基于对象模型工作流的失败处理与失败恢复*

高 军[1,2], 王海洋[2]

[1](北京大学 计算机科学技术系,北京 100871);

[2](山东大学 计算机科学系,山东 济南 250100)

E-mail: gaojun@db.pku.edu.cn

http://www.cs.pku.edu.cn

摘要: 工作流的失败处理和失败恢复是工作流管理系统的重要组成部分.提出了一种新的基于对象的工作流模型,在此模型下设计了工作流失败处理和失败恢复的策略.与传统方法相比,新策略考虑了工作步骤之间的控制依赖和数据依赖,并且对工作步骤应用的具体实现方式进行了探讨,以提高工作流失败处理和失败恢复的效率.

关键词: 工作流;失败处理;强数据依赖;强控制依赖;自恢复工作流

中图法分类号: TP391 **文献标识码**: A

工作流管理系统根据其应用领域的不同,可粗略分为特定工作流管理系统、决策工作流管理系统和商用工作流管理系统.其中,在对数据的一致性和正确性要求较高的商用工作流系统中,存在着工作流的失败处理和失败恢复问题.

工作流的失败处理和失败恢复问题的特殊性,是由于工作流管理系统为适应工作流长事务的要求,而放松了数据库中事务的ACID属性要求.对于工作流的失败处理和失败恢复,Jian Tang提出了CU(consistency unit)的概念来解决工作步骤执行失败后如何保持数据一致性的处理方法[1].但是,这种方法未考虑工作步骤之间存在的数据依赖.在FLEXIBLE TRANSACTION中引入子事务的概念[2],如果事务T的某个子事务执行失败,则整个事务T执行失败,这种方法同样未考虑数据依赖,而且不适合工作流长事务的特性.LINEAR SAGAS系统的底层模型的表达能力不够[3],并且工作流的失败恢复执行效率较低.

在本文中,我们提出了一种面向对象的工作流概念模型,该模型是Gregor提出的工作流对象模型的扩展[3].新的模型增强了原模型的表达能力和失败处理能力.在该模型的基础上,提出了两个新的概念——强数据依赖和强控制依赖,以满足工作流之间的数据一致性的要求.另外,我们对工作步骤的具体实现方式也进行了探讨,提出自恢复工作流以提高在工作流失败恢复时的执行效率.

1 支持失败处理和失败恢复的工作流概念模型

1.1 不同性质的工作流要求不同的失败处理和失败恢复策略

在工作流执行过程中,如果整个工作流执行失败,则按照工作步骤应用的具体语义,分别处理:

(1) 无须恢复,继续运行.

* 收稿日期: 1999-09-21; 修改日期: 2000-02-25

基金项目: 山东省自然科学基金资助项目(Q97G01158)

作者简介: 高军(1975-),男,山东临沂人,博士生,主要研究领域为工作流,信息集成;王海洋(1965-),男,山东文登人,博士,教授,博士生导师,主要研究领域为演绎数据库,计算机协同工作环境.

适合第(1)种恢复策略的情况包括两种:① 从应用程序的角度看不会影响最终结果的工作步骤,如订购车票、取消订购,如果无须付定金,则不必设法恢复该工作步骤所造成的影响;② 从语义上讲无意义的或无必要恢复的,如在电子贸易工作流中用户登录时检查身份的工作步骤.

(2) 反复执行.

适合第(2)种恢复策略的情况:如果根据具体的执行环境,我们知道某工作步骤能够执行成功,如申请银行帐号.考虑到系统所出现的问题,可设最大执行次数,如果超过最大执行次数,则按系统失败处理.

(3) 向后恢复,执行其恢复工作步骤.

第(3)种恢复策略适合工作流中的关键步骤,如果此类工作步骤执行失败,有必要将其执行结果全部恢复.一般我们对其恢复工作步骤仔细设计,在工作流管理层假定其恢复工作步骤一定能够执行成功.当然,在工作流管理层更可靠的策略是,如果某关键工作步骤的恢复工作步骤执行失败,则重复执行恢复工作步骤,直至执行成功.

(4) 向前恢复,执行其替代工作步骤.

第(4)种恢复策略主要适合工作步骤为非关键步骤的情况.例如,工作步骤处于多路选择结构中的一条选择路径上,当该工作步骤执行失败后,可以选择在其他分支上的替代工作步骤执行.替代工作步骤是对工作流长事务的一种支持.

(5) 需要用户干预.

根据工作步骤执行是用户参与的程度,工作步骤可分为面向应用的工作步骤和面向用户交互的工作步骤.在面向用户交互的工作流中,如果因某一工作步骤执行出现错误而导致整个工作流运行失败,则系统软件所做的只能是提醒用户重新完成.例如,在办公工作流中,办公人员如果在发信时将邮件的地址写错,则只能由工作人员重新发送.

上述工作步骤的失败处理的性质可由工作流设计者在工作流定义阶段来定义,工作流的调度部件将根据上述信息执行不同的恢复操作.

1.2 支持失败处理和失败恢复的工作流概念模型

在Gregor的工作流对象定义中[13],仍对工作步骤和工作流对象分别定义.在本文中,工作步骤的概念和工作流的概念不再严格区分.

定义1. 设工作步骤集合为T,如果工作步骤$t_1\in T$和工作步骤$t_2\in T$之间存在控制依赖,则记为$C(t_1,t_2)$;如果工作步骤$t_1\in T$和工作步骤$t_2\in T$之间存在数据依赖,则记为$D(t_1,t_2)$,工作步骤t定义在工作流W中,则记为$\mathrm{IN}(t,W)$;工作流可递归定义如下:

(1) 单一工作步骤$t\in T$构成工作流;

(2) 如果存在工作流A,工作步骤t,$\exists a\{(D(a,t)\text{ or }C(a,t))\text{ and }\mathrm{in}(a,A)\mid a\in T,t\in T\}$,工作流$A$和工作步骤$t$的工作步骤通过控制依赖或数据依赖得到一个新的结构$C$,则$C$为工作流.

通过上述定义,不难得出,从面向对象的角度讲,工作流对象和工作步骤对象是一致的.采用工作流对象和工作步骤对象一致的定义,便于工作流定义的逐步细化,更加符合人类的思维方式.

本文中工作流概念模型定义

```
class workflowdefination
  {attributes    //描述工作流的各种属性(时限属性、权限属性、操作人员属性、资源属性)
                   在本文中讨论上述工作流的恢复属性
   paramcter     //本工作流的输入参数和输出参数
```

```
status          //工作流对象的状态
task            //描述工作任务的实现部件
failtask        //描述工作任务执行失败后的执行策略
recoverytask    //描述工作流被触发"恢复事件"后的操作
sub_workflow_list   //子工作流序列
sub_workflow_para_maptable   //子工作流的参数对应关系
sub_workflow_rule      //描述工作流之间的控制依赖关系
}
```

在 Gregor 的工作流对象定义中[3],工作步骤之间的控制依赖的关系以"and/or split"和"and/or join"来定义.这种关系描述方法复杂,而且不能表达工作流的某些特定流程,如条件选择、循环等.在本文中,子工作流的控制依赖是通过消息传递和消息响应规则集的方式来完成的.

在本文中,工作流消息规则的形式为(ON A.EVENT,CONDITION,CALL TASK | B.SENDEVENT),表明工作流 A 在事件 EVENT 被触发并且 CONDITION 成立的情况下,或者触发另一工作流的事件,或者调用一项工作任务.

其中,工作流对象接受的消息及可能的典型处理方式如下:

START(启动工作流):调用本工作流的任务;

FAIL(工作流运行失败):触发其他工作流的恢复事件或调用本工作流的失败任务;

COMMIT(工作流运行成功):触发其他工作流的启动事件;

SUSPEND(工作流挂起):挂起正在运行的工作流;

RESUME(工作流继续执行):工作流在挂起处继续执行;

COMPENSATE(工作流运行恢复程序):调用本工作流的恢复任务.

其中,CONDITION 是工作流对象状态或工作流参数的布尔表达式的逻辑表达式,该逻辑表达式支持对条件的"或"以及"与"操作.因此,可以方便地涵盖 Gregor 的工作流对象定义中工作步骤之间存在的"and/or split"和"and/or join"的控制依赖关系.

工作流之间的数据依赖关系可用工作流参数对应来表示,其形式为 $A.\text{in}=F(B.\text{out})$,其中,$A$ 和 B 为工作流对象,$A.\text{in}$ 表示工作流对象 A 的输入参数,$B.\text{out}$ 表示工作流对象 B 的输出参数,函数 F 是描述工作流定义 A 和 B 的输入/输出参数的函数关系.

在 FLEXIBLE TRANSACTION 模型中[2],利用工作步骤的恢复工作步骤来完成对工作流的失败恢复.在本文中,我们在工作流对象模型定义中加入恢复任务属性,该属性描述了在工作流执行失败后的操作.这样做,一方面减少了系统中存在的工作流的数量,简化了工作流的设计;另一方面,也符合工作流对象的封装性质,即工作流对象本身能够进行失败处理和失败恢复.

Table 1 The strategy of failure handling and failure recovery according to different recovery-attribute mentioned in section 1.1 in workflow (suppose worktask is T)

表 1 对第 1.1 节中不同恢复属性的工作流的失败处理和失败恢复策略(设工作任务为 T)

Recovery-attr①	Recovery task②	Failure task③
Recovery-attr=1	Null④	Null
Recovery-attr=2	Null	T (worktask)⑤
Recovery-attr=3	T_A(maybe the same as T)⑥	Null
Recovery-attr=4	Call event⑦	Null
Recovery-attr=5	Null	Hint⑧

①恢复属性,②恢复任务,③失败任务,④无,⑤T(工作任务),⑥T_A(可能和 T 一致),⑦触发事件,⑧提示处理.

2 用户数据一致性定义和工作流的响应规则之间的转化

工作流设计本身表达了工作步骤之间的相互联系、相互作用的关系.对用户来讲,工作流内部定义中的消息响应规则应当是透明的.工作流管理系统应当使用户着重于对工作流之间较高层的逻辑关系的设计.除了工作流的一般控制结构(顺序、选择、循环、并行)和数据参数的对应关系以外,还应当支持对失败处理和失败恢复的扩展定义.

2.1 子工作流间的强数据依赖

子工作流间存在的相互间的数据依赖,表达了子工作流参数之间的输入/输出关系.子工作流之间的参数类型可能为整数、字符等基本类型,也可能为文件,又可能为数据库中的某项记录.

定义 2. 在执行子工作流的向后恢复策略中,设子工作流 A 的输出参数集合为 $S_{A[OUT]}$,子工作流 B 的输入参数为 $S_{B[IN]}$,如果子工作流 B 的输入参数集合和子工作流 A 的输出参数集合在具体的环境中有关联,即 $S_{B[IN]}$ 和 $S_{A[OUT]}$ 之间存在函数关系,并且在语义上,如果子工作流 A 执行失败,若执行子工作流 A 的恢复操作后,子工作流 B 的输入参数集合无意义,则称子工作流 B 强数据依赖子工作流 A.其中,子工作流 B 称为强依赖子工作流,子工作流 A 称为被强数据依赖子工作流.记作 $A\rightarrow B$.

在强数据依赖中,注意以下两点:

首先,强数据依赖 $A\rightarrow B$ 表明,如果子工作流 A 执行其恢复工作任务,则子工作流 B 也必须执行其恢复工作任务.

其次,并非所有的数据依赖都是强数据依赖,但强数据依赖一定是数据依赖.

子工作流的强数据依赖是非自反的,如果存在 $A\rightarrow A$,则说明子工作流 A 的参数集中某些输入参数导出某些输出参数,这在语义上无意义.

子工作流的强数据依赖是传递的.如果 $A\rightarrow B$,并且 $B\rightarrow C$,则 $A\rightarrow C$.也就是说,如果子工作流 A 执行其恢复工作任务,则属于子工作流 A 强数据依赖的传递闭包的所有工作流都要执行相应的恢复工作任务.

用户定义强数据依赖后,工作流管理系统将该强数据依赖转换为相应工作流对象的消息响应规则.如强数据依赖 $A\rightarrow B$,系统在工作流对象 A 中增加的消息响应规则为(on A.fail, true, B.sendevent(compensate))和(on A.fail,true,B.sendevent(fail)).

例 1:如果工作流对象 A,B,C 之间存在强数据依赖 $A\rightarrow B$ 和 $B\rightarrow C$,则工作流 A 执行恢复工作任务后,工作流对象之间的消息响应顺序如图 1 所示.

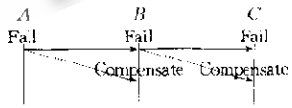


Fig. 1 The order of event invoked between the workflow objects in example 1
图1 例1中工作流对象间事件的调用顺序

2.2 子工作流间的强控制依赖集

在工作流中,为保证数据间的一致性,定义了工作流的强控制依赖集.工作流的强控制依赖集在语义上类似于数据库系统中事务的概念,描述了工作流之间存在的或者全部执行成功,或者全部不执行的语义要求.

定义 3. $W=(S,R)$是一个工作流，其中 $S=\{S_1,S_2,\ldots,S_N\}$是子工作流的集合，$R=\{R_1,R_2,\ldots,R_M\}$是消息响应规则集合.在该工作流中定义强控制依赖集 $X=(Y,Z)$，其中 Y 是 S 的子集，Z 是 R 在子工作流 Y 上的投影.如果 $S_i\in Y$，S_i 执行失败，则触发 Y 集合的其他子工作流的恢复工作事件.集合 X 称之为强控制依赖集.

强控制依赖集的定义，允许用户描述工作流中子工作流集合或者全部完成，或者全部不执行的关系，保证了工作流在执行过程中所操作的外部数据的一致性.例如，我们在定义购物的工作流时，可以将付款和提货置于一个强控制依赖集中，其中任何一个子工作流运行失败，其他子工作流就执行恢复任务.

用户定义强控制依赖集 $Y(Y_1,Y_2,\ldots,Y_N)$后，工作流管理系统将该强控制依赖转换为相应工作流对象的消息响应规则.即在工作流内部定义中，系统追加消息响应规则到 Y_i 的工作流对象的定义中，消息响应规则共$(n-1)*n$条.根据消息响应顺序，其消息响应规则形式为$\{(\text{on } Y_i.\text{fail},\text{true},Y_k.\text{sendevent(compensate)})\mid 1<i\leqslant n,1<k\leqslant n,I()k\}$.

强控制依赖和强数据依赖都是保持工作流操作外部数据一致性的一种手段，它们之间存在着以下差别：

(1) 强数据恢复依赖表明，如果子工作流 A 和子工作流 B 存在强数据依赖的关系，则子工作流 B 的恢复任务的执行导致子工作流 A 的恢复任务的执行.但如果子工作流 A 失败，对子工作流 B 无影响；但定义在强控制依赖中的子工作流之一失败，将导致整个处于强控制依赖集的其他子工作流恢复任务执行.

(2) 如果子工作流之间存在数据恢复依赖，则其输入/输出参数之间一定存在某种对应关系，所以，在强数据依赖关系中，实际上严格存在着子工作流执行的先后顺序.但是，在强控制依赖集中，子工作流之间不存在相应的数据之间的关系，子工作流间不存在严格的执行顺序.

例 2：在强控制依赖集$\{A,B,C\}$中，子工作流 A 运行失败，触发子工作流 A 和 B 的恢复事件如图 2 所示.

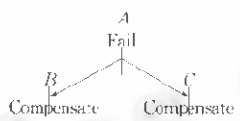


Fig. 2 The flow of event between the workflow objects in example 2
图2 例2中工作流对象消息流程示意

2.3 基于对象模型工作流的定义

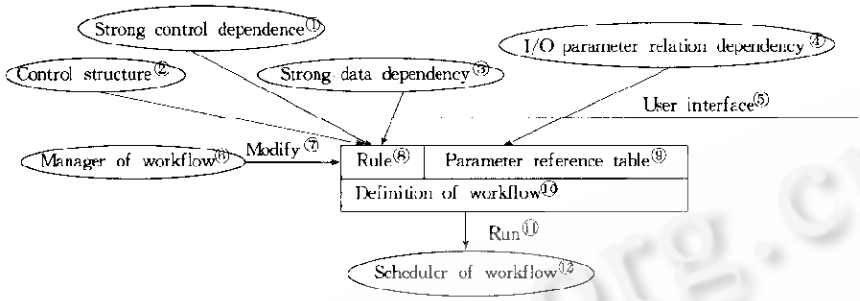
在本文提出的工作流的概念模型中，用户定义工作流的扩展性质以及上述性质由用户界面到工作流对象的消息响应规则的转化如图 3 所示.

用户利用系统提供的界面完成工作流的定义.工作流的控制结构连同工作流的强数据依赖和强控制依赖，经过系统转化，成为基于对象模型工作流定义的消息响应规则；而工作流之间的输入/输出参数关系转化到工作流定义的参数对照表中.工作流的管理人员可以修改工作流的定义.工作流的调度部件根据工作流的定义运行工作流.

3 工作流的恢复效率提高

定义 4. 如果子工作流 A 工作任务的输入参数集合是 I，函数 F 表达工作流执行参数后对外部数据的状态，设原外部数据的初始状态为 S，则在 S 状态下执行输入参数集 I_1，得到新的状态

$F_S(I_1)$;在 S 状态下执行输入参数集 I_2,得到新的状态 $F_S(I_2)$;如果需要外部数据状态由 $F_S(I_1)$恢复至$F_S(I_2)$,则称这种恢复为部分恢复.



①强控制依赖,②控制结构,③强数据依赖,④输入/输出参数依赖关系,⑤用户界面,⑥工作流管理人员,⑦修改,⑧工作流规则,⑨参数对照表,⑩工作流定义,⑪运行,⑫工作流调度部件.

Fig. 3 The conversion from the user defined extended properties to inner properties of workflow definition
图3 用户扩展性质到工作流内部定义的转换

定义 5. 如果对于工作流 A,$F_S(I_1)*F_S(I_2)=F_S(I_1+I_2)$,其中 F,S 的含义同上,I_1 和 I_2 是两种可能的输入参数集,并且在 I_1,I_2 输入期间,没有对外部数据库的其他操作,则称该工作流 A 为自恢复工作流.其中 $F_S(I_1)*F_S(I_2)$表示外部数据在 S 状态下,该子工作流先执行参数 I_1,再执行参数 I_2 后外部数据的状态;$F_S(I_1+I_2)$表示外部数据在 S 状态下,该子工作流执行参数 I_1-I_2 后对外部数据的状态;"+"表示在参数集合中可执行的运算操作;"="表示外部数据状态在逻辑上是一致的.

不妨设子工作流 A 是自恢复子工作流,如果参数集 $I_2=-I_1$,则易得到 $F_S(I_1)*F_S(-I_1)=F_S(I_1-I_1)=F_S(0)$.其中,"-"为"+"的逆运算;0 为一个特殊输入参数集;$F(0)$的状态表明,以 0 为输入参数集,执行该工作流不改变外部数据的状态.

上述操作表明,如果子工作流是可自恢复的,则不必设计其恢复工作任务,通过修改其输入参数,执行原工作任务的方法即可使外部数据状态在逻辑上达到一致.

如果工作流为可自恢复工作流,则对工作流进行部分恢复时效率高.如用户的订购要求由 30 项变成 20 项,则可以通过修改参数的方法来完成.在形式上,如果子工作流的原参数集为 I_1,新参数集为 I_2,则参数间差集为 $I=I_2-I_1$,执行输入参数集 I,即 $F_S(I_1)*F_S(I_2-I_1)=F_S(I_1-I_2-I_1)=F_S(I_2)$,达到部分恢复的效果.如果该工作流不支持自恢复,则部分恢复工作只能为先执行工作流的恢复工作任务,然后再重新按新参数集执行工作任务.可见,自恢复子工作流在处理部分恢复时,效率比一般的非自恢复子工作流要高,并且不必单独设计恢复工作任务.

如果工作流 A 为自恢复工作流,则在工作流对象定义中,recoverytask=task.

4 结 论

本文提出了一种工作流的对象建模方法,并在该模型基础上提出了工作流失败处理和失败恢复策略.这种处理方法同时考虑了子工作流之间的控制依赖和数据依赖,工作流之间控制灵活,工作流调度部件设计简单.同时,本文提出了可自恢复工作流方法,减少了程序设计量,提高了工作流执行恢复操作的效率.

References:

[1] Tang, Jian, Veijalainen, J. Transaction-Oriented work-flow concepts in inter-organizational environments. In: Pissinou, N., Silberschatz, A., Park, E. K., *et al.*, eds. Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95). New York: ACM Press, 1995. 279~288.

[2] Alonso, G., Agrawal, D. Advanced transaction models in workflow contexts. In: Stanley, Y. W. Su., ed. Proceedings of the 12th International Conference on Data Engineering. New York: IEEE Computer Society, 1996. 574~581.

[3] Joeis, G., Herzog, O. Managing evoling workflow specfication. In: Geller, J., Lochovsky F. H., eds. Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems. New York: IEEE-CS Press, 1998. 310~321.

Failure Handling and Recovery in Workflow Based on Object Model*

GAO Jun[1,2], WANG Hai-yang[2]

[1] (*Department of Computer Science and Technology, Beijing University, Beijing* 100871, *China*);

[2] (*Department of Computer Science, Shandong University, Jinan* 250100, *China*)

E-mail: gaojun@db.pku.edu.cn

http://www.cs.pku.edu.cn

Abstract: The failure handling and the failure recovery are the key parts of workflow management. In this paper, a workflow model based on object is introduced. And the strategy of failure handling and failure recovery is given under the proposed workflow model. Compared with the traditional method, the new strategy takes into account both the control dependency and the data dependency between workflow tasks during the failure handling and the failure recovery. What is more, how to implement the workflow task application is explored to improve the efficiency in the failure handling and the failure recovery.

Key words: workflow; failure handling; strong data dependency; strong control dependency; self-recoverable workflow

* Received September 21, 1999; accepted February 25, 2000

Supported by the Natural Science Foundation of Shandong Province of China under Grant No. Q97G01158