用进程代数描述可适应工作流的模型方法

魏丫丫,林 闯,田立勤
(清华大学计算机科学与技术系,北京 100084)

摘 要: 工作流模型技术是当前的一个研究热点,工作流模型必须支持动态可适应性并且能够模型大规模的复杂系统才能被广泛应用.作者首次提出用进程代数——通信顺序进程(Communication Sequential Process,CSP)描述工作流的方法,并给出了模型可适应问题的解决方法.进程代数的合并(composition)特点可以将简单的工作流模型组合成复杂的工作流模型,从而解决复杂系统的模型问题,并为工作流模型的可重用性提供有力的支持.

关键词: 进程代数; 可适应工作流; 一致性; 有效性
中图分类号: TP302 **文献标识码**: A **文章编号**: 0372-2112 (2002) 11-1624-05

Represent and Analyse with Adaptive Workflow by Process Algebra

WEI Ya-ya, LIN Chuang, TIAN Li-qin
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: Today workflow technology is a hot research topic and workflow models should support adaptation and be able to model large-scale and complex systems in order to be widely applied. We first put forward using Process Algebra (PA) - Communication Sequential Process (CSP) to model workflow systems and solve the adaptation of the models. The composition property of PA can be used to construct complex systems from simple systems, so it can solve the difficult problem to model complex systems and support the reusability of workflow models.

Key words: process algebra; adaptive workflow; consistency; soundness

1 引言

近年来,工作流的描述是一个热点问题,一个工作流系统包括一组过程及过程之间的相互依赖关系,过程启动和终止条件,以及对每个过程的描述.目前有很多模型方法来描述工作流模型,常见的有 Petri 网等. Petri 网描述的工作流模型虽然直观,并且一定程度上能解决工作流模型的可适应性问题,但对于复杂系统的描述有一定的局限性.本文提出用进程代数(PA)描述工作流模型的方法.进程代数有很多种,包括 ACP (Algebra of Communicating Processes),CCS (Calculus of Communicating Systems),CSP (Communication Sequential Process)等.文献[1]讨论过用 ACP 描述工作流模型,但只是简单的描述工作流模型,并且不支持模型动态的结构变化(change).本文采用 CSP 描述工作流模型,不仅能够描述复杂系统,同时也支持结构变化的系统.它的模型合并特点是解决复杂系统模型问题的可行方案之一.进程代数描述的特点是:(1)进程代数的等价理论能够简化模型;(2)进程代数概念简单并且描述能力很强;(3)进程代数的合并理论可将简单工作流模型通过操作符组合成复杂模型.

由于现实的系统可能无法事先确定所有的过程,并且过程间的关系在执行过程中并非一成不变的.因此工作流系统要得到广泛的应用就要保证它的可适应性.目前有很多文章采用 Petri 网或其他的描述方法来解决可适应性问题.但是对于复杂的系统,Petri 网描述能力有限.本文的贡献在于给出 CSP 描述工作流的模型方法和动态处理变化的方法,并保证语法和语义上的一致性和正确性[2],共包括两方面的内容:(1)如何描述变化发生后的模型;(2)如何将原模型中执行的实例移交到新模型中继续执行,并保证系统的正确性和一致性.

2 工作流模型的描述

2.1 过程模型描述

工作流系统中每个执行的任务(task)称为一个过程(process),并且过程是系统中的原子操作[3],在通信顺序进程(Communication Sequential Process,CSP)[4]中用大写字母如 P, Q, R 等表示.每个过程执行一个服务提供点(service provider site)提供的服务.过程之间的关系有:

(1)顺序关系(sequence): $P; Q$

(2)并行关系(parallel): $P \parallel Q$

收稿日期:2001-11-30;修回日期:2002-08-26
基金项目:国家自然科学基金(No. 90104002 和 60173012);国家高技术研究发展计划(863 计划)课题(No. 2001AA112080);国家重点基础研究发展规划项目(G1999032707)

(3) 选择关系(conditional):$P \prod Q$

(4) 递归关系(iteration):$P = R; P$

还有两种过程描述符:

ε:空操作,表示过程不做任何动作但能够成功的完成.

√:中止操作,表示过程是否中止,若 P -> √,则表示过程 P 成功的完成并中止.

过程模型描述系统实例过程的执行,一个系统实例是指从过程(start)开始执行,然后按照工作流模型继续执行,直到最后一个过程(end)成功完成. 过程之间不仅有上述依赖关系,还包括一些状态信息来唯一标识过程的状态以及过程之间的依赖关系:

(1) 结构:指任务之间相互依赖的关系,包括顺序,并行,选择,递归;顺序关系 $P; Q$ 表示进程过程 P 成功执行完之后执行过程 Q. 选择关系表示该选择过程选择过程 P 或者 Q,最终只有一个过程被执行. 并行关系 $P \| Q$ 表示过程 P 和 Q 分别独立的执行. 递归是指实例执行一个循环过程.

(2) 数据:指过程读或写的数据. 假定工作流模型中每个过程涉及的数据变量的矢量集为{ v1, v2, …, vk}, Input(P) 表示过程 P 输入要涉及的数据变量. Output(P) 是过程 P 输出所涉及的数据变量.

(3) 资源:表示这个过程执行过程中涉及的其他资源如:人力资源或物力资源. 某个任务涉及的资源用矢量集 resource(P) = {person1, person2, …, person n machine1, machine2, …, machine n} 来表示. 过程执行前操作者将通过修改边界状态来决定过程是否最终被执行.

(4) 约束条件:表示过程所涉及的限制条件如时间限制等. 比如某个过程要求在一定时间内完成,时间可以有绝对时间和相对时间等其他属性. 约束条件用 restriction(P) 表示.

(5) 补偿信息:指一个四元矢量{compensable, compensation-task, redo, undo}. compensable = Y 表示这个过程可以补偿,回滚时执行补偿过程后可以恢复到未执行前的状态,redo = Y 表示在该过程被取消(undo)后可以再次执行(redo).

(6) 日志(log):日志是保证系统完整性的关键性记录,系统为每个实例保存一个日志. 日志记录过程 P 完成(P -> √)前的所有状态,包括执行中的所有服务,访问的所有数据(包括过程读和写的数据). 日志中的每项记录用一个三元集表示{被修改的过程,修改前的状态,修改后的状态}.

(7) 边界:过程 P 的边界有两种可能,用函数 border(P) 来表示. border(P) = {failer, successful}, border(P) = failer 表示边界是失败的,与边界相连的过程仍然不能执行,这个参数用于操作员做控制. 即使过程被列入工作表(worklist)后要得到操作者的允许才能执行,否则不能执行. 当 border(P) = successful 时,表示与此边界相连的后继过程可以执行.

2.2 工作流执行的描述

过程的状态分为非活动的(not-actived),活动的(actived),完成的(completed),中断的(terminated),失败的(failed)和跳过的(skipped). 下面我们对过程的状态给出一个形式化的描述. 首先给出几个定义:

定义 1 与某个过程 P 相关的两个函数 after(P) 和 before(P). after(P):所有和过程 P 直接相联的后继过程 Q. 只要 P 成功的完成,所有 Q 都有可能被执行. before(P):before(P) 中所有过程执行完,过程 P 才有可能被执行.

定义 2 过程 P 的状态定义成一个与时间相关的函数 state(P), state(P) = {not-actived, actived, running, completed, failed, skipped}[5]. 工作流中实例的状态分为三种:初始执行的(initial),正在执行的(running)和成功完成的(completed). 一个过程执行完成(P -> √)后,同时生成该过程的日志,并且生成数据的日志.

下面描述一个实例执行的全过程(假定过程状态转换记为 state1-> state2). 规则如下:

规则 1 当一个工作流实例开始执行时,初始化所有过程,state(P) = not-actived, Border(P) = failer, state(P) = not-actived 表示该过程还没有进入任务表,过程 P 状态被修改为 actived 后被列入任务表.

规则 2 当一个过程 P 成功执行后,修改所有 output(P) 的过程,使得满足限制条件的过程的状态从 not-actived -> actived. 过程的状态 state(P) = actived 并在边界 border(P) = successful 的触发下,过程 P 的状态将转入可执行状态(running),该过程可以执行. running -> completed 表示过程成功执行完毕.

规则 3 过程成功执行完后生成日志,并保存所有执行信息如参与者的数目,完成的任务等,以便对模型进行修改,回滚,和性能分析. 如果得到操作员(operator)的许可,边界条件 border(P) 由 failer -> successful;过程可以马上执行.

规则 4 State(P) = failed 是指某个过程执行失败或被中途取消,则该过程所有属于 after(P) 的过程的输入条件不满足,并删除这个过程其他相关服务,恢复该过程涉及的数据执行前的状态,即保证过程的原子性.

下面举一个例子说明过程执行中状态的变化:一个工作流实例描述为 Sys = start; P; ($Q1 \prod Q2$); end;因此 Before(P) = start, After(P) = {$Q1$, $Q2$}. 过程 P 被执行的前提是 start -> √,当它的限制条件满足时 state(P) 转入 actived 状态. 同理,当 P -> √,且 $Q1$ 和 $Q2$ 的其他限制满足时,$Q1$ 和 $Q2$ 转入 actived 状态.

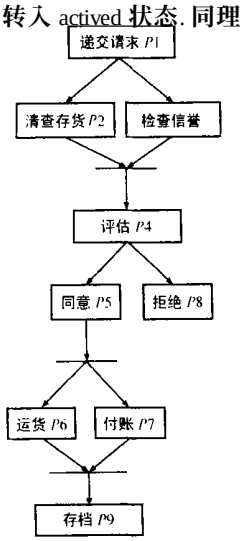

图 1　系统模块图

上面给出了模型的执行过程,系统本身必须满足 soundness 特性,对于 CSP 描述的模型同样也要保证 soundness 特性. Soundness 能保证 WF 语义的完整性.

一个工作流模型是 sound 的,当且仅当这个模型在任何实例下都能正确中止,并且系统不出现死锁,活锁和死任务等. 因此,验证 soundness 特性可以发现模型中一些错误的或者危险的结构,从语义上保证模型的正确性[6]. 对于不满足 soundness 特性的系统模型,通过

语义修改来保证满足 soundness 特性. 采用 CSP 描述一个工作流系统功能模块图如图 1 所示.

这是一个办公系统,每个请求的用户先递交一张请求单,当清查存货以及检查信誉过程执行后,执行评估并给出两种意见:拒绝或者同意. 若同意,则执行两个动作:运货和交钱,结束后存档.

例 1. 系统 M = start; $P1$; ($P2$ ‖ $P3$); $P4$; (($P5$; ($P6$ ‖ $P7$); $P9$) ∏$P8$); end.

初始化:

(1) 所有过程的边界条件置为 failer;

(2) 假定所有过程 X: state(X) = not-actived.

实例执行开始,border(start) = successful,过程 start 由 not-actived - > actived - > running,过程 start 成功执行完后,state(start) = completed,系统不断扫描各节点过程的前置集的状态,当某个节点的前置集中所有过程的状态为 completed,然后检查本身的约束条件是否满足,满足时该节点状态转为 actived,如果该节点没有成功执行,或者因其他限制条件不满足而没有执行,则要修改该节点状态为 not-actived 和边界条件为 failer. Start 执行完后,则有 state(Q) = completed(∀Q ∈before($P1$)). 并且 $P1$ 的状态 state($P1$) 由 not-actived →actived. 当 $P1$ 的时间限制满足时,且访问的数据不出现读写冲突的情况下,得到操作员许可后,state($P1$) →running,过程开始执行. 当 state($P1$) = completed,则 $P2$ 和 $P3$ 的前置集的状态 state(Q) = completed(∀Q ∈before($P2$)),并且 state(Q) = completed(∀Q ∈before($P3$)),那么 $P2$, $P3$ 可能同时执行,然后根据 $P2$, $P3$ 的约束条件判断最终执行哪个过程.

上面给出了一个工作流模型的描述方法. 但工作流模型的一个重要的问题是可适应问题. 下面我们讨论 CSP 描述的工作流模型如何适应模型变化,并提出一套适应变化的最小的和最完备的语义.

3 处理模型变化的方法

由于工作流系统设计者不能预测所有可能发生的变化,并且系统在执行前不一定能确定工作流模型的结构,因此系统模型需要有一定的灵活性(flexibility)来处理这些变化. 工作流变化包括很多种,如资源一级的和过程一级的. 过程一级的变化一般有两种:实例变化和结构变化. 结构变化是指修改过程模型的定义. 实例变化是指实例执行过程中发生的变化,在执行过程中出现执行错误,比如某个系统实例执行过程中,工人罢工将导致这个工作流实例无法正常执行[7]. 由于过程一级的变化是最常见的一类变化,下面我们主要考虑过程一级的结构变化,结构变化有以下几种可能的动作:

(1) Extend:增加一个新过程,将新过程插入到原模型是一类常见操作.

(2) Reduce:删除系统模型的某个过程,或者跳过某个过程(skip tasks)等.

(3) Replace:过程的替换是将原来过程模型中的一个过程替换为另外一个过程.

(4) Relink:过程的重新排序,比如将两个并行执行的过程改为串行执行.

结构变化除了有上述几种特性外,还有时间特性:瞬时的和永久的. 瞬时变化限制该变化只在一段时间内有效. 永久变化是指该变化发生后,新模型将替代原模型,原模型不再有效. 瞬时变化的时间期限到达之后,撤销新模型,原模型重新有效,这样可能会影响一些正在执行的实例. 我们把这种撤销处理等同于对模型再一次引入永久结构变化[8]. 对于结构变化处理有以下几种策略:

(1) Flush:发生结构变化后,当前实例按照原模型继续执行,只有这些实例执行完后,新实例才按照变化后的新模型开始执行;

(2) Abort:结构发生变化后,当前执行的实例被取消,然后按照新模型开始执行新实例;

(3) Migrate:这种策略移交当前所有正在执行的实例,但不采用 Flush 或者 Abort 方法. Migrate 策略对不同实例做不同的处理,正在执行的实例正确移交到新模型中继续执行.

除了以上三类策略外,还有适应(Adapt),和构建(Build)[7]策略. Flush 这种策略可能出现如下情况:所有正在执行的实例执行完之后才能按照新模型执行,但是正在执行的实例要全部执行完毕可能需要很长的时间,显然这种时间消耗对有些系统来说是不能容忍的. Abort 策略也一样,有些系统可能不允许取消所有正在执行的实例. 显然,migrate 是一种更有意义的策略. 系统发生动态变化时正确转交正在执行的实例,同时保证系统的一致性和正确性.

4 一致性(consistency)和正确性(correctness)

4.1 一致性和正确性定义

定义 3 对原模型修改以后,原模型中正在执行的实例如何成功的转换到新模型中继续执行,并保证转换后的执行过程在新模型看来没有错误,这种特性称为一致性.

定义 4 变化后的新模型仍满足 soundness 特性,并且按照新模型重新开始执行的实例不会出错,原模型中正在执行的实例转换到新模型中也能保证一致性. 这种特性称为正确性[9].

为保证系统的一致性和正确性,系统要满足以下特性:

(1) 要保证发生变化后模型语法的完整性,即 soundness 特性. 让分析者通过模型结构来发现模型的一些错误.

(2) 数据流的连贯性. 删除一个过程,可能导致后续过程的输入数据流不连贯. 系统可以通过增加一个过程来提供后续过程的输入数据流,或者删除那些输入数据流不连贯的后续过程.

(3) 语义的正确性. 发生结构变化后系统仍能正确的执行. 简单的从原模型中刚执行过的过程转换到新模型中相应的过程执行,可能会导致错误. 如例 1 所示的系统,系统修改为运货和付账串行,($P6$ ‖ $P7$); $P9$ = > $P7$; $P6$; $P9$. 如果当前实例有一个客户执行到运货完毕 $P6$ 但还没有付账,若此时模型被修改了,该客户转换到新模型的过程 $P6$,过程 $P6$ 已经结束,接着执行过程 $P9$. 这样导致这个实例中的客户只运货没交钱,出现系统错误. 因此转换时要保证系统语义正确

性.

(4) 失败的原子性. 如果一个过程执行失败或者取消时,该过程不能继续执行,为了保证系统正确性,必须撤销这个过程所执行的一切服务,并恢复该过程对数据所做的修改,保证过程失败的原子性[10].

(5) 保证数据流的正确性. 为了恢复过程修改的数据,因此每次修改数据都要保存数据日志. 采用一个二元向量(数据修改前的状态,执行修改的过程)来记录数据元素. 为保证数据流的正确性,要遵守以下两点规则:

①一个过程执行前它所有的输入参数都要被提供,保证数据输入的正确性;

②保证写与写(write-after-write)的正确性,即保证不冲突. 两个并行的分支不能同时写同一个数据.

4.2 动态结构变化

下面讨论系统模型结构变化时如何保证一致性和正确性.

(1) 增加过程. 增加一个过程 P^*,并给出相应的 before(P^*)、after(P^*)和过程 P^*要修改的数据来确定该过程的相对位置和其他参数. 将插入操作用一个函数表示:insert(P^*, before(P^*), after(P^*), (d, mode), state, border, (compensable, compensation-task, redo, undo), restriction).

实施规则:

①先求出原模型中受影响的最小过程集合 M',包括被插入过程 P^*的前置集 before(P^*)(用 $M1$ 表示),后置集 after(P^*)(用 $M2$ 表示)以及 before(P^*)和 after(P^*)之间的所有过程的集合(用 $M3$ 表示)描述.

②若修改 P^*与 $M3$ 选择执行或者并行执行,即将 $M3$ 修改成 $(M3 \prod P^*)$ 或者 $(M3 \parallel P^*)$. 如果在连续执行的两个过程间插入一个过程,也可以表示为 $\varepsilon \prod P^*$或者 $\varepsilon \parallel P^*$.

③将插入过程 P^*的边界初始化为 border(P^*) = failer, state(P^*) = not-actived.

④扫描所有受变化影响的过程,并修改它们的前置集与后置集.

对于变化后的模型语义的描述,采用精化规则可以简化语义,并保证语义间的等价性.

定理 1　下面两种等价结构都是成立的.

$(a)\ A;(X \parallel \varepsilon);B \Leftrightarrow A;X;B$

$(b)\ A;\varepsilon;B \Leftrightarrow A;B$

证明　在(a)中表达式左式表示:执行过程 A 后可能执行过程 X 和空过程 ε,过程 $X \parallel \varepsilon$ 执行完之后才能执行过程 B,过程 $X \parallel \varepsilon$ 在外观看来和过程 X 的效果是一样的,所以 $A;(X \parallel \varepsilon);B$ 和 $A;X;B$ 在外观看来是一样的,因此 $A;(X \parallel \varepsilon);B \Leftrightarrow A;X;B$.

同理,在(b)中左式表示过程 A 执行后继续执行空操作然后执行过程 B,在外观看来和 $A;B$ 执行效果一样,因此 $A;\varepsilon;B \Leftrightarrow A;B$.

(2) 删除过程. 删除是一个常见的修改操作. 要保证一致性与正确性,要遵守以下规则:

* 不能删除开始(start)和终止(end)过程,这将导致系统无法开始和中止. 删除操作要有意义,比如 start; X; end,删除 X 就没有意义.

* 删除的任务 X 的前置集(before(X))中的活动所关联的数据流 data(before(X))是否等于 data(after(X)). 如果不等,说明如果一个实例执行到 before(X)后就会出错,导致后续过程参数丢失,应采取相关措施保证一致性:

①用户增加一个过程 X',过程 X 为后续的 after(X)提供相应的数据流.

②用户删除一些 before(X)里的过程,保证所有过程的输入输出数据流匹配.

③对于已经执行完成的过程不允许删除,这类删除在很多情况下没有意义.

定理 2　在满足上述条件情况下删除一个过程 P 的方法:将原模型中过程 P 出现的地方用 ε 代替,并采用精化规则后得到的新模型. 这种删除操作能保证工作流系统的语法和语义正确性.

证明　假定系统删除一个过程 P,按照上述操作删除了过程 P 和过程 P 提供的一切相关服务. 替代的空操作 ε 不影响工作流模型的正确中止,不影响工作流模型的 soundness 特性. 采用精化规则后消去空操作 ε 带来的影响,上面已经证明了精化操作满足一致性和正确性要求,因此删除操作能够保证模型正确性和一致性.

(3) 改变过程相对执行顺序. 常见的是将并行的过程改为串行的,或者相反等. 相应的处理方法为:

①从原模型中提取变化的过程段的描述记为 M^*.

②改变相对顺序也是一个插入与删除的过程,首先删除原模型中相应过程,该过程用 ε 代替,然后做相应的精化处理. 插入过程按照增加操作来处理. 将得到的模型精化处理,并修改受变化影响的过程集 M 的前置集与后置集.

如对例 1 模型中的清查存货 $P2$ 和检查信誉 $P3$ 实行串行化,即过程 $P3$ 执行后才能执行过程 $P2$. 原模型 M = start; $P1$; ($P2 \parallel P3$); $P4$; (($P5$; ($P6 \parallel P7$); $P9$) $\prod P8$); end. 修改步骤为:

(a) $M^* = P1;(P2 \parallel P3);P4$;修改为 $P1;(P2 \parallel \varepsilon);P4$,并精化为 $P1;P2;P4$.

(b) $P3$ 在 $P2$ 前,并与 $P2$ 串行执行,则在 M^*中增加过程 $P3$,before($P3$) = $P1$, after($P3$) = $P2$;插入过程 $P3$ 后 $P1;P3;P2;P4$.

(c) 扫描所有受变化影响的过程集 $M' = \{P1, P2, P3, P4\}$ 的前置与后置集. 得到 after($P1$) = {$P2$}, after($P2$) = {$P4$}, before($P2$) = {$P3$}, before{$P4$} = {$P3$}.

得到语法描述正确的新模型后,实例转换采用 Migrate 策略,并且这种转换不会产生系统错误,保证系统的正确性和一致性的策略称为移交策略. 为了正确的移交正在执行的实例,有三个基本策略:

* change-over:对于正在执行的实例,分为两种:顺从的(compliant)和非顺从的(non-compliant). 如果正在执行的过程没有进入 M^*,也即变化没有影响正在执行的实例或者实例执行的是一条没有变化的分支. 则一个工作流实例当前的执

行状态对于新模型来说是安全的,我们称这种情况为顺从(compliant)情况;显然这种情况下可以将原模型中的实例转换到新模型中去,正在执行的实例不受影响并能保证系统的一致性和正确性.

* rollback:如果当前执行的实例不满足顺从情况,属于非顺从的情形.如果非顺从的实例已经进入 M^* 或者已经执行完 M^* 中的过程,系统可以执行回滚[11].对于能够执行回滚的系统,回滚能保证系统的可恢复性和一致性.

* 对于正在执行的非顺从实例,如果加上一些限制或者一些状态约束后,仍然能够成功的从旧模型转化到新模型中去执行,这也是一种有效的策略.但是显然这种情况和具体的系统有关,如 $B;C$ 变化为 $B \parallel C$ 是一种潜在的顺从结构.

工作流系统除了要提供模型的可适应性之外,还要提供可靠性.比如一个服务提供点失败了,则所有访问该服务的过程将不能执行,除了做出一些补偿性过程以外,过程最好有转向功能,正在执行的实例可以立刻转向另一个服务提供点继续执行,来保证系统的可靠性.

5 总结与展望

本文提出了 CSP 描述工作流模型的方法.主要讨论了如何描述工作流模型的执行过程和工作流结构变化时的处理方法.

本文对最常见的结构变化给出了相应的处理方法,并保证变化后系统的一致性和正确性,使得 CSP 描述的工作流模型具有广泛的应用性.CSP 描述的工作流有它自己的特点,从进程代数的描述能力来看,比其它描述工具(如 Petri 网)来说,CSP 模型方法具有简单,描述能力强的特点.该模型方法还可以利用合并的特点来描述复杂系统,并且能够支持可适应工作流系统.

参考文献:

[1] Wong K F, Low B T and Yongjie R. A workflow model for chinese business processes[J]. International Journal of Computer Processing of Oriental Languages, 2001, 14(3): 233 - 258.

[2] Van der Aalst W M P, Basten T, Verbeek H MW, Verkoulen P A C, Voorhoeve M. Adaptive workflow on the interplay between flexibility and support[A]. In: Filipe J; Cordeiro J: Proceedings of the first International Conference on Enterprise Information Systems[C]. Setúbal, Portugal: March 1999. 2, 353 - 360.

[3] Van der Alast W M P, Jablonski S. Dealing with workflow change: identification of issues and solutions[J]. International Journal of Computer Systems, Science, and Engineering, 2000, 15(5): 267 - 276.

[4] Mantis H M C. Communicating sequential processes: a synopsis. http://citeseer.nj.nec.com/247709.html[DB/OL].

[5] Ellis C A, Karim K. A workflow change is a workflow[A]. Van der Aalst W M P, Desel J, and Oberweis A, editors. Business Process Management: models, Techniques, and Empirical Studies[C]. Volume 1806 of Lecture Notes in Computers Science. Springer-Verlag, Berlin: 2000.

[6] Van der Aalst W M P. The application of Petri nets to workflow management[J]. Journal of Circuits, Systems, and Computers, 1998, 8(1): 21 - 66.

[7] S W Sadiq. Workflows in dynamic environments? Can they be managed[A]? Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99)[C]. Woollongong, Australia: 1999. 27 - 28.

[8] Reichert M, Peter D. ADEPTflex-Supporting dynamic changes of workflows without loosing control[J]. Journal of Intelligent Information Systems (JIIS), Special Issue on Workflow and Process Management, 1998, 10(2): 93 - 129.

[9] Ellis C, Keddara K, and G Rozenberg. Dynamic change within workflow systems[A]. In Comstock N and Ellis C, editors, Conference on Organizational Computing Systems[C]. ACM SIGOIS, ACM, Milpitas, CA: Aug 1995.

[10] Cichiocki A, Rusinkiewicz M. Migrating workflows[A]. In NATO-ASI, Advances in Workflow Management Systems and Interoperability[C]. 1997.

[11] Sadiq S W, Maria E O. Dynamic modification of workflows[R]. Technical Report No. 442. Department of Computer Science and Electrical Engineering, University of Queensland, Brisbane, Australia. October 1998.

作者简介:

魏丫丫 女,1979 年 9 月出生于湖北公安,清华大学博士研究生,感兴趣的方向有计算机网络,系统性能评价,随机 Petri 网,随机进程代数等.

林 闯 男,1948 年 7 月生于辽宁省,博士,清华大学计算机系教授,博士导师,计算机网络研究所所长,同时为《计算机学报》编委,国家自然科学基金重大研究计划"网络和信息安全"的科学指导专家,中科院网络中心和北京科技大学兼职教授.主要研究领域为计算机网络,系统性能评价,随机 Petri 网,逻辑推演和推理系统.曾在美国 Purdue 大学、美国 Texas 大学奥斯订分校、香港科技大学、香港浸会大学和美国 Florida 大学长期做研究工作.已在国内外一级期刊上和 IEEE Computer Society 的学术年会上发表论文 90 多篇,并在国内出版专著 2 本.

田立勤 男,1970 年 10 月出生于陕西定边县,硕士,讲师,研究方向是计算机网络,工作流模型和系统性能评价.