 得到，则

$\forall ct\Rightarrow MC(w,\{Op_3\},wt[x],wt[y],ct)\equiv$True

4)操作 Op_4的迁移条件 假定 $wt[y]$是由 $wt[x]$删除结束状态 End_z 得到，则

$End_{zw}\notin E_{ct}\Rightarrow MC(w,\{Op_4\},wt[x],wt[y],ct)=$True

5)操作 Op_5的迁移条件 假定 $wt[y]$是由复合工作流 $wt[x]$增加子工作流 sb_wf 得到，则

$\forall ct\Rightarrow MC(w,\{Op_5\},wt[x],wt[y],ct)\equiv$True

6)操作 Op_6的迁移条件 假定 $wt[y]$是由复合工作流 $wt[x]$删除子工作流 sb_wf 得到，s 是在工作流 w 中对应 sb_wf 的子工作流实例，则

$Start_s\notin E_{ct}\Rightarrow MC(w,\{Op_6\},wt[x],wt[y],ct)=$True

7)操作 Op_7的迁移条件 假定 $wt[y]$是由复合工作流 $wt[x]$修改子工作流 sb_wf 的起始条件 $start\_conditon$ 得到，s 是在工作流 w 中对应 sb_wf 的子工作流实例，则

$Start_s\notin E_{ct}\vee(Start_s\in E_{ct}\wedge wt[y].start_conditon=True)\Rightarrow MC(w,\{Op_7\},wt[x],wt[y],ct)=$True

8)操作 Op_8 的迁移条件　假定 $wt[y]$ 是由复合工作流 $wt[x]$ 增加数据流 df 得到：

如果 df 指向 w 的某一子工作流 sb_wf 的实例 s 输入参数，则

$Start_s \in E_{ct} \Rightarrow MC(w,\{Op_8\},wt[x],wt[y],ct)=\text{True}$

如果 df 指向 w 的某一子工作流 sb_wf 的实例 s 输出参数，则

$End_s \in E_{ct} \Rightarrow MC(w,\{Op_8\},wt[x],wt[y],ct)=\text{True}$

9)操作 Op_9 的迁移条件　假定 $wt[y]$ 是由复合工作流 $wt[x]$ 删除数据流 df 得到：

如果 df 指向 w 的某一子工作流 sb_wf 的实例 s 输入参数，则

$Start_s \in E_{ct} \Rightarrow MC(w,\{Op_9\},wt[x],wt[y],ct)=\text{True}$

如果 df 指向 w 的某一子工作流 sb_wf 的实例 s 输出参数，则

$End_s \in E_{ct} \Rightarrow MC(w,\{Op_9\},wt[x],wt[y],ct)=\text{True}$

10)操作 Op_{10} 的迁移条件　假定 $wt[y]$ 是由复合工作流 $wt[x]$ 修改子工作流 sb_wf 的结束条件 $end\_conditon$ 得到，s 是在工作流实例 w 中对应 sb_wf 的子工作流实例，则

$End_s \in E_{ct} \vee (End_s \in E_{ct} \wedge wt[y].end\_conditon=True) \Rightarrow MC(w,\{Op_{10}\},wt[x],wt[y],ct)=\text{True}$

4.2　迁移条件的检查及恢复

在时刻 ct，操作集 OP 决定了工作流实例 w 是否能够从 $w[x]$ 迁移到 $w[y]$。因此，工作流实例 w 只要满足所有操作的迁移条件，就能够从 $w[x]$ 迁移到 $w[y]$。假定工作流类型版本从 $w[x]$ 迁移到 $w[y]$ 的操作集 $OP=\{op_1,op_2,\cdots,op_n\}$，则工作流实例 w 的迁移条件：

$MC(w,\{op_1,op_2,\cdots,op_n\},wt[x],wt[y],ct)=MC(w,\{op_1\},wt[x],wt[y],ct) \wedge \cdots \wedge MC(w,\{op_n\},wt[x],wt[y],ct)$

当上式的值为 False，即工作流的不满足迁移条件时，需要对工作流进行恢复，使其满足迁移条件，恢复算法如下：

1. 根据操作集 OP，我们按照上式进行分解工作流的迁移条件，依次确定不满足工作流迁移条件的有序操作集 OP'。

2. 根据操作集 OP'，我们确定工作流实例 w 需要恢复到的状态 $state_{rec}$。

3. 定义 WF 为工作流实例的集合，$state_{now}$ 为工作流实例 w 的当前状态，我们采用以下算法确定恢复工作流实例 w 所引起级联恢复的工作流的实例集 WF：

假定工作流处于起始状态 $state_{now}$=Initial，工作流处于非活跃状态，迁移条件始终满足，在此我们不予考虑。

假定工作流处于执行状态 $state_{now}$=Executing 和工作流处于结束状态 $state_{now}=End_w$，WF 初始值为 $\{w\}$。

如果工作流实例 w 是复合工作流，则对 w 的子工作流 subworkflow 的实例 s，如果 $start_s \in E_{ct}$，则 $WF=WF \cup \{s\}$。

我们可以根据工作流实例 w 的上级工作流 *super_workflow* 的数据流(data flow)定义来得以确定，如果工作流实例 w' 引用工作流实例 w 的输出参数值，并且 $start'_w \in E_{ct}$ 则 $WF=WF \cup \{w'\}$。

我们可以根据工作流实例 w' 输入参数判定，如果工作流实例 w' 引用工作流 w 的结束状态，并且 $start'_w \in E_{ct}$，则 $WF=WF \cup \{w'\}$。

4. $\forall w \in WF$，我们按照上面的算法计算 WF，直至 WF 不再发生变化。由于工作流的个数有限，因此必定在某一步终止。

5. 按照工作流起始事件发生的时间戳 timestamp，对需要恢复的工作流的实例集 WF 进行降序排列。

6. 通过工作流的恢复将 w 恢复状态 $state_{rec}$，同时将 WF 中的其它工作流实例恢复到状态 Initial。

7. 工作流恢复完毕，将工作流恢复的日志的回滚标志(Rollback)置为 True，工作流实例 w 满足迁移条件。

4.3　迁移步骤

现在我们采用下面的算法对工作流的进行迁移：

Step1　在时刻 ct 触发该工作流实例 w 的 *suspend* 事件，保存当前工作流的信息，包括起执行状态、参数值等。

Step2　按照4.2节的算法检查工作流实例 w 进行迁移的条件。

Step3　如果工作流实例 w 不满足迁移条件，按照4.2节的算法将工组流实例 w 恢复为满足迁移条件。

Step4　如果工作流实例 w 满足迁移条件，进行下面的操作：

保留指向实例 w 的指针，和工作流实例 w 所处的状态，然后释放 w 的内存；创建 $wt[y]$ 的一个实例 w'，取出 w 的信息赋给 w'，w' 指向 w 的指针。这样我们将工作流实例 w' 完全替代了工作流实例 w。

Step5　触发工作流实例 w' 的 resume 事件。

Step6　工作流实例 w 完成迁移。

结论　在本文中提出了一种基于工作流类型版本化和工作流迁移的动态工作流模式修改的方法。工作流类型被版本化，新的版本是对现有的工作流类型进行修改操作而得到的。我们提出了不同于文[4]的工作流迁移方法，我们的方法更为切实可行。在文[4]中的工作流迁移是直接对工作流实例进行修改操作，本文中提出了一种创建新工作流类型版本实例，并对其状态信息设置为原有工作流在满足迁移条件时刻所处的状态，以新的工作流实例代替原有的工作流实例的方法，这是一种更加行之有效的方法。同时在我们的工作流迁移中支持工作流的恢复，这样工作流的模式修改的适应范围就大大拓广了。另外，为了提高模式修改的效率，我们可以有选择地迁移工作流，只迁移符合条件的工作流实例，以避免不必要的工作流实例的迁移。因此，我们的工作流模式修改效率更高，适用的范围更广。

参考文献

1　何炎祥，郑振楣，石树刚 编著. 面向对象的数据库. 武汉大学出版社，1995

2　Kradofer M, Geppert A. Modeling Concepts for Workflow Specification: [Technical Report 97.05]. Department of Computer Science, University of Zurich, 1997

3　Monk S, Sommerville I. Schema Evolution in OODBs using Class versioning. SICMOD Record, 1993, 22(3)

4　Kradofer M, Geppert A. Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration: [Technical Report 98.02]. Department of Computer Science, University of Zurich, 1998

5　Reichert M, Rinderle S, Dadam P. A Formal Framework for Workflow Type and Instance Changes Under Correctness Constraints: [Technical Report UIB-2003-01]. University of Ulm, Faculty of Computer Science, April 2003

6　Sadiq S, Marjanovic O, Orlowska M. Managing change and time in dynamic workflow processes. The Int'l Journal of Coop. Inf. Syst., 2000, 9(1&2)

7　Rinderle S, Reichert M, Dadam P. Evaluation of Correctness Criteria For Dynamic Workflow Changes. In: Proc. Int'l Conf. on Business Process Management (BPM '03), Eindhoven, The Netherlands, LNCS 2678, 2003. 41～57

8　Reichert M, Rinderle S, Dadam P. ADEPT Workflow Management System: Flexible Support For Enterprise-wide Business Processes (Tool Presentation). In: Proc. Int'l Conf. on Business Process Management (BPM '03), Eindhoven, The Netherlands, LNCS 2678, 2003. 379～79