软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(3):417−438 [doi: 10.3724/SP.J.1001.2011.03962]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

过程感知信息系统中过程的动态演化*

宋 巍[1,2+], 马晓星[1], 胡 昊[1], 吕 建[1]

[1](计算机软件与新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京理工大学 计算机科学与技术学院,江苏 南京 210094)

Dynamic Evolution of Processes in Process-Aware Information Systems

SONG Wei[1,2+], MA Xiao-Xing[1], HU Hao[1], LÜ Jian[1]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Institute of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China)

+ Corresponding author: E-mail: wsong@mail.njust.edu.cn

Song W, Ma XX, Hu H, Lü J. Dynamic evolution of processes in process-aware information systems. *Journal of Software*, 2011,22(3):417−438. http://www.jos.org.cn/1000-9825/3962.htm

Abstract: Supporting the evolution of process model at run-time stage and propagating the changes of process model to the active process instances are fundamental requirements for any flexible process-aware information systems (PAIS). Instance migration is regarded as the mainstream technique that deals with the dynamic evolution of process-aware information systems. By using this technique, active process instances are migrated to the modified process model, so that they can benefit from the optimization process. The challenges are 1) to guarantee that the process instances can be correctly migrated to the modified process model; 2) to efficiently check whether the process instances can be migrated. From this view, this paper surveys the state-of-the-art dynamic evolution of processes in PAIS. Finally, some potential research directions on evolution of processes are proposed for future references.

Key words: process-aware information system; process model; dynamic evolution; process instance migration; correctness; high efficiency

摘 要: 支持过程模型的动态修改并将过程模型的修改传播到当前正在运行的过程实例上,是柔性过程感知信息系统的基本要求.过程实例迁移是应对过程感知信息系统中过程动态演化的主流技术途径,它将尚未执行结束的过程实例动态地迁移到修改后的过程模型上继续执行,从而使得这些过程实例可以享受到过程优化带来的便利.过程实例迁移的挑战在于保证过程实例迁移的正确性以及过程实例迁移检验的高效性,以此为切入点,综述了过程感知信息系统中过程动态演化技术的研究进展.最后,展望了过程演化技术未来应当关注的研究方向.

关键词: 过程感知信息系统;过程模型;动态演化;过程实例迁移;正确性;高效性

中图法分类号: TP311 **文献标识码**: A

* 基金项目: 国家自然科学基金(60736015, 60973044, 61003019); 国家重点基础研究发展计划(973)(2009CB320702); 国家高技术研究发展计划(863)(2009AA01Z117); 核高基重大专项(2009ZX01043-001-06)

收稿时间: 2010-04-22; 修改时间: 2010-06-10; 定稿时间: 2010-11-03

CNKI 网络优先出版: 2010-12-08 15:17, http://www.cnki.net/kcms/detail/11.2560.TP.20101208.1517.000.html

© 中国科学院软件研究所 http://www.jos.org.cn

随着软件系统规模和复杂度的日益提高,编程的关注点从小规模编程(programming-in-the-small)逐步过渡到大规模编程(programming-in-the-large)[1].小规模编程关注于简单应用程序或单个模块的开发,而大规模编程关注于应用程序或模块间的组合与协同.通过复用已有的应用程序或模块,大规模编程能够便捷地开发符合要求的复杂软件系统.业务过程的建模和制定可以看作是一类大规模编程方式[2](例如 BPEL[3]编程),业务过程模型(简称过程模型)描述了一系列的活动以及活动之间的执行顺序.依据业务过程,企业级的应用程序可以有效地组合在一起,从而实现预期的业务目标.

企业应用程序集成(包括 B2B 集成)的协同逻辑通过业务过程建模的方式被“显式”地刻画并维护,而不是固化在代码当中.由于过程模型保留了原先开发者对系统的认识,这种设计方式可以使企业应用程序集成能够快速地响应需求(目标)和环境的变化[4,5].过程模型显式化的思想在企业应用程序集成、工作流管理、计算机支持的协同工作以及 Web 服务组合等大规模编程领域得到了广泛应用[6−8].针对这些应用领域的共性,学术界凝炼出过程感知信息系统(process-aware information system,简称 PAIS)[6,7]的概念:PAIS 是一个以过程模型为基础,管理和执行一个包括人、应用程序和信息资源的可操作过程的软件系统[6,7].

随着 Internet 成为主流的计算平台,PAIS 的运行环境也开始由企业内部封闭、静态与可控逐步走向了跨企业间的开放、动态与难控,这要求 PAIS 必须具有动态演化的能力[9].PAIS 是一个复杂的系统,其演化可能涉及过程、数据、业务规则等多个方面.由于过程模型在 PAIS 中所起的基础性作用,数据、业务规则等方面的演化可以通过过程的演化而予以体现,因此,业界重点关注于 PAIS 中过程的动态演化.PAIS 过程动态演化面临一些理论和技术上的挑战,其中一些技术难题尚未完全解决.文献[10]仅从过程演化正确性准则的角度综述了 2004 年之前过程演化方面的工作,但是并没有涉及过程动态演化的其他方面.近年来,随着服务计算等新型计算风范(paradigm)的出现,过程动态演化的研究得到了进一步发展.本文旨在从 PAIS 过程动态演化所面临的问题与挑战入手,提出一些与过程动态演化相关的技术框架,较为全面地综述了 PAIS 中过程动态演化方面的相关研究,希望能为 PAIS 动态演化的进一步研究提供一定的参考价值.

本文第 1 节概述 PAIS 中过程动态演化所面临的挑战.第 2 节~第 4 节分述现有方法与技术如何应对这些挑战.第 5 节总结全文,并指出 PAIS 中过程静态和动态演化未来应当关注的研究方向.

1 过程感知信息系统中过程的动态演化概述

1.1 背景知识

PAIS 这类软件将过程模型显式化,使得过程模型与可复用的应用程序相分离.其中,过程模型对应于大规模编程,它描述了过程中活动(任务)之间的结构关系,如顺序、选择、并行等,而应用程序层则是一个由可复用软件实体(例如 Web 服务、Java 组件等)构成的集合,集合中的软件实体是松散耦合的.实际的可操作过程可以通过为过程模型的各个活动绑定相应的软件实体而得到[6,7].过程模型与应用程序之间通过过程引擎(process engine)桥接.引擎生成并运行过程模型的实例(process instance),负责选取过程实例中各个活动的执行软件实体并监控过程实例的运行.与操作系统中进程的定义类似,过程实例是一个动态的概念:过程实例可以看作是相应过程的一次具体实施.活动执行实体的绑定遵循信息隐蔽(information hiding)的原则,只要执行实体满足活动的接口要求,执行实体的内部实现以及演化不会对过程以及过程实例产生影响.同时,过程模型显式化遵循关注分离(separation of concerns)的原则,当系统需要演化时,只需修改过程模型而无须修改底层的应用程序.例如,若将过程模型中原本顺序执行的两个活动改为并行执行,这一变动并不需要对绑定到活动上的执行实体进行修改.

为应对需求和环境的不断变化[9],PAIS 过程应当具有自主演化的能力.PAIS 的过程演化是指过程在交付使用之后,为了能够适应环境的变化、持续满足用户的需求而经历的一系列变更的过程.过程演化又区分为静态演化和动态演化两个方面:过程的静态演化主要是指根据需求和环境的变化,使用一些演化操作对过程模型进行修改,使过程模型从一个版本升级到另一个版本的过程;而过程的动态演化主要是指当过程模型发生改变时,将过程模型的相应变化动态地传播到正在运行的过程实例上的过程.PAIS 的分层结构保证了其过程静态演化可以较为容易地实现.不同的过程版本对应着不同的过程模式(process schema),本文将过程版本升级前后的过

© 中国科学院软件研究所 http://www.jos.org.cn

程模式分别称为源模式(source schema)与目标模式(target schema).

过程的动态演化主要有两类情形[11,12]:特设式(ad-hoc)演化和进化式(evolutionary)演化.特设式演化只影响特定的过程实例,它通常是由过程实例在运行过程中遭遇异常(exception)而引发的.例如,当负责执行某一活动的软件实体响应过慢,为保证相应过程实例能够在截止日期前完成,该过程实例可以跳过某些非关键活动的执行.这种个案处理的演化方式称为特设式演化,仅作用于特定的过程实例(只需要修改此过程实例所对应的过程模型),而不会影响到其他过程实例.注意:若过程实例的动态微调不会影响到过程模型以及后续的过程实例,那么这种过程实例的动态微调不是动态演化,但可视为过程的一种柔性机制[13].过程实例的进化式演化是由过程模型的修改与升级而引发的,模型的修改必然会对所有正在运行中的过程实例产生影响,这种由于过程模型升级所导致的过程实例演化称为进化式演化.特设式动态演化还是进化式动态演化的主要区别在于它们的起因不同,然而根据 PAIS 过程模型驱动过程实例执行的特点,这两种类动态演化的实现机理是相似的[11,12]:它们都需要首先进行过程的静态演化,而后进行过程的动态演化,即将过程模型的变动传播到正在运行的过程实例上.不失一般性,本文以过程实例的进化式演化为例来综述过程实例的动态演化.

随着服务计算风范的出现,PAIS 中过程的含义有了新的发展.过程编制(process orchestration)从单个企业或组织的视角定义了一个过程的内部实现[14];而过程编排(process choreography)则从全局的视角描述了参与协作的各个过程间的对等消息交互[14],在本文中,过程编排又称为过程协作.过程编制和过程编排都可视为一种大规模编程方式.过程的动态演化可以发生在编制和编排两个层面上.

过程模型的制定,依赖于需求分析的结果以及预定义的环境信息.在运行阶段,PAIS 的内外部需求以及运行环境往往会发生变更,从而导致过程模型的修改与升级,进而引发过程实例的进化式演化.需求和环境的变更具体表现为:业务过程的再工程(business process reengineering)、新的法律法规的出现、用户需求的变更等.新生成的过程实例应当按照目标模式执行,而如何处理源模式所遗留的过程实例是一个具有挑战性的问题,若处理不当容易引发过程模型版本管理混乱、过程运行与过程模型定义脱节等问题.一般而言,有 3 类处理方式[15]:(1) 按照目标模式从头开始执行,这类方式舍弃了已执行部分的运行结果;(2) 按照源模式继续执行,这类方式要么不能使当前的过程实例享受新模式带来的好处,要么不能满足新的法律法规或新的用户需求;(3) 尽可能地将当前的过程实例动态地迁移到目标模式下继续执行,这类方式避免了上述两类方式的缺点,成为过程实例动态演化的一种主流实现途径.注意:虽然可以采用个案处理的方式来应对过程实例的进化式演化,但当待迁移的过程实例数目较多时,需要多次重复同样的修改操作,这种处理方式低效且易错[16].而实例迁移机制无须逐一地修改过程实例,只需检验过程实例能否迁移到目标模式下运行即可.

过程实例能否迁移取决于过程实例的当前状态,过程实例的当前状态是由过程实例的变量及其当前取值(名值对)共同确定的.过程实例可迁移的前提条件是:该过程实例由当前状态迁移到目标模式后能够继续正确执行.原则上说,任意过程实例都可以迁移.对于那些不满足迁移正确性的过程实例,可以采用回退和补偿[17,18]等机制使得过程实例恢复到一个可迁移状态(极端情况下可能回退到初始状态),从而,那些不可迁移的过程实例也变得可迁移[19,20].或者采用特设式的演化方式,由过程建模人员或程序员为每个不可迁移的过程实例量身打造一个专门的目标模式[19,21].然而,这涉及一个代价问题,只有当过程实例迁移所带来的收益大于补偿等措施所付出的代价时,相应的过程

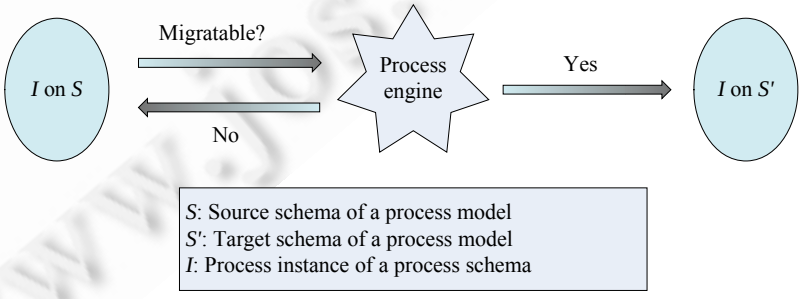


Fig.1 Process instance migration

图 1 过程实例迁移

© 中国科学院软件研究所 http://www.jos.org.cn

实例才有必要迁移.在迁移机制下,可迁移的过程实例按照目标模式继续执行,而那些不可迁移的过程实例仍旧在源模式下执行(如图 1 所示).

1.2 过程动态演化面临的挑战

为支持 PAIS 过程的动态演化,需要在过程引擎中添加一个单独的模块,用于过程实例的可迁移性检验以及可迁移过程实例的状态映射.因此,过程引擎就具有了两方面的职能:(1) 基本职能:过程实例的创建与执行;(2) 演化职能:过程实例的迁移.由于目标模式和源模式共享一个执行引擎,对于可迁移的过程实例,它到目标模式的状态映射是一个实现细节问题,因此,本文主要关注于 PAIS 过程动态演化理论层面的问题.

过程实例迁移的挑战在于保证过程实例迁移的正确性以及实例迁移检验的高效性[10].过程实例迁移是否正确,直接关系到过程实例能否迁移到目标模式下继续执行.从业务过程的角度看,过程实例迁移的正确性主要体现在两个方面:(1) 通用正确性:迁移过后过程实例依然可以顺利执行,而不会发生执行阻塞或流产等故障,若过程参与到一个全局的过程协作,过程实例的迁移不能影响到与其他过程的交互;(2) 特定应用正确性:过程实例从旧过程切换到新过程之后不会违反应用领域内的一些业务规则和用户的特定需求.过程实例迁移的检验效率也很关键,它关系到 PAIS 的服务质量.大多数 PAIS 系统是一类反应式系统(reactive system)[22],这类系统经常需要与用户或协作方进行交互.当 PAIS 过程需要演化时,为提高正在源模式下运行的过程实例与目标模式需求的匹配性,已运行过程实例需要立即暂停执行,直到过程实例迁移的正确性检验完毕之后方可恢复执行.当待迁移的过程实例较多时,系统演化可能会导致系统暂停服务的时间过长.这种类似于拒绝服务(denial of service)的情形对于用户与协作方来说是难以接受的,因此,还需保证实例迁移检验的高效性.该问题对那些具有实时性的 PAIS 过程尤为重要.

现有研究主要关注于过程实例迁移的通用正确性.不难发现,如果目标模式的正确性不能保证,那么迁移后的过程实例的正确性就无法得到保证.目标模式的正确性,是过程实例迁移正确性的前提条件.因此,应该首先明确过程模型的正确性定义.尽管存在多种不同的过程模型,但它们都是从控制流(control flow)和数据流(data flow)的角度来定义过程模型的正确性.过程模型的控制流正确性关注于过程模型的结构正确性,例如无死锁、弱终止性(weak termination)等;过程模型的数据流正确性则关注过程模型中活动之间数据依赖的可满足性,例如需要避免过程模型中某一活动的输入没有事先定义的情形.需要说明的是,过程模型的正确性定义的严格性与其验证的复杂性往往是一对矛盾,因此在实际应用中,我们应该依据需求制定合适的正确性定义.例如,在工作流管理(workflow management)领域,合理性(soundness)[23]从控制流的角度给出了过程模型(工作流网 WF-net)的正确性定义.为使过程模型的正确性定义具有通用性,合理性仅从过程模型的结构角度进行定义,具体包括 3 方面的内容[23]:弱终止性(无死锁和活锁)、恰当终止性(无悬而未决的状态)、无死活动(任何活动都有机会发生).在合理性基础上,还需进一步考虑数据流正确性,以避免一些常见的数据流错误[24].需要说明的是,当某一过程参与到一个全局的过程协作中时,若该过程需要单独演化,不仅需要保证该过程目标模式的正确性,还要保持过程协作的正确性.因为直接验证过程协作的正确性较为复杂,并且协作方有时并不会公开其私有过程(private process)[18,25],所以当某一过程发生演化时,我们需要一种局部的检查方法来保证用目标模式取代源模式不会影响全局协作的正确性(伙伴服务不受影响)[26,27].

然而,即便目标模式是正确性的,仍然存在着过程实例迁移的正确性不能得到满足的情形.例如,当某一过程实例迁移到正确的目标模式后,该实例的执行可能进入了一个死锁状态,这往往是由于过程模型的演化过于激进,从而导致过程实例已执行过的部分与目标模式的定义不匹配而造成的.如果该过程实例的执行不可回退或者执行效果不可补偿,那么该过程实例不能进行迁移而应在源模式下继续执行.这就引入了一个研究问题:待迁移的过程实例满足怎样的正确性条件才可迁移到目标模式下?过程模型的正确性定义启发了研究者:过程实例迁移的通用正确性可以从不引入控制流错误和数据流错误的角度来定义[28].本文将实例迁移过程中所引发的控制流错误和数据流错误统称为动态演化错误(dynamic change bug)[11,28].例如,Aalst 认为,若过程实例的迁移不会引入死锁或活锁、跳过活动、重复执行活动等动态演化错误[10],那么该过程实例就是可迁移的.对于待迁移的过程实例而言,由于它首先参照源模式执行而后参照目标模式执行,并且在源模式的何种状态下切换以及

© 中国科学院软件研究所 http://www.jos.org.cn

切换到目标模式的何种状态下均不易确定,因此直接验证待迁移过程实例的正确性十分困难,从而很难检验待迁移过程实例是否可以迁移到目标模式下.实际上,因为过程实例的迁移需要将该实例在源模式下的执行状态映射到目标模式,对于一个一般的程序而言,确定某一状态映射所获得的目标状态是否有效(valid)是一个不可判定问题[29],其中,有效状态能够保证程序的执行,可以在有限时间内到达目标流程下的一个可达状态.上述结论表明,我们无法直接验证过程实例迁移的正确性.但是,我们可以为过程实例迁移的正确性找到一些充分条件,从而利用这些充分条件来判断过程实例是否可以迁移.本文将过程实例迁移的正确性定义作为检验过程实例可迁移性的源准则,而将源准则的充分条件作为检验过程实例可迁移性的目标准则,简称过程实例的可迁移性准则(migratability criteria).

过程实例迁移正确性的充分条件使得过程实例的可迁移性检验具有了切实的可操作性,这只是问题的一个方面.另一个方面在于实例迁移的速度问题[10,30].在实际项目中我们发现,过程实例迁移速度主要受以下 3 方面因素影响:(1) 可迁移性检验首先需要读取过程实例的已执行活动序列信息以及演化前后的过程模型信息,而这些信息可能持久化在系统的数据库中,从数据库中读取这些信息会花费一些时间;(2) 当获取了上述信息后,我们可利用现有的可迁移性检验方法来判断过程实例的可迁移性.现有过程实例的可迁移性检验方法时间开销很大,有些方法甚至是指数级的[31,32].当待迁移的过程实例很多时,实例可迁移性检验的效率问题会变得尤其突出;(3) 如果过程实例不可以迁移,那么过程引擎只需在演化前的过程模型上恢复过程实例的执行;如果过程实例可以迁移,此时需将过程实例的当前状态(实例变量的取值)映射到演化后的过程模型上.其中,因素(1)和因素(3)多依赖于具体的实现平台与技术,而因素(2)涉及更多的理论和方法.因此,就过程实例的迁移速度问题,我们仅关注过程实例可迁移性的高效检验.若过程实例可迁移,还需要进一步确定过程实例迁移的有效目标状态,即过程实例迁移到目标模式的何种状态下恢复执行.通过分析我们注意到,在现有的绝大部分方法中,过程实例的可迁移性检验与目标状态的确定紧密相关.因此,本文将这两方面问题放在一起进行分述.

通过上述分析,我们归结出 PAIS 过程动态演化需要考虑的 3 方面关键问题:(1) 目标模式的正确构建;(2) 过程实例可迁移性准则的设定(需保证过程实例迁移的正确性);(3) 过程实例可迁移性的高效检验.其中:前两个方面在侧重于过程实例迁移正确性的同时,也会影响到过程实例迁移的效率;而第(3)个方面关乎过程实例迁移的效率(如图 2 所示).

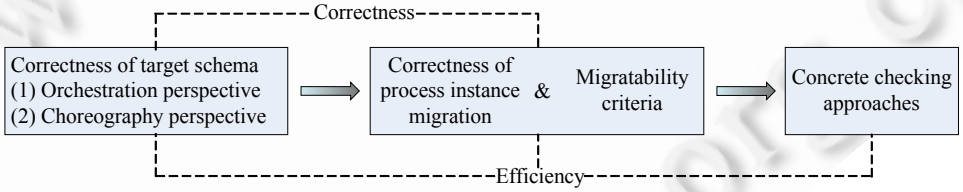


Fig.2 Three key issues involved in the dynamic evolution of processes

图 2 过程动态演化需要考虑的 3 方面关键问题

2 目标模式的正确构建

从源模式到目标模式的转变虽然是一种静态演化,但却是过程实例动态演化的基础.为保证过程实例迁移的正确性,需要保证目标模式的正确性.同时,如果演化过程参与到一个过程协作时,还必须保持过程协作的正确性,该问题可以转化为演化过程的目标模式与源模式的一致性(consistency)检验问题.

2.1 目标模式的正确性

过程模型通常用可视化的标记语言来表示.例如:面向领域专家的非形式化建模工具,如业务流程建模标注(BPMN)、事件驱动过程链(EPC);面向软件设计人员的形式化与半形式化工具,如 UML 活动图、有向图、Petri

© 中国科学院软件研究所 http://www.jos.org.cn

网、进程代数等以及在实现层面面向大规模编程的 BPEL.经过业界的共同努力,这些过程模型大多能够相互转化[33−37].在学术界,Petri 网被公认为是一种好的过程模型建模语言[23,38],原因在于:(1) Petri 网的图形化表示;(2) Petri 网拥有严格的语义和形式化的分析技术;(3) Petri 可以显式刻画过程实例的控制流执行状态;(4) Petri 网可以方便地描述多种过程模式[39].由于本文会涉及一些具体的理论与技术细节,为表达严谨,本文拟采用 Petri 网来建模过程模型.因此,首先对 Petri 网的基础知识[38]作一简单介绍.

Petri 网是一个三元组 $N=(P,T,F)$,其中:$P=\{p_1,p_2,\ldots,p_m\}$为库所集合;$T=\{t_1,t_2,\ldots,t_n\}$为变迁集合,且满足 $P\cap T=\Phi$;$F\subseteq(P\times T)\cup(T\times P)$为连接库所和变迁的有向弧集合.Petri 网本质上是一个有向二分图,其中,库所用圆圈表示,变迁用方框表示,库所(变迁)只能通过有向弧与变迁(库所)相连.上述定义只描述了 Petri 网的结构部分,为表达 Petri 网的执行状态,托肯(token)被引入到 Petri 网中.这些托肯是一些标志,用小黑点表示.托肯只能包含在库所当中,托肯在 Petri 网库所中的分布即表示 Petri 网的状态,称为标识(marking).M 常用来表示 Petri 网的标识,它是一个 m 维向量,其中,分量 $M(p)$表示库所 p 中的托肯数,一般用 M_0 表示初始标识.$M=[p_i,2p_j]$表示在标识 M 下,库所 p_i 和 p_j 中分别含有一个和两个托肯,而其余库所中不含托肯.本文用二元组(N,M)表示 Petri 网 N 下的标识 M.在标识 M 下,若变迁 t 的每个输入库所都含有一个托肯,则 t 在标识 M 下有触发权(enabled),记为$(N,M)[t\rangle$.具有发生权的变迁可以触发(fire),变迁 t 发生会消耗掉 t 的输入库所中的托肯,同时在 t 的每个输出库所中添加一个托肯,从而获得一个新的标识 M'.上述过程可以表示为$(N,M)[t\rangle(N,M')$.同样地,在标识 M 下,若变迁序列σ中的变迁可以接连获得触发权,记为$(N,M)[\sigma\rangle$;若σ中的变迁全部触发后获得一个新的标识 M',该过程可以表示为

$$(N,M)[\sigma\rangle(N,M').$$

本文以基本 Petri 网为例说明业务过程的建模、验证以及演化问题.在处理一些实际问题时,基本 Petri 网的表达能力确实不足以支持实际业务流程建模.在这种情形下,我们可以采用高级 Petri 网、时间扩展的 Petri 网等[23,34,40].这需要根据所面临的问题选择合适的 Petri 网(或者其他模型)来建模业务流程.例如,如果实际问题涉及业务过程的时间行为特性分析或活动实时调度[40−42],那么可以选用时间扩展的 Petri 网对业务过程建模.我们可以利用现有的技术[33−35],将 EPC、UML 活动图、有向图以及 BPEL 等形式的过程模型映射为基本的 Petri 网.其中,Petri 网中的变迁表示过程中的活动,而标识表示某一过程实例的控制流状态[23].控制流状态标记了相应过程实例的执行进度,它的作用与程序计数器(program counter)的作用类似.

为保证目标模式的正确性,需要考虑两方面的问题:(1) 如何定义过程模型的正确性;(2) 如何验证目标模式的正确性.过程模型的正确性准则可以从两个层面进行定义:(1) 通用正确性:不涉及过程模型中各个活动的语义(semantics),仅从过程模型的结构(控制流)和数据依赖(数据流)的角度定义过程模型的正确性;(2) 特定应用正确性:考虑过程模型中各个活动的具体语义,将过程模型所应满足的领域知识(domain knowledge)[43,44]或用户指定的特定需求也列为过程模型的正确性定义.举一个特定应用正确性的例子:在医用工作流领域,病人不能同时使用 Aspirin 和 Marcumar 这两类药物.这些领域知识或用户需求可以用计算机易识别的规约(specification)或约束(constraint)的形式进行表示.此外,过程的原子性(atomicity)特性需要保证过程模型中任意一条路径上不会有一个不可重试的活动发生在一个不可补偿的活动之后[16,17].就通用程度和使用范围而言,过程的原子性介于通用正确性和特定应用正确性之间.但由于判断过程的原子性需要考虑过程中各个活动的可补偿性(compensability)以及可重试性(retriability)等语义信息,因此本文将过程的原子性归入特定应用正确性范畴.

当需要修改过程模型时,存在两种方法用以构建正确的目标模式:(1) 根据变动后的用户需求以及过程执行环境信息,从头开始重新构建目标模式或者对源模式进行修改而得到目标模式.然后检验目标模式,若目标模式不满足规定的正确性准则,则返回继续修改直到目标模式满足正确性为止;(2) 根据用户最新的需求以及过程执行环境的变化信息,在过程模型源模式的基础上进行修改(这些修改操作能够保持过程模型的正确性)以获得目标模式,这类方法被称为 correctness by construction.现有工作大多采用第 2 类处理方式[18,32,45−48],因为它可以使过程模型的修改以增殖的(incremental)方式进行,修改后所得到的目标模式仍旧是正确的.当过程实例需要演化时,第 2 类处理方式为过程实例的迁移节省了时间,从而可以提高过程实例迁移的效率.过程模型所支持的增殖式修改操作与过程模型的正确性定义有关,过程模型的正确性定义越严格,它所支持的修改操作就越受限

© 中国科学院软件研究所 http://www.jos.org.cn

制,因此需要在它们之间加以权衡.

由于通用正确性是最基本的正确性准则,现有工作大多关注于过程模型的通用正确性,而除过程的原子性外[17,18],特定应用正确性关注还较少[43,44].业界普遍认为,过程模型的通用正确性应该从控制流和数据流的角度进行定义.过程模型的结构信息(活动执行顺序上的依赖关系)即反映了过程模型的控制流,而为了分析过程模型的数据流,需要将变量(数据)添加到过程模型中.目前有两类作法:(1) 将变量以及对变量的读写操作作为一阶实体(first-class entities)引入到过程模型中[30,49,50];(2) 将活动的输入和输出变量作为活动的标签附加到活动上[15,24].在第 1 类方法中,变量以及对变量的操作均被显式地刻画,这有助于过程建模人员发现数据流错误;这类方法的缺点在于,当使用它们对大规模系统进行建模时,相应的过程模型可能会变得非常复杂;第 2 类方法虽然不如第 1 类方法直观,但用第 2 类方法制定的过程模型较为简单.当过程模型需要演化时,需要保证目标模式控制流和数据流的正确性.下面,我们利用图 3 所示的技术框架对相关工作进行介绍.

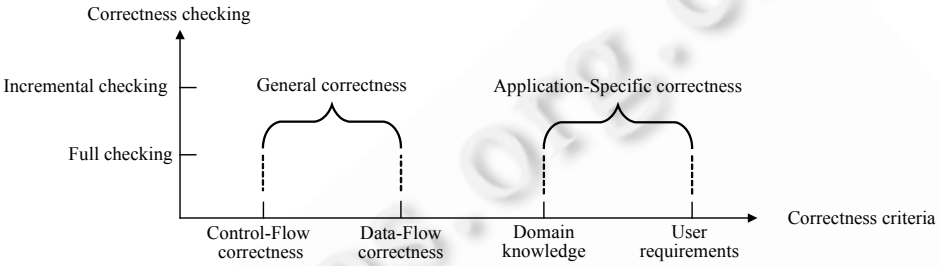


Fig.3 Technical framework for ensuring the correctness of the target schema

图 3 保证目标模式正确性的技术框架

在工作流和业务过程管理领域,工作流网的合理性从控制流的角度定义了过程模型的正确性[23].合理性的定义包括以下 3 方面的内容:(1) 弱终止性(无死锁和活锁);(2) 恰当终止性(无悬而未决的状态);(3) 无死活动(任何活动都有机会发生).需要说明的是,过程模型合理性定义中的弱终止性可以保证过程模型是无死锁和活锁的,因此若某一过程模型存在死锁或活锁,那么它肯定不满足合理性.我们可以验证图 4(a)所示的过程模型是合理的,因此它不存在死锁和活锁;而如图 4(b)所示的过程模型则不满足合理性,因为:(1) 如果活动 t_3 早于活动 t_2 触发,过程实例的执行会进入一个死锁状态(只有库所 s_5 中含有一个拖肯而库所 s_4 中不含拖肯);(2) 如果活动 t_5 触发,尽管活动 t_6 可以触发,但却产生了活锁,因为活动 t_2 和 t_4 构成了一个无法跳出的死循环.因此,若图 4(a)和图 4(b)分别为某一过程模型的源模式和目标模式,那么目标模式将无法保持源模式的控制流正确性.为进一步刻画工作流网的数据流,过程模型 WFD-net[24]在工作流网的基础上将每个活动读、写以及销毁的变量通过标签函数附加到相应的活动上.WFD-net 的控制流正确性定义与工作流网一致;虽然 WFD-net 没有从正面给出数据流的正确性准则,但却指出了过程模型需要避免的几类数据流错误.工作流网支持一些可以保持过程模型控制流正确性的增殖式修改操作:删除活动[45]、添加活动、替换活动、调整活动间的执行顺序等[32],因此当过程模型需要演化时,可以利用这些增殖式的修改操作来构建正确的目标模式.下一步的工作可以探讨,是否可以将这些修改操作扩展到也能够保持过程模型的数据流正确性.

德国乌尔姆大学的 ADEPT[30,49,50]项目组定义了一种有向图模型(称为 WSM nets),用于过程模型建模.WSM Nets 不仅描述了活动、活动之间的控制依赖,还显式地刻画了变量(数据)以及活动对变量的读写操作.这种方法将活动以及数据均作为过程建模的一阶实体,从而将过程模型的控制流和数据流(数据读写)刻画在一个统一的模型中.WSM Nets 的控制流正确性主要包括两点:(1) 弱终止性;(2) 无死活动.该定义与工作流网的合理性定义类似.WSM Nets 的数据流正确性需要满足以下两条规则:(1) 任何活动的输入变量必须在使用之前被定义过;(2) 并发执行的两个活动不能写同一个变量(除非这两个活动通过同步弧进行同步).此外,ADEPT 项目从控制流变更的角度提出了若干种修改模式[46,47],这些修改操作的正确性可以通过前置条件(pre-condition)和后

© 中国科学院软件研究所 http://www.jos.org.cn

置条件(post-condition)予以保证.

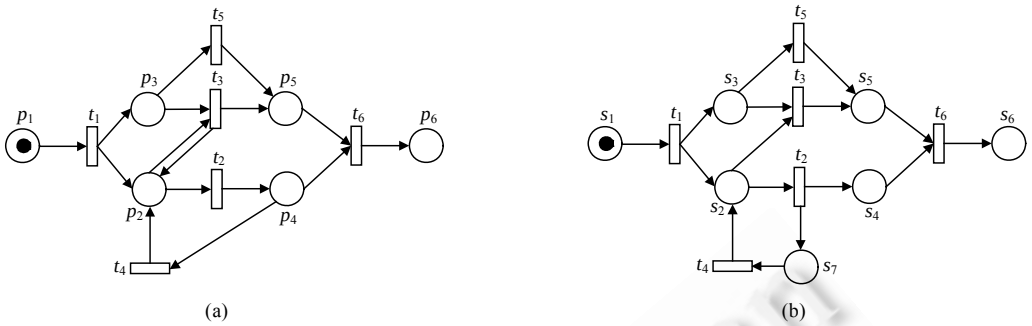


Fig.4 Process models without and with control flow errors[51]

图 4 不存在/存在控制流错误的过程模型[51]

WIDE[15]项目提出的过程模型以及过程模型的正确性准则均与 ADEPT 类似,虽然 WIDE 也提出了一个完备的修改操作集来帮助获得正确的目标模式,然而该方法中的过程模型的正确性准则不够严格.例如,它的过程模型中可能存在死活动.此外,还有的工作通过引入一些模式不变量(schema invariant)来定义过程模型的正确性,并讨论了能够保持模式不变量的修改操作,从而利用这些修改操作可以获得正确的目标模式[48].

通过上述相关工作的讨论不难发现,过程模型的正确性准则决定了相应过程模型所能支持的增殖式的修改操作.这启发我们应该根据实际应用的需求,明确过程模型的正确性准则,然后再来考虑可以保持过程模型的正确性的修改操作.例如,我们的前期工作[52]根据服务组合领域的特点,首先给出了服务组合过程模型的数据流正确性定义,然后提出了一系列可以保持服务组合数据流正确性的演化操作准则.

2.2 目标模式与源模式的一致性

在开放环境下,跨组织的过程协作日益频繁.例如,为实现一个业务目标,某一过程会参与到一些全局的过程编排中.为保证单个过程发生演化时不会影响到与它交互的伙伴过程,不但需要保证目标模式的正确性,还必须保证该演化过程的目标模式与源模式之间的一致性.即用目标模式取代源模式后,能够保持演化过程所参与过程编排的正确性[25−27].因此,目标模式与源模式之间的一致性判断依赖于过程编排的正确性准则[23].过程编排层面的正确性准则与单个过程的正确性准则类似,可以从通用正确性和特定应用正确性两个层面进行考虑.然而,由于参与过程编排的过程可能具有无穷的状态并且过程间通信的异步性可能会使得通信信道中有无穷多个消息,从而导致用时序逻辑(temporal logic)刻画的过程编排的正确性准则不可判定[53,54].为此,现有工作大多考虑状态有限的过程编排,并主要从控制流的角度来定义过程编排的正确性.

在过程编排的通用正确性方面,现有工作主要还是从控制流的角度来定义的,例如不存在没有指定的接收(no unspecified reception)[55]、无死锁[27,56]、无死锁外加有限通信(limited communication)[57]、弱终止性(即无死锁和活锁)[58]等.需要说明的是,上述过程编排的控制流正确性定义也可以将数据流正确性考虑在内.例如,若某一活动的某一输入没有定义,那么该活动就无法执行,进而会引发死活动甚至死锁等控制流错误.特定应用正确性需要考虑过程编排中活动的语义信息,结合领域规则和用户需求而制定.例如,在跨组织工作流和服务组合领域,全局过程协作的原子性常用来作为过程编排的正确性准则[18].

因为直接验证过程编排的正确性较为复杂,并且伙伴服务有时并不愿公开其私有过程,所以当某一过程发生演化时,我们需要一种局部的方法(目标模式与源模式之间的一致性)来保证局部的过程演化不会影响全局过程编排的正确性[25−27].若过程 P_1 和过程 P_2 的组合满足某一过程编排的正确性准则,那么 P_1 和 P_2 称为相容的(compatible),与过程 P_1 相容的所有过程构成 P_1 的相容伙伴过程集合.若过程 P_1 的任一相容伙伴过程也是过程 P_2 的一个相容伙伴过程,那么过程 P_2 与过程 P_1 满足一致性(过程 P_1 可以被过程 P_2 替换).一致性的定义,保证了符合相应条件的过程替换不会影响到全局过程协作的正确性.此外,通过一致性的定义我们不难发现,过程间的一致性其实是一种前序关系(preorder)[56].前序关系是一种满足自反性、传递性的二元关系,因为前序关系对对

© 中国科学院软件研究所 http://www.jos.org.cn

称性没有要求,所以等价关系和偏序关系都是前序关系.

由于与某一过程 P_1 相容的过程可能是无穷的,直接按照定义来判断过程的一致性会面临一些挑战.因此,前期的研究大多关注于是否可以使用经典进程代数理论中的进程等价关系如路径等价(trace equivalence)、互模拟(bi-simulation)等来作为一致性准则[26].强互模拟(strong bi-simulation)是过程间最强的一种等价关系,不管过程编排的控制流正确性准则如何定义,满足强互模拟关系的过程替换都可以保证过程编排的正确性,因此,强互模拟关系可以用来作为过程间的一致性准则.然而,强互模拟对于过程演化来说过于严格,因此业界致力于研究尽可能弱化的过程一致性准则,有如下两条途径:(1) 在过程间的等价关系中寻找.一些弱化的互模拟关系如弱互模拟(weak bi-simulation)等可用于定义过程间的一致性.由于过程的内部非确定性选择(internal non-deterministic choice)可能会导致过程通信的死锁,因此路径等价不适合用来定义过程间一致性[59];(2) 在过程间的偏序关系中寻找.由于目标模式强模拟(strongly simulating)源模式并不能保证过程替换后全局过程编排的无死锁性,因此强模拟关系以及经典进程代数中的其他偏序关系均不适合用来定义过程间的一致性[26,59].在过程实现(或过程优化)等方面的研究中,业界提出了行为继承(behavior inheritance)[60]和参照(accordance)[27,56−58]等偏序关系来作为过程间的一致性准则.需要说明的是,这些偏序关系同样适用于定义目标模式与源模式之间的一致性.

为利用过程间一致性的定义来指导过程演化,我们需要考虑如何保证目标模式与源模式的一致性.有以下两类方法:(1) 根据过程间一致性的定义验证目标模式与源模式的一致性;(2) 通过定义一些可以保持过程间一致性的增殖式修改操作(或称转换规则)来保证目标模式与源模式的一致性.第(2)类方法有助于不了解形式化方法的过程建模人员高效地设计符合要求的目标模式.图 5 对上述技术路线进行了总结,其中,列在各个坐标轴中的条目仅是示意性的描述而非严格的全序关系.

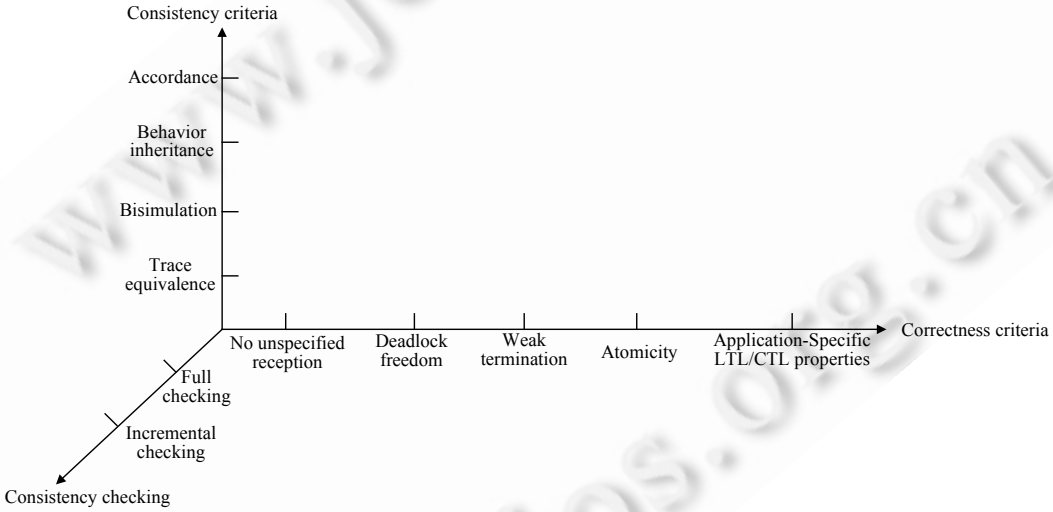


Fig.5 Technical framework for ensuring the consistency between the target and source schemas

图 5 保证目标模式与源模式一致性的技术框架

过程间的一致性问题在过程替换方面的研究中得到了广泛关注[25−27,58−61].通过过程编排,过程协作的各个参与方可以明确自己的过程协议(process protocol),从而明确自己如何与其他过程进行交互.过程协议包括了一系列通信操作以及这些操作之间的依赖关系.在某些文献中,过程协议也被称为公共视图(public view)[25]或契约(contract)[62].过程协作的参与方根据过程协议来实现自身的内部过程(例如 orchestration),这种设计方式被称为契约式设计(design by contract).在契约式设计的研究中,文献[55]将“不存在没有指定的接收”看作是最基本的过程编排正确性准则,它规定如果过程编排中的任一过程在任意状态下都不会收到它没有事先指定的消息,

© 中国科学院软件研究所 http://www.jos.org.cn

那么该过程编排就被认为是正确的.在这种正确性定义下,作者提出了相应的过程间的一致性定义(该定义没有标注在图 5 中):若某一过程实现在任一状态下所能接收的消息不少于过程协议在相应状态下所能接收的消息,同时过程实现在任一状态下所能发送的消息不多于过程协议在相应状态下所能发送的消息,那么该过程实现可以替换过程协议.同样的结论适用于目标模式对源模式的替换.然而,“不存在没有指定的接收”这种过程编排的正确性定义不能保证无死锁,因此作者进一步提出了过程编排无死锁条件下的过程一致性定义,这种一致性定义可看作是下面将要介绍的依照关系的特例.

在契约式设计中,行为继承和参照等一致性准则不受限于某一特定的过程编排正确性准则.过程的投影继承(projection inheritance)定义了一种一致性关系[25,60],它允许在父类过程的基础上有条件地添加可以抽象为哑元(silent action)的非通信活动等一组增殖式的修改操作,例如在两个活动之间添加一个新的非通信活动.该定义基于分支互模拟(branching bi-simulation)[11]的概念,因此能够保证当用一个子类过程替换一个父类过程时,过程编排的正确性依然满足[22].然而,这种继承关系仍然较为严格,例如它们不允许将彼此间没有数据依赖关系的顺序执行的活动变为并行执行.

依照(accordance)[25,27,56−58]规定,在保证过程通信活动不变的前提下,若过程 P_2 的相容伙伴过程集合包含过程 P_1 的相容伙伴过程集合,过程 P_2 可以用来替换过程 P_1.不难发现,依照关系与过程间一致性定义完全吻合,因此满足依照关系的局部过程替换能够保证全局过程编排的正确性.问题的难点在于,某一过程 P_1 的相容伙伴过程可能是无穷的,因此判断过程间的依照关系是一个具有挑战性的问题.经过业界的努力,某一过程 P 的相容伙伴过程集合可以用有限的方式(带注释的状态机)进行表示,从而使得判断两个过程的相容伙伴过程集是否存在包含关系变为可能[56,57].已经证明,投影继承关系是一种特殊的依照关系[25],因此依照关系更加弱化.除支持投影继承所支持的修改操作外,它允许更多的可以保持过程间一致性的修改操作.例如,它支持将两个没有数据依赖的连续执行的活动变为并行执行,也允许交换它们的执行次序.基于依照关系,文献[61]讨论了 BPEL 流程之间的一致性问题,并提出了一系列能够保持 BPEL 流程一致性的 BPEL 语法层面的增殖式转换规则.此外,文献[59]所定义的“模拟”关系与依照关系类似.

在特定应用正确性方面,为保证过程协作的原子性,文献[63]给出了过程模型修改操作所应遵守的准则(guideline).若某一参与过程依照这些准则进行演化,那么该过程的演化不会破坏全局过程编排的原子性.

上述工作大多从控制流的角度考虑过程模型演化前后的一致性,而较少将过程模型的数据流或数据依赖因素考虑在内.下一步的研究问题可以关注过程演化时的控制依赖(control dependence)一致性和数据依赖(data dependence)一致性.

3 过程实例可迁移性准则的设定

当过程模型演化时,目标模式的正确性以及目标模式与源模式的一致性是一种静态正确性,它们可以保证新创建的过程实例的正确性,即从目标模式的初始状态开始执行的过程实例的正确性.然而,当过程模型演化时,还应保证那些在源模式下创建并且尚未执行完毕的过程实例的正确性,即要求迁移的过程实例必须能够遵守目标模式正确性所规定的性质.过程实例迁移的正确性是一种动态正确性.

为了让过程实例的执行尽快地享受到目标模式所带来的便利,当过程模型演化时,过程引擎应立即暂停过程实例的执行,并尽可能地让过程实例在当前状态下进行迁移.然而,待迁移的过程实例能否遵守目标模式正确性所规定的性质,与过程实例的当前状态有关.若过程实例执行得太快以至于它的执行历史和当前状态不能匹配目标模式的定义,那么过程实例从当前状态迁移就可能无法保证迁移的正确性.存在以下两种可能的补救措施[10,19,20]:(1) 状态回滚(state rollback),利用活动补偿等措施使得过程实例的执行回滚到先前的一个状态,该状态可以保证过程实例的正确迁移;(2) 延迟迁移(delayed migration),这种补救措施需要保证存在着一个后续状态使得过程实例能够正确迁移.例如,当过程实例的执行处于一个循环中时,当过程实例执行下一次循环的时候可能自动地回退到一个能够保证过程实例迁移正确性的状态.若通过以上机制仍不能保证过程实例迁移的正确性,那么相应过程实例不能迁移到目标模式下.

© 中国科学院软件研究所 http://www.jos.org.cn

当寻找到一个可迁移的状态后,过程实例的迁移需要将可迁移状态下的相关变量(源模式和目标模式下所共有的变量)的取值映射到目标模式下,同时为目标模式下所独有的变量赋予相应的缺省值[64].由于过程模型的源模式和目标模式共享一个引擎,从源模式到目标模式之间的状态转换可以在一定程度上实现自动化,必要的时候需要程序员手工控制.需要说明的是,过程实例在暂停的时间内仍有可能收到外部的消息和事件,这些消息和事件可以暂时保存在专门的队列里,等过程实例恢复执行后再予以处理[64].

综上,图 6 总结出过程实例迁移需要考虑的 4 方面关键问题.据此,表 1 对过程实例迁移的代表性工作进行了归纳比较.其中,过程实例迁移的正确性准则(见第 3.1 节)以及相应的可迁移性准则(见第 3.2 节)对保证过程实例迁移的正确性具有至关重要的作用.因此,下面主要针对这两方面的研究现状进行综述.

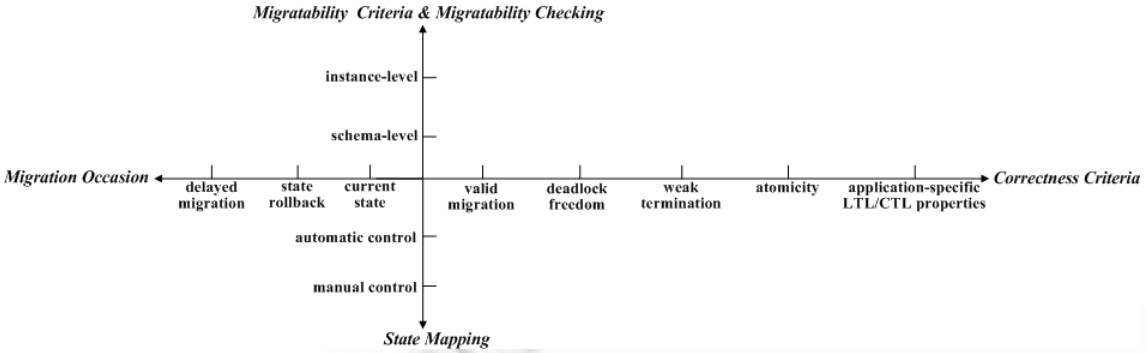


Fig.6 Technical framework for process instance migration

图 6 过程实例迁移的技术框架

Table 1 Comparison of related work on process instance migration

表 1 过程实例迁移相关工作的比较

Representive work	Correctness criteria	Migratability criteria & checking	Migration occasion	State mapping
Flow nets[28,67]	Deadlock freedom	Schema level	Current state	Manual control
WF-Nets[11,31,51]	Validity (including weak termination, duplication and skipping of tasks)	Schema level	Current state	Automatic control
WIDE[19]	Legality (similar to weak termination)	Instance level	Current state & state rollback	Automatic control
ADEPT[12,16,30,43,44,49,50]	Soundness (similar to weak termination) and application-specific LTL properties	Instance level	Current state & state rollback & delayed migration	Automatic control
eFlow[64]	Behavioral consistency (including atomicity)	Instance level	Current state & state rollback	Automatic control
ServiceMosaic[21]	Compatibility (similar to weak termination)	Instance level	Current state	Automatic control
Mia[68]	Deadlock freedom	Schema level	Current state & delayed migration	Automatic control
Dilepis[15,52,71]	Valid migration (considering both control flow and data flow corrcetness)	Instance level	Current state & delayed migration	Automatic control
LiveMig[32]	Valid migration (considering control flow corrcetness)	Schema level	Current state & delayed migration	Automatic control
IBM WID and WPS extension[66]	Valid migration (considering both control flow and data flow corrcetness)	Schema level	Current state & delayed migration	Automatic control
RAPIDware[65]	Application-specifi LTL properties	Instance level	Current state & rollback	Automatic control

© 中国科学院软件研究所 http://www.jos.org.cn

3.1 过程实例迁移的正确性准则

本文第 1 节已经说明了过程实例迁移的必要性,然而并非所有的过程实例都是可迁移的.为判断过程实例的可迁移性,应该明确指定过程实例迁移的正确性准则.尽管不同的研究者分别提出了不同的过程实例迁移的正确性准则,但这些正确性准则往往是从“避免出错”的角度来定义的[10,11,15,28].根据错误类型的不同,现有的过程实例迁移的正确性准则也可分为两类:(1) 通用正确性准则;(2) 特定应用正确性准则.不难发现,这与过程模型的正确性分类类似.

过程实例迁移的通用正确性准则不涉及过程模型中各个活动的具体语义,它的出发点在于保证过程实例在过程模式切换的情况下依然能够顺利执行并达到预定的终止状态.导致过程实例在迁移后不能达到终止状态的动态错误主要有两类:(1) 控制流错误:过程实例在迁移执行后进入死锁或活锁状态;(2) 数据流错误:过程实例迁移之后当执行某一活动时,该活动的输入没有事先定义或被不合理地定义.上述两类错误通常被称为动态演化错误[11,28],其发生的原因是:(1) 目标模型存在控制流和数据流错误,这种错误情形可以通过目标模式的正确性予以避免;(2) 模式演化过于激进,导致待迁移过程实例的已执行部分不符合目标模式的定义.例如,过程实例已经执行了源模式下的活动 a_1,但在目标模式下活动 a_1 的输入发生了改变.在这种情形下,如果把过程实例迁移到目标模式下很可能会引发数据流错误,从而导致过程实例无法正常运行并使过程实例不能达到预定的终止状态.为避免把不该迁移的过程实例迁移到目标模式下,可以将“过程实例迁移不会引入动态演化错误”作为一种通用过程实例迁移的正确性准则.符合该准则的过程实例可以迁移到目标模式,不符合该准则的过程实例应该在源模式下继续执行.

在一些特定的应用领域中,过程模型以及过程实例还需满足一定的领域知识,这些领域知识代表了特定领域的业务规则或用户的个性化需求.当过程模型发生演化时,不仅需要保证目标模式不会违反领域知识,还应判断过程实例的迁移是否会违反这些领域知识.因此在有领域知识的情况下,一个过程实例能够迁移不仅要满足通用正确性准则,还要满足特定应用正确性准则[43,44].例如,过程的原子性以及隔离性可以作为一类特殊的特定应用正确性准则[63,64].在软件自适应领域,文献[65]将这两类准则统一为用时序逻辑描述的系统规约,某一进程能够从源程序迁移到目标程序的前提条件是,该进程的迁移不会违背系统所定义的规约.

在如图 6 所示的过程实例迁移正确性准则中,过程实例迁移的有效性是最为基本的准则.过程实例迁移的有效性要求在目标模式下存在一个有效的迁移状态,在该状态恢复过程实例的执行能够保证过程实例可以在有限时间内到达目标模式下的一个可达状态.只有当过程实例的迁移满足有效性时,检验过程实例的其他正确性准则才有意义.然而在一般情形下,过程实例迁移的有效性是一个不可判定问题[29].为此,研究者分别提出了一些能够保证过程实例迁移有效性的充分条件作为过程实例的可迁移性准则.

3.2 过程实例的可迁移性准则

由于现有的可迁移性准则仅是过程实例有效迁移的充分条件而非必要条件[15],所以不满足某一可迁移性准则的过程实例也有可能保证过程实例迁移的有效性.我们认为,过程实例的可迁移性准则的设定是过程实例迁移正确性(具体是有效性)和实例迁移检验高效性的权衡考虑,实例可迁移性准则在保证过程实例迁移正确性的同时,也兼顾了过程实例迁移的高效性.需要说明的是,若过程实例迁移是有效的,那么根据有效性的定义不难发现,过程实例的迁移也不会引入死锁或活锁等错误.因此,当用可迁移性检验准则确定了过程实例迁移的有效性时,可以在此基础上进一步验证其他正确性准则(例如一些用时序逻辑描述的属性)是否成立.下面,我们具体介绍过程实例的可迁移性准则以及它们如何保证目标状态的有效性.

现有的过程实例可迁移性准则可分为两类:(1) 模式级的可迁移性准则;(2) 实例级的可迁移性准则.模式级的可迁移性准则直接在源模式和目标模式的状态之间建立映射关系,而无须考虑待迁移过程实例的历史执行信息.因此,该类准则又称为结构准则[30].与此相反,实例级的可迁移性准则是根据过程实例的历史执行信息(trace)来判断过程实例的可迁移性.因此,该类方法又称为行为准则[30].无论是模式级准则还是实例级准则,都必须满足以下两个条件:(1) 过程实例可迁移性准则必须保证过程实例迁移的有效性;(2) 过程实例的可迁移性准

© 中国科学院软件研究所 http://www.jos.org.cn

则不能过于严格.

3.2.1 模式级的可迁移性准则

模式级可迁移性准则的代表性工作是 Aalst 提出的过程继承技术[11],过程之间的继承关系是在阻塞(blocking)和隐蔽(hiding)两个算子的帮助下通过分支互模拟关系来定义的.若目标模式 S'和源模式 S 之间存在着继承关系,则可以在 S 和 S'的状态之间建立起一种部分映射(partial mapping).通过状态映射即可获得待迁移过程实例的有效目标状态,从而运行在源模式 S 下的过程实例可以迁移到目标模式 S'下.需要说明的是,基于过程继承技术的过程实例迁移只涉及控制流状态映射而没有考虑数据流的状态映射.

在多数情况下,过程实例的有效目标状态可以通过幂等映射(idempotent mapping)直接获得.例如,图 7(a)和图 7(b)是用 Petri 网描述的某一过程模型的源模式 S 和目标模式 S',S'在 S 的库所 p_2 和 p_4 之间增加了一个选择分支(活动 t_4 和 t_5 构成的分支).如果将此分支阻塞,则 S'与 S 符合分支互模拟关系.因此,S'与 S 满足继承关系,即 S'是 S 的子类.因此,运行在源模式 S 下的过程实例可以迁移到目标模式 S'下,且通过幂等映射即可求出相应的有效目标状态.图 7(a)和图 7(b)刻画了处于 S 的状态$(S,[p_3])$下的过程实例可以迁移到 S'的$(S',[s_3])$状态下,并且不难发现目标状态$(S',[s_3])$是有效的.同样地,若图 7(b)为源模式 S,图 7(a)为目标模式 S',除状态$(S,[s_5])$外,S 下其余状态都可以通过幂等映射得到相应的目标状态.因此,这种过程继承技术的方法有时只能建立部分映射.此外,在某些情况下,过程实例的目标状态需要在幂等映射的基础上作适当调整.例如,图 7(c)和图 7(d)是用 Petri 网描述的某一过程模型的源模式 S 和目标模式 S',S'在 S 的基础上增加了一个与活动 t_2 并行执行的活动 t_4.如果对活动 t_4 施加隐蔽操作,则 S'与 S 符合分支互模拟关系.在建立 S 与 S'的状态映射时,需要在幂等映射的基础上作适当调整.如图 7(c)和图 7(d)所示,如果将状态$(S,[p_3])$映射到$(S',[s_3])$,$(S',[s_3])$不是一个有效目标状态(该状态在目标模式下不可达),过程实例迁移后会发生死锁.此时,需要在库所 s_5 中添加一个拖肯.当然,也可在库所 s_6 中添加一个拖肯,但这种方式将跳过变迁 t_4,当变迁 t_3 数据依赖于 t_4 时会引发数据流错误.

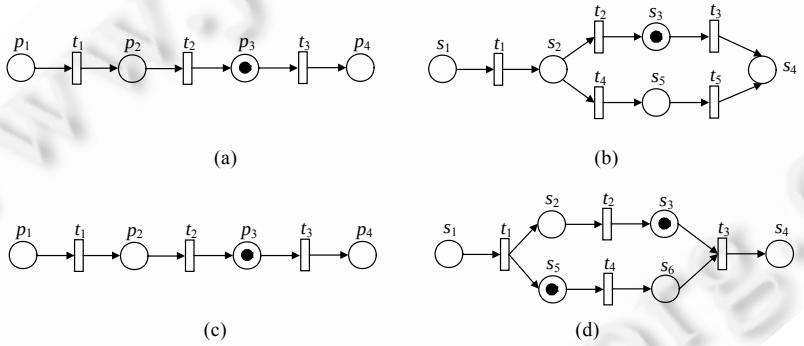


Fig.7 State mapping between source and target schemas with inheritance relation

图 7 过程继承条件下源模式与目标模式之间的状态映射

过程继承技术虽然可以保证目标状态的有效性,但该方法能够适用的前提是源模式和目标模式之间存在继承关系,当这种关系不成立的情况下,过程继承技术不再适用.为弥补这一不足,Aalst 提出了一种不限定过程模型演化类型的过程实例迁移方法[31].该方法通过比较变化前后的过程模型,找出过程模型中所有发生了变化的区域.只有当过程实例不在变化区域里执行的时候,才允许该过程实例的迁移.若考虑数据流因素,还应利用变化活动之间的数据依赖关系来帮助划分变化区域[66].虽然这类方法可以避免动态演化错误,但可能使某些过程实例不能在第一时间进行迁移,从而使得这些过程实例不能立即享受到过程演化所带来的好处.

此外,Ellis 通过引入额外的跳转变迁(称为 flow jumper)建立起源模式和目标模式的状态映射[67].这种方法可以保证目标状态的有效性.然而,该方法需要过程模型建模者手工设定跳转变迁,并且网的规模也因跳转变迁的引入而变得较为复杂.因此,该方法并没有得到普及.需要说明的是,现有的模式级可迁移性准则大多只关注于实例迁移的控制流而忽略了数据流[11,67,68].若想进一步了解模式级可迁移性准则的早期研究(2004 年之前),

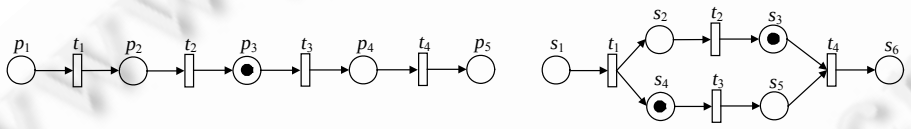
© 中国科学院软件研究所 http://www.jos.org.cn

读者可见文献[10].

3.2.2 实例级的可迁移性准则

实例级的可迁移性准则根据过程实例的执行历史信息(已执行的活动序列)来判断相应实例的可迁移性.这类方法的优点在于它不限定过程模型的演化类型,从而可以支持更多的过程模型修改操作.最常用的一种实例级的可迁移性准则是轨迹重现(trace replaying)技术[19].这里的轨迹是抽象的,它是实际可操作过程的执行轨迹在过程模型抽象层次上的表现[69].轨迹重现技术的基本思想是,检查待迁移过程实例的已执行活动序列σ是否可以在目标模式下重现.如果σ可以在目标模式下重现,那么该过程实例可迁移;反之,该过程实例只能继续运行在源模式下.注意:在检验σ是否可重现时,不仅要求σ中的每个活动能够在目标模式下重现,而且要求每个活动的输入和输出变量也要在目标模式下重现.

轨迹重现技术要用到过程实例的已执行活动序列信息,因此需要过程引擎能够将过程实例的已执行活动按照结束时间的先后顺序记录在内存或外存中.当过程模型由源模式 S 演化到目标模式 S'时,若 S 下某一过程实例的已执行活动序列σ可以在 S'下按照既定顺序重现,即有$(S', M'_0)[\sigma\rangle(S',M')$成立,那么该过程实例可以迁移到目标模式,且目标状态为(S',M').由于(S',M')是 S'的一个可达状态,因此相应实例迁移是有效的.因此,轨迹重现技术满足可迁移性准则的第 1 个条件:保证过程实例迁移的有效性.同时,相对于过程继承技术,轨迹重现技术的粒度更低,它可以对单个过程实例的可迁移性进行分析而无须考虑模式之间的继承关系.因此,轨迹重现技术具有更好的柔性.一方面,对于图 7 中的两种情况,利用轨迹重现技术同样可以检验相关过程实例的可迁移性.另一方面,轨迹重现技术能够支持更多的过程实例迁移.例如,图 8(a)和图 8(b)分别为过程模型的源模式 S 和目标模式 S',其中活动 t_1~t_4 的输入和输出保持不变.过程继承技术不允许将 S 中的过程实例迁移到 S'中,因为 S 和 S'不存在继承关系.然而,轨迹重现技术允许将 S 中的任意过程实例迁移到 S'中,并允许将 S'中的部分过程实例迁移到 S 中.例如,S 中已执行活动序列为 t_1t_2 的过程实例可以迁移到 S'的状态$(S',[s_3,s_4])$下恢复执行,S'中已执行活动序列为 t_1t_2 的过程实例可以迁移到 S 的状态$(S,[p_3])$下恢复执行.



(a) (b)

Fig.8 Migratability checking for process instances based on the technique of trace replaying

图 8 基于轨迹重现技术的过程实例可迁移性检验

如果目标模式所作的修改只影响到过程实例未曾执行过的区域,那么按照轨迹重现技术,该过程实例是可迁移的.待迁移过程实例的已执行活动序列不能在目标模式下重现的原因在于,目标模式修改了该待迁移过程实例已执行过的区域.然而在某些情形下,即使目标模式修改了该待迁移过程实例已执行过的区域,过程实例的迁移并不会引入动态演化错误,此时,传统的轨迹重现技术显得过于保守.因此,研究者们对传统的轨迹重现技术进行了弱化[10,14,50,70,71].下面对可弱化的情形以及相应的弱化途径作简要介绍.

针对循环修改操作的弱化[50,70]:当过程模型存在着循环(loop),目标模式可能对源模式中循环的内部结构进行了修改.若源模式下,某一过程实例在相应的循环体内重复执行的次数超过一次,如果采用轨迹重现技术作为准则,该过程实例的已执行活动序列很可能无法在目标模式下重现.然而在通常情况下,只有循环的最后一次执行才能对过程实例的后续执行产生影响.为此,在进行轨迹重现时,可以忽略循环的历史执行(最后一次执行除外)所记录下来的活动条目[50].由于这种处理方式去除了待迁移过程实例的已执行活动序列中与可迁移性检验不相关的信息,轨迹重现技术得以弱化.需要说明的是,这种弱化同样适用于嵌套循环.

针对删除活动操作的弱化[70,71]:目标模式可能删除源模式中的一些活动,如果某些删除的活动已经记录在该过程实例的已执行活动序列中,那么按照传统的轨迹重现技术,相应过程实例是不可迁移的.然而,重现在目

© 中国科学院软件研究所 http://www.jos.org.cn

标模式下已经删除的活动并没有意义.因此在进行轨迹重现检验时,可以忽略目标模式下已删除的活动.

针对交换活动执行顺序操作的弱化[15]:目标模式可能颠倒了源模式下两个活动执行的先后顺序,若这两个活动已经记录在某一待迁移过程实例的已执行活动序列中,按照传统的轨迹重现技术,则该过程实例不可迁移.然而,如果这两个活动之间不存在数据依赖关系,它们的执行顺序并不关键.在这种情形下,也应允许该过程实例的迁移.

针对增加活动操作的弱化[15]:目标模式可能在源模式的两个活动 a 和 b 之间添加一个或多个新的活动,若 a 和 b 已经记录在某一过程实例的已执行活动序列中,那么按照传统的轨迹重现技术,该过程实例不可迁移.然而,如果 a 和 b 之间添加的活动并没有影响到活动 b 的变量使用情况(但可能影响到 b 的后续活动的变量使用),则活动 a 和 b 的执行结果可以在目标模式下重用.此时,可以将过程实例迁移到目标模式下活动 a 执行后所得的状态下继续执行.当执行到活动 b 时,可以重用在 b 在源模式下的执行结果而无须重新执行.这种情形下,过程实例也被认为可以在目标模式下重现(称为分段重现).

轨迹重现技术的放松途径(1)~途径(4)分别从 4 个不同的侧面对传统的轨迹重现技术进行了放松,将它们进行组合可以得到更多的放松途径.同时,放松途径之间的松弛程度的比较构成一种偏序关系,因此,各种放松途径构成的集合(包括传统的轨迹重现技术)关于偏序“松弛程度”做成一个格(lattice).根据格的性质,我们不难定义轨迹重现技术体系下最为松弛的可迁移性判定标准.此外,探讨其他有别于轨迹迁移技术的其他可迁移性准则也是未来应当关注的研究问题.

4 过程实例可迁移性的高效检验

为使用户感觉不到服务的中断,过程实例的迁移必须高效.过程实例迁移的高效性主要受以下 3 方面因素的影响:(1) 目标模式的快速设计;(2) 采用具有切实可操作性的过程实例可迁移性准则;(3) 过程实例可迁移性的高效检验(包括目标状态的确定).其中,前两个方面的内容已经在本文的第 2 节和第 3 节讨论.本节关注于第(3)个方面的相关研究.具有切实可操作性的过程实例可迁移性检验准则只给出了过程实例可迁移的定义,在实际应用中还要考虑如何高效地判断这些可迁移性检验准则是否成立,若成立,还需要高效地确定过程实例迁移的有效目标状态.

在模式级的可迁移性准则中,过程继承技术是自动化程度较高的:(1) 在过程实例可迁移性检验方面:过程继承技术可以利用增殖式的过程转换规则(transformation rule),高效地确定过程模型的目标模式和源模式之间是否存在继承关系.如果目标模式和源模式存在继承关系,那么在源模式下运行的过程实例可以迁移到目标模式下;(2) 在目标状态的确定方面:过程继承技术可以通过源模式和目标模式之间的幂等映射(或在幂等映射基础上的调整),高效地确定可迁移过程实例的有效目标状态.过程继承技术的局限性在于它仅支持目标模式与源模式之间满足继承性的情形.没有这一局限性的模式级的可迁移性准则要么不能在模式演化的第一时间进行实例的迁移[31,66],要么自动化程度不高[67],都不利于过程实例可迁移性的快速检验.

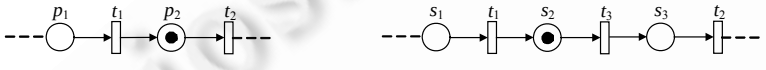
轨迹重现技术(包括轨迹重现技术的各种弱化途径)是应用最为广泛的实例级的可迁移性准则.由于过程实例的已执行活动序列可能存在很多条目,在待迁移的过程实例较多的情况下,如果直接利用轨迹重现的定义来检验相应过程实例的可迁移性并不高效.例如在医疗和金融等领域,当过程模型升级时,待迁移的过程实例数目往往以“万”为数量级[72].过程实例的已执行活动序列的每一个条目大约需要 40 个字节的存储空间(活动标识符、活动的输入和输出、时间戳等)[72],假设某一过程模型含有 50 个活动,待迁移的过程实例为 2 万个,这些过程实例的已执行活动序列的平均长度为 25,那么记录这些待迁移实例的已执行活动序列信息总共需要 20M 存储空间.因此,如果直接按照轨迹重现的定义进行过程实例的可迁移性检验可能需要较长的时间,进而影响到服务质量.此外,PAIS 系统可能会将过程实例的历史执行信息记录在硬盘上,从硬盘读入这些数据到内存后,还需要对这些数据进行适当的预处理,这会进一步影响过程实例迁移的效率.因此,过程实例可迁移性的高效检验具有实际意义.为应对这一挑战,业界提出了多种不同的处理方案.下面,我们对有代表性的一些工作进行介绍.

ADEPT 项目提出一种高效的过程实例可迁移性检验方法[50,72],该方法针对过程模型的每一类修改操作(例

© 中国科学院软件研究所 http://www.jos.org.cn

如添加、删除活动等)给出了相应的过程实例可迁移检验方法,该方法仅仅用到过程实例在源模式下的状态信息以及该过程实例的部分轨迹信息.对于添加活动,如图 9(b)所示的过程模型的目标模式 S'在如图 9(a)所示的过程模型的源模式 S 的活动 t_1 和活动 t_2 之间添加了一个新的活动 t_3,使得活动 t_2 成为活动 t_3 的直接后继活动.在源模式下,若某一待迁移过程实例尚未执行活动 t_2,那么该过程实例是可迁移的且目标状态为活动 t_1 执行完毕后所得的状态$(S',[s_2])$,并且不难发现目标状态$(S',[s_2])$是有效的.该过程实例可以迁移的原因在于活动 t_2 尚未记录在该过程实例的已执行活动序列中.同样地,若图 9(b)所示的为过程模型的源模式 S 而图 9(a)所示的为过程模型的目标模式 S',其中,S'通过在 S 中删除活动 t_3 获得.在源模式下,若某一待迁移过程实例尚未执行活动 t_3,那么该过程实例是可迁移的且有效的目标状态为活动 t_1 执行完毕后所得的状态$(S',[p_2])$.同样地,该过程实例可以迁移的原因在于活动 t_3 尚未记录在过程实例的已执行活动序列中.活动的移动可以通过删除活动和添加活动这两类操作予以实现,因此该方法也能处理活动的执行顺序发生了改变的情形.由于 ADEPT 将数据(变量)以及对数据的操作作为一阶实体,ADEPT 还给出了单独的数据流修改操作情形下过程实例可迁移性的检验方法.

ADEPT 的检验方法在不需要重现过程实例的已执行活动序列的前提下也能够判断过程实例的可迁移性(证明见文献[50,72]),并且目标状态的确定也是自动化的,因此该方法具有较高的效率.然而,这种检验方法只允许目标模式对待迁移过程实例的未来进行修改,对于目标模式修改了待迁移过程实例已执行过的区域的情形支持有限(仅支持那些在未执行的选择分支上的修改操作).此外,在某些情形下,源模式到目标模式的修改操作并不能直接得到,而必须对源模式与目标模式进行比较方可获得.因此,ADEPT 的这种检验方法固然高效,但应用范围有限.



(a) (b)

Fig.9 Efficiently checking the migratability of process instances by using the method of ADEPT

图 9 利用 ADEPT 中的方法高效检验过程实例的可迁移性

为弥补 ADEPT 方法的不足,文献[73]尝试给出了相应的解决方案.首先,为保证过程实例的可迁移性,过程模型的演化不能过于激进.为此,作者要求过程模型的目标模式必须与源模式相容(compatible).这里的相容性定义有两个条件:(1) 过程模型的源模式和目标模式都必须满足合理性[23];(2) 目标模式必须包含源模式的全部路径(到达终止状态的活动序列),但允许在源模式的这些路径上添加或者删除一些活动.其次,作者给出了一个求解变化区域(change region)的算法,该算法通过比较源模式和目标模式可以求解出过程模型的所有变化区域.每一个变化区域都有新旧两个版本 CR'和 CR,即源模式下某一区域 CR 在目标模式下变为 CR'.本文中,我们将 CR 和 CR'分别称为源变化区域和目标变化区域.该方法将判断源模式和目标模式的相容性问题转化为检验变化区域的相容性问题.通常情况下,变化区域的规模比过程模型的规模要小得多,因此相容性的检验可以得到简化.此外,利用变化区域还可高效地检验源模式下过程实例的可迁移性:(1) 若过程实例尚未执行到源变化区域 CR,则过程实例可以迁移到目标模式下(这和 ADEPT 中的方法类似),且目标状态可以通过幂等映射获得;(2) 若过程实例的执行已经进入到源变化区域 CR,此时只需判断该过程实例在 CR 里执行的活动序列是否可以在目标变化区域 CR'里重现即可,若可重现,目标状态也随之确定.为表达的方便,本文将这种方法称为变化区域法.

下面举例解释变化区域法.图 10(a)和图 10(b)分别给出了某一过程模型的源变化区域 CR 和目标源变化区域 CR',CR 和 CR'分别隶属于该过程模型的源模式 S 和目标模式 S'.CR 下有一条执行路径 t_1t_2, CR'下有两条执行路径 t_1t_4 和 t_3t_4.CR'的路径 t_1t_4 在 CR 的路径 t_1t_2 上删除了一个活动 t_2 的同时添加了一个新的活动 t_4,此外,CR'在 CR 的基础上增加了一条新的路径 t_3t_4.若过程模型只有这一处变化区域,不难发现 CR'和 CR 是相容的,从而 S'和 S 也是相容的.假设在源模式 S 下某一过程实例的已执行活动序列为 $\sigma=\rho t_1$,若利用轨迹重现技术,需要在目标模式 S'下重现σ的所有条目才能够检验该过程实例是否可迁移.实际上,由于过程模型只有这一处变化区域,我们只需判断该过程实例在源变化区域 CR 里执行的活动 t_1 是否能够在目标变化区域 CR'里重现即可.由于活动 t_1 可以在 CR'里重现,因此该过程实例是可迁移的且目标状态为$(S',[s_2])$.

© 中国科学院软件研究所 http://www.jos.org.cn

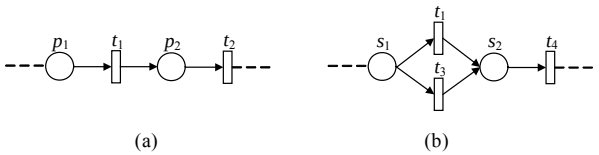


Fig.10 Efficiently checking the migratability of process instances by using the method of change region

图 10 利用变化区域法高效地检验过程实例的可迁移性

变化区域法缩小了轨迹重现的范围,因而可以高效地检验过程实例的可迁移性[59].然而变化区域法仅仅关注到控制流的正确性,暂时没有考虑到数据流的正确性.此外,这种方法限定目标模式与源模式相容,因此在过程模型演化时,它不支持两个活动的执行顺序发生改变的修改操作.

从本质上说,上述两种方法都是从避免重现待迁移过程实例的全部历史信息(已执行活动序列)来提高可迁移性检验效率的.这体现了一种"约减"的思想,而文献[71]中的方法则体现了"复用"的思想.这种复用方法的提出基于以下两点观察:(1) 尽管源模式下存在这大量待迁移的过程实例,但是可以依据这些过程实例的已执行活动序列将它们分为若干等价类.由于同一等价类下的过程实例具有相同的已执行活动序列,若分析了其中一个过程实例的可迁移性结果,则该等价类下的其余过程实例可以复用这一结果;(2) 这些等价类之间存在偏序关系,若某一等价类 C_1 中过程实例的已执行活动序列 σ_1 为另一等价类 C_2 中过程实例的已执行活动序列 σ_2 的前缀子串,那么在目标模式下重现 σ_2 的过程中也必然要重现 σ_1,因此在检验等价类 C_2 中过程实例的可迁移性的同时也检验了等价类 C_1 中过程实例的可迁移性.因此,在重现 σ_2 时若将中间结果保存下来,即可获得 σ_2 的全部前缀子串的重现结果.

图 11(a)和图 11(b)分别给出了某一过程模型的源模式 S 和目标模式 S',S'移动了 S 中活动 t_5 的位置并在活动 t_6 后添加了一个新的活动 t_7.下面通过这个简单的例子解释上述复用方法.假设当过程模型演化时,运行在源模式 S 下的过程实例有很多.我们可以根据这些过程实例的已执行活动序列信息,将这些过程实例分为 7 个等价类,这些等价类下过程实例的已执行活动序列分别为 t_1,t_1t_2,$t_1t_2t_3$,$t_1t_2t_3t_6$,t_1t_4,$t_1t_4t_5$,$t_1t_4t_5t_6$.然而不难发现,这些已执行活动序列实际上是活动序列 $t_1t_2t_3t_6$ 或 $t_1t_4t_5t_6$ 的前缀子串.因此,我们只需按照轨迹重现技术重现活动序列 $t_1t_2t_3t_6$ 和 $t_1t_4t_5t_6$,其余已执行活动序列的重现结果(相应等价类中过程实例的可迁移性以及可迁移过程实例的目标状态)也随之确定.

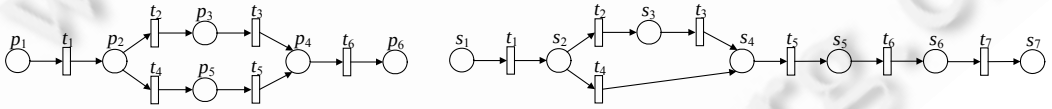


Fig.11 Efficiently checking the migratability of process instances by using the REUSE method

图 11 利用复用法高效地检验过程实例的可迁移性

利用这种复用的方法,全部待迁移过程实例的可迁移性以及目标状态可以在多项式时间内得以确定,具有较高的效率.需要说明的是,尽管文献[71]所用的轨迹重现技术没有考虑数据流因素,但这种复用的思想也同样适用于考虑数据流的轨迹重现技术[14].进一步的研究可以对复用法与变化区域法的效率进行比较.此外,若综合考虑过程迁移的控制流和数据流状态,后续工作或许可以利用程序依赖图(program dependence graph)以及程序切片(program slicing)技术获得更加直观和高效的过程实例可迁移性检验方法.

5 总结与展望

软件动态(在线)演化的相关研究分布在诸如程序设计语言[29,74−76]、软件体系结构[77−80]、软件构件[81,82]、PAIS 等多个研究领域,本文的综述侧重在 PAIS 领域.有关 PAIS 过程动态演化方面的研究一直方兴未艾,Allis 在 1995 年首次提出过程动态演化的概念距今已有 16 年的时间[28],在此期间,关于 PAIS 动态演化方面的理论和技术均得到了长足发展.本文从 PAIS 中过程动态演化(也即过程实例动态演化)问题所面临的挑战入手,从如何

© 中国科学院软件研究所 http://www.jos.org.cn

保证过程实例动态演化的正确性和高效性两个方面来组织和介绍现有研究成果,希望能为 PAIS 动态演化以及软件演化的进一步研究提供一定的参考价值.

随着 Internet 技术的快速发展,软件的规模和复杂性日益提高,过程技术也在软件系统中发挥着愈来愈重要的作用.网络环境的开放、动态和多变性,以及用户使用方式的个性化促使了网构软件[9,83,84]这一新型软件形态的出现,而过程技术作为一种快速搭建客户应用的手段同样适用于网构软件的构建.因此,下一代的过程管理技术应该更加关注过程的可演化性.我们认为,PAIS 过程演化的后续研究可以从以下几个方面进行:

(1) PAIS 过程静态和动态演化时过程模型的版本升级问题.现阶段尽管存在一些辅助的过程建模工具,PAIS 的过程模型大多是专门的业务过程建模人员或领域专家依据他们的经验制定的,在过程模型进行版本升级时也需要大量人的参与.为提高版本升级的效率,可以借助挖掘(mining)技术减少人的工作量[20,85].例如,在制定目标模式时可以直接从类似场景的过程模型变种中挖掘出一个新的过程模型作为目标模型,也可以从修改记录中挖掘出一些变化模式(change pattern)来帮助领域专家从源模式出发快速获得目标模式.此外,还可应用一些 AI 规划的方法自动或半自动地获得演化后的过程片段[86].

(2) PAIS 过程动态演化时过程实例的迁移问题.一方面,应该进一步探讨更为弱化的过程实例可迁移性准则,将那些迁移过后不会引入动态演化错误的过程实例尽可能地迁移到目标模式下继续执行;另一方面,对于那些不可迁移的过程实例(迁移过后可能会引发动态演化错误),可以采用补偿活动等手段使得过程恢复到一个可以迁移的状态,或直接在过程层或过程实例层采取针对性的过程调整措施[87],使得在调整过后,这部分过程实例可以迁移到目标模式.

(3) 过程编排层面(或分布式过程)的静态和动态演化问题.过程(或服务)编排是跨组织的过程之间进行协作的一种组织方式.由于过程编排可能没有一个统一的中心协调者,参与编排的各个过程处于各个自治的组织内部,使得过程编排的(静态和动态)演化存在诸多挑战[88,89].目前我们也正致力于这方面的工作,现阶段的工作涉及如何保证过程编排演化的可行性,如何保证参与编排的各个过程共同演化(co-evolution)的一致性.未来的工作还应考虑过程编排演化时的协商机制.

(4) PAIS 过程动态演化技术在实际中的应用.PAIS 属于实用性较强的一个领域,现如今它已广泛地渗透到商业管理、金融保险、股票证券、医疗保健等许多领域.对于那些长时间运行的 PAIS 过程,例如,贷款与还贷过程、签证申请过程、科学工作流等,它们在运行过程中存在频繁变动的可能,如何利用 PAIS 过程动态演化技术以支持上述应用的动态演化是今后的研究热点.考虑这些特定应用时,过程模型的正确性的不仅会涉及通用正确性,还应该考虑一些领域知识[43,44,65].其中主要的研究点包括:领域知识和个性化用户需求的表示、过程实例演化时领域知识的高效验证、特定应用中过程实例的可迁移性准则等.

致谢 特别感谢南京大学徐家福先生对本文内容、措辞、定义以及翻译等方面的指导;向对本文的工作给予支持和帮助的国际同行 Wolf 教授团队、Dumas 教授团队以及南京大学软件研究所陶先平老师、徐锋老师、曹春老师、黄宇老师、余萍老师、许畅老师,博士生马骏、邹阳、杨启亮等表示感谢.最后向对本文提出建设性意见的审稿专家致以诚挚的谢意.

References:

[1] DeRemer F, Kron HH. Programming-in-the-Large versus programming-in-the-small. IEEE Trans. on Software Engineering, 1976, 2(2):80−86. [doi: 10.1109/TSE.1976.233534]

[2] Singh MP, Chopra AK, Desai N, Mallya AU. Protocols for processes: Programming in the large for open systems (extended abstract). In: Proc. of the Companion to the 19th Annual ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications. New York: ACM Press, 2004. 120−123. http://portal.acm.org/ft_gateway.cfm?id=1028712&type=pdf [doi: 10.1145/1028664.1028712]

[3] Alves A, *et al*. Web services business process execution language version 2.0. OASIS Standard, 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

© 中国科学院软件研究所 http://www.jos.org.cn

[4] Ardagna D, Ghezzi C, Mirandola R. Rethinking the use of models in software architecture. In: Becker S, Plasil F, Reussner R, eds. Proc. of the 4th Int'l Conf. on Quality of Software-Architectures. LNCS 5281, Berlin, Heidelberg: Springer-Verlag, 2008. 1−27. [doi: 10.1007/978-3-540-87879-7_1]

[5] Epifani I, Ghezzi C, Mirandola R, Tamburrelli G. Model evolution by run-time parameter adaptation. In: Proc. of the 31st Int'l Conf. on Software Engineering. Vancouver: IEEE Computer Society Press, 2009. 111−121. http://portal.acm.org/citation.cfm?id=1555027 [doi: 10.1109/ICSE.2009.5070513]

[6] Dumas M, van der Aalst WMP, ter Hofstede AHM. Process-Aware Information Systems: Bridging People and Software Through Process Technology. Hoboken: John Wiley & Sons, 2005.

[7] Dumas M, van der Aalst WMP, ter Hofstede AHM, Wrote; Wang JM, Wen LJ, Trans. Process-Aware Information Systems. Beijing: Tsinghua University Press, 2009 (in Chinese).

[8] Fan YS. Fundamentals of Workflow Management Technology. Beijing: Tsinghua University Press/Springer-Verlag, 2001 (in Chinese).

[9] Lü J, Tao XP, Ma XX, Hu H, Xu F, Cao C. On agent-based software model for internetware. Science in China (E), 2005,35(12): 1233−1253 (in Chinese with English abstract).

[10] Rinderle S, Reichert M, Dadam P. Correctness criteria for dynamic changes in workflow systems—A survey. Data & Knowledge Engineering, 2004,50(1):9−34. [doi: 10.1016/j.datak.2004.01.002]

[11] van der Aalst WMP, Basten T. Inheritance of workflows: An approach to tackling problems related to change. Theoretical Computer Science, 2002,270(11):125−203. [doi: 10.1016/S0304-3975(00)00321-2]

[12] Reichert M, Rinderle S, Dadam P. On the common support of workflow type and instance changes under correctness constraints. In: Meersman R, *et al*., eds. Proc. of the OTM Confederated Int'l Conf. LNCS 2888, Berlin, Heidelberg: Springer-Verlag, 2003. 407−425. [doi: 10.1007/978-3-540-39964-3_26]

[13] Schonenberg MH, Mans RS, Russell NC, Mulyar NA, van der Aalst. Towards a taxonomy of process flexibility. In: Bellahsene Z, *et al*., eds. Proc. of the Forum at the CaiSE 2008 Conf. Berlin, Heidelberg: Springer-Verlag, 2008. 81−84.

[14] Peltz C. Web services orchestration and choreography. IEEE Computer, 2003,36(10):46−52. [doi: 10.1109/MC.2003.1236471]

[15] Song W, Ma XX, Lü J. Instance migration in dynamic evolution of Web service compositions. Chinese Journal of Computers, 2009, 32(9):1816−1831 (in Chinese with English abstract).

[16] Reichert M, Dadam P. Enabling adaptive process-aware information systems with ADEPT2. In: Handbook of Research on Business Process Modeling. Hershey: Information Science Reference, 2009. 173−203.

[17] Hagen C, Alonso G. Exception handling in workflow management systems. IEEE Trans. on Software Engineering, 2000,26(10): 943−958. [doi: 10.1109/32.879818]

[18] Ye CY, Cheung SC, Chan WK, Xu C. Atomicity analysis of service composition across organizations. IEEE Trans. on Software Engineering, 2009,35(1):2−28. [doi: 10.1109/TSE.2008.86]

[19] Casati F, Ceri S, Pernici B, Pozzi G. Workflow evolution. Data & Knowledge Engineering, 1998,24(3):211−238. [doi: 10.1016/S0169-023X(97)00033-5]

[20] Sadiq SW. Handling dynamic schema change in process models. In: Proc. of the 11th Australasian Database Conf. Washington: IEEE Computer Society Press, 2000. 120−126. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=819822&tag=1 [doi: 10.1109/ADC.2000.819822]

[21] Ryu SH, Casati F, Skogsrud H, Benatallah B, Saint-Paul R. Supporting the dynamic evolution of Web service protocols in service-oriented architectures. ACM Trans. on the Web, 2008,2(2):Article No.13. [doi: 10.1145/1346237.1346241]

[22] Kindler E. Model-Based software engineering and process-aware information systems. Trans. on Petri Nets and Other Models of Concurrency, LNCS 5460, 2009,2:27−45. [doi: 10.1007/978-3-642-00899-3_2]

[23] van der Aalst WMP. The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers, 1998, 8(1):21−66. [doi: 10.1142/S0218126698000043]

[24] Trčka N, van der Aalst WMP, Sidorova N. Data-Flow anti-patterns: Discovering data-flow errors in workflows. In: van Eck P, Gordijn J, Wieringa R, eds. Proc. of the 21st Int'l Conf. on Advanced Information Systems Engineering. LNCS 5565, Berlin, Heidelberg: Springer-Verlag, 2009. 425−439. [doi: 10.1007/978-3-642-02144-2_34]

[25] van der Aalst WMP, Lohmann N, Massuthe P, Stahl C, Wolf K. From public views to private views—Correctness-by-Design for services. In: Dumas M, Heckel R, eds. Proc. of the 4th Int'l Workshop on Web Services and Formal Methods. LNCS 4937, Berlin, Heidelberg: Springer- Verlag, 2007. 139−153. [doi: 10.1007/978-3-540-79230-7_10]

[26] Decker G, Weske M. Behavioral consistency for B2B process integration. In: Krogstie J, Opdahl AL, Sindre G, eds. Proc. of the 19th Int'l Conf. on Advanced Information Systems Engineering. LNCS 4495, Berlin, Heidelberg: Springer-Verlag, 2007. 81−95.

[27] van der Aalst WMP, Lohmann N, Massuthe P, Stahl C, Wolf K. Multiparty contracts: Agreeing and implementing inter-organizational processes. The Computer Journal, 2010,53(1):90−106. [doi: 10.1093/comjnl/bxn064]

[28] Ellis CA, Keddara K, Rozenberg G. Dynamic change within workflow systems. In: Proc. of the 19th Int'l Conf. on Organizational Computing Systems. New York: ACM Press, 1995. 10−21. http://portal.acm.org/citation.cfm?id=224021 [doi: 10.1145/224019.224021]

[29] Gupta D, Jalote P, Barua G. A formal framework for on-line software version change. IEEE Trans. on Software Engineering, 1996, 22(2):120−131. [doi: 10.1109/32.485222]

© 中国科学院软件研究所 http://www.jos.org.cn

[30] Reichert M, Rinderle-Ma S, Dadam P. Flexibility in process-aware information systems. Trans. on Petri Nets and Other Models of Concurrency, LNCS 5460, 2009,2:115−135.

[31] van der Aalst WMP. Exterminating the dynamic change bug: A concrete approach to support workflow change. Information Systems Frontiers, 2001,3(3):297−317. [doi: 10.1023/A:1011409408711]

[32] Zeng J, Huai JP, Sun HL, Deng T, Li X. LiveMig: An approach to live instance migration in composite service evolution. In: Proc. of the 7th IEEE Int'l Conf. on Web Service. Washington: IEEE Computer Society Press, 2009. 679−686. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5175884 [doi: 10.1109/ICWS.2009.96]

[33] van der Aalst WMP. Formalization and verification of event-driven process chains. Information and Software Technology, 1999, 41(10):639−650. [doi: 10.1016/S0950-5849(99)00016-6]

[34] Li JQ, Fan YS, Zhou MC. Timing constraint workflow nets for workflow analysis. IEEE Trans. on Systems, Man, and Cybernetics, 2003,33(2):179−193. [doi: 10.1109/TSMCA.2003.811771]

[35] Ouyang C, Verbeek E, van der Aalst WMP, Breutel S, Dumas M, ter Hofstede AHM. Formal semantics and analysis of control flow in WS-BPEL. Science of Computer Programming, 2007,67(2-3):162−198. [doi: 10.1016/j.scico.2007.03.002]

[36] Lohmann N, Kleine J. Fully-Automatic translation of open workflow net models into simple abstract BPEL processes. In: Kühne T, Reisig W, Steimann F, eds. Proc. of Modellierung. LNI P-127, Berlin, Heidelberg: Springer-Verlag, 2008. 57−72.

[37] Ouyang C, Dumas M, van der Aalst WMP, ter Hofstede AHM, Mendling J. From business process models to process-oriented software systems. ACM Trans. on Software Engineering and Methodology, 2009,19(1):Article No.2. [doi: 10.1145/1555392.1555395]

[38] Murata T. Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 1989,77(4):541−580. [doi: 10.1109/5.24143]

[39] van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B, Barros AP. Workflow patterns. Distributed and Parallel Databases, 2003, 14(1):5−51. [doi: 10.1023/A:1022883727209]

[40] Song W, Dou WC, Liu XP. Timing constraint Petri nets and their schedulability analysis and verification. Journal of Software, 2007,18(1):11−21 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/11.htm [doi: 10.1360/jos180011]

[41] Song W, Ma XX, Ye CY, Dou WC, Lü J. Timed modeling and verification of BPEL processes using time Petri nets. In: Proc. of the 9th Int'l Conf. on Quality Software. Washington: IEEE Computer Society Press, 2009. 92−97. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05381507 [doi: 10.1109/QSIC.2009.20]

[42] Song W, Ma XX, Cheung SC, Dou WC, Lü J. A public-view approach to timed properties verification for B2B Web service compositions. In: Proc. of the 6th IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society Press, 2009. 427−434. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5283926 [DOI: 10.1109/SCC.2009.18]

[43] Ly LT, Rinderle S, Dadam P. Semantic correctness in adaptive process management systems. In: Dustdar S, Fiadeiro JL, Sheth A, eds. Proc. of the 4th Int'l Conf. on Business Process Management. LNCS 4102, Berlin, Heidelberg: Springer-Verlag, 2006. 193−208. [doi: 10.1007/11841760_14]

[44] Ly LT, Rinderle S, Dadam P. Integration and verification of semantic constraints in adaptive process management systems. Data & Knowledge Engineering, 2008,64(1):3−23. [doi: 10.1016/j.datak.2007.06.007]

[45] van der Aalst WMP, Dumas M, Gottschalk F, ter Hofstede AHM, Rosa ML, Mendling J. Correctness-Preserving configuration of business process models. In: Fiadeiro J,Inverardi P, eds. Proc. of the 11th Int'l Conf. on Fundamental Approaches to Software Engineering. LNCS 4961, Berlin, Heidelberg: Springer-Verlag, 2008. 46−61. [doi: 10.1007/978-3-540-78743-3_4]

[46] Weber B, Rinderle S, Reichert M. Change patterns and change support features in process-aware information systems. In: Krogstie J, Opdahl AL, Sindre G, eds. Proc. of the 19th Int'l Conf. on Advanced Information Systems Engineering. LNCS 4495, Berlin, Heidelberg: Springer-Verlag, 2007. 574−588. [doi: 10.1007/978-3-540-72988-4_40]

[47] Weber B, Reichert M, Rinderle-Ma S. Change patterns and change support features—Enhancing flexibility in process-aware information systems. Data & Knowledge Engineering, 2008,66(3):438−466. [doi: 10.1016/j.datak.2008.05.001]

[48] Kradolfer M, Geppert A. Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In: Proc. of the 4th IECIS Int'l Conf. on Cooperative Information Systems. Washington: IEEE Computer Society Press, 1999. 104−114. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=792162 [doi: 10.1109/COOPIS.1999.792162]

[49] Reichert M, Dadam P. ADEPT$_{flex}$-supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems, 1998,10(2):93−129. [doi: 10.1023/A:1008604709862]

[50] Rinderle S, Reichert M, Dadam P. Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases, 2004,16(1):91−116. [doi: 10.1023/B:DAPD.0000026270.78463.77]

[51] van der Aalst WMP, Jablonski S. Dealing with workflow change: Identification of issues and solutions. Int'l Journal of Computer Systems Science & Engineering, 2000,15(5):267−276.

[52] Song W, Ma XX, Cheung SC, Hu H, Lü J. Preserving data flow correctness in process adaptation. In: Proc. of the 7th IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society Press, 2010. 9−16. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05557209 [doi: 10.1109/SCC.2010.24]

[53] Fu X, Bultan T, Su JW. Synchronizability of conversations among Web services. IEEE Trans. on Software Engineering, 2005, 31(12):1042−1055. [doi: 10.1109/TSE.2005.141]

[54] Massuthe P, Serebrenik A, Sidorova N, Wolf K. Can I find a partner? Undecidability of partner existence for open nets. Information Processing Letters, 2008,108(6):374−378. [doi: 10.1016/j.ipl.2008.07.006]

© 中国科学院软件研究所 http://www.jos.org.cn

[55] Bordeaux L, Salaün G, Berardi D, Mecella M. When are two Web services compatible? In: Shan MC, *et al*., eds. Proc. of the 5th Int'l Workshop on Technologies for E-Services. LNCS 3324, Berlin, Heidelberg: Springer-Verlag, 2005. 15−28. [doi: 10.1007/978-3-540-31811-8_2]

[56] Stahl C, Massuthe P, Bretschneider J. Deciding substitutability of services with operating guidelines. Trans. on Petri Nets and Other Models of Concurrency, LNCS 5460, 2009,2:172−191. [doi: 10.1007/978-3-642-00899-3_10]

[57] Lohmann N, Massuthe P, Wolf K. Operating guidelines for finite-state services. In: Kleijn J, Yakovlev A, eds. Proc. of the Int'l Conf. on Application and Theory of Petri nets and Other Models of Concurrency. LNCS 4546, Berlin, Heidelberg: Springer-Verlag, 2007. 321−341. [doi: 10.1007/978-3-540-73094-1_20]

[58] Wolf K. Does my service have partners? Trans. on Petri Nets and Other Models of Concurrency, LNCS 5460, 2009,2:152−171. [doi: 10.1007/978-3-642-00899-3_9]

[59] Martens A. Consistency between executable and abstract processes. In: Proc. of the 2nd IEEE Int'l Conf. on e-Technology, e-Commerce and e-Service. Washington: IEEE Computer Society Press, 2005. 60−67. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01402269 [doi: 10.1109/EEE.2005.53]

[60] van der Aalst WMP, Weske M. The P2P approach to interorganizational workflows. In: Dittrich KR, Geppert A, Norrie MC, eds. Proc. of the 13th Int'l Conf. on Advanced Information Systems Engineering. LNCS 2068, Berlin, Heidelberg: Springer-Verlag, 2001. 140−156. [doi: 10.1007/3-540-45341-5_10]

[61] König D, Lohmsnn N, Moser S, Stahl C, Wolf K. Extending the compatibility notion for abstract WS-BPEL processes. In: Proc. of the 17th Int'l Conf. on World Wide Web. New York: ACM Press, 2008. 785−794. http://www2008.org/papers/pdf/p785-koenigA.pdf [doi: 10.1145/1367497.1367603]

[62] Castagna G, Gesbert N, Padovani L. A theory of contracts for web services. In: Proc. of the 35th ACM Sigplan-Sigact Symp. on Principles of Programming Languages. New York: ACM Press, 2008. 261−272. http://portal.acm.org/citation.cfm?id=1328471 [doi: 10.1145/1538917.1538920]

[63] Ye CY, Cheung SC, Chan WK. Process evolution with atomicity consistency. In: Proc. of the 2nd Int'l Workshop on Software Engineering for Adaptive and Self-Managing Systems. Washington: IEEE Computer Society Press, 2007. Article No.19. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4228619

[64] Casati F, Ilnicki S, Jin LJ, Krishnamoorthy V, Shan MC. Adaptive and dynamic service composition in eFlow. In: Wangler B, Bergman L, eds. Proc. of the 12th Int'l Conf. on Advanced Information Systems Engineering. LNCS 1789, Berlin, Heidelberg: Springer-Verlag, 2000. 13−31. [doi: 10.1007/3-540-45140-4_3]

[65] Zhang J, Cheng BHC. Model-Based development of dynamically adaptive software. In: Proc. of the 28th Int'l Conf. on Software Engineering. New York: ACM Press, 2006. 371−380. http://portal.acm.org/citation.cfm?id=1134337 [doi: 10.1145/1134285.1134337]

[66] Lam L, Tang Q, Zou ZL, Fong L, Frank D. Identifying data constrained activities for migration planning. In: Proc. of the 7th IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society Press, 2009. 364−371. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05283934 [doi: 10.1109/SCC.2009.33]

[67] Ellis CA, Keddara K. A workflow change is a workflow. Business process management, models, techniques, and empirical studies. LNCS 1806, Berlin, Heidelberg: Springer-Verlag, 2000. 201−217. [doi: 10.1007/3-540-45594-9_13]

[68] Liske N, Lohmann N, Stahl C, Wolf K. Another approach to service instance migration. In: Baresi L, Chi CH, Suzuki J, eds. Proc. of the 7th Int'l Joint Conf. on Service-Oriented Computing and ServiceWave. LNCS 5900, Berlin, Heidelberg: Springer-Verlag, 2009. 607−621. [doi: 10.1007/978-3-642-10383-4_44]

[69] Maoz S. Using model-based traces as runtime models. IEEE Computer, 2009,42(10):28−36. [doi: 10.1109/MC.2009.336]

[70] Rinderle-Ma S, Reichert M, Weber B. Relaxed compliance notions in adaptive process management systems. In: Li Q, *et al*., eds. Proc. of the 27th Int'l Conf. on Conceptual Modeling. LNCS 5231, Berlin, Heidelberg: Springer-Verlag, 2008. 232−247. [doi: 10.1007/978-3-540-87877-3_18]

[71] Song W, Ma XX, Dou W C, Lü J. Toward a model-based approach to dynamic adaptation of composite services. In: Proc. of the 6th IEEE Int'l Conf. on Web Service. Washington: IEEE Computer Society Press, 2008. 561−568. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4670221 [doi: 10.1109/ICWS.2008.65]

[72] Rinderle S. Schema evolution in process management systems [Ph.D. Thesis]. Ulm: Ulm University, 2004.

[73] Sun P, Jiang CJ. Analysis of workflow dynamic changes based on Petri net. Information and Software Technology, 2009,51(2):284−292. [doi: 10.1016/j.infsof.2008.02.004]

[74] Hicks M, Nettles S. Dynamic software updating. ACM Trans. on Programming Languages and Systems, 2005,27(6):1049−1096. [doi: 10.1145/1108970.1108971]

[75] Stoyle G, Hicks M, Bierman G, Sewell P, Neamtiu I. Mutatis mutandis: Safe and predictable dynamic software updating. ACM Trans. on Programming Languages and Systems, 2007,29(4):Article 22. [doi: 10.1145/1255450.1255455]

[76] Subramanian S, Hicks M, McKinley KS. Dynamic software updates: A VM-centric approach. In: Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation. New York: ACM Press, 2009. 1−12. http://portal.acm.org/citation.cfm?id=1542478 [doi: 10.1145/1542476.1542478]

[77] Oreizy P, Gorlick MM, Taylor RN, *et al*. An architecture-based approach to self-adaptive software. IEEE Intelligent System, 1999, 14(3):54−62. [doi: 10.1109/5254.769885]

© 中国科学院软件研究所 http://www.jos.org.cn

[78] Garlan D, Cheng SW, Huang AC, Scomerl B, Steenkiste P. Rainbow: Architecture-Based self-adaptation with reusable infrastructure. IEEE Computer, 2004,37(10):46−54. [doi: 10.1109/MC.2004.175]

[79] Huang G, Wang QX, Mei H, Yang FQ. Research on architecture-based reflective middleware. Journal of Software, 2003,14(11): 1819−1826 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1819.htm

[80] Ma XX, Yu P, Tao XP, Lu J. A service-oriented dynamic coordination architecture and its supporting system. Chinese Journal of Computers, 2005,4(28):467−477 (in Chinese with English abstract).

[81] Kramer J, Magee J. The evolving philosophers problem: dynamic change management. IEEE Trans. on Software Engineering, 1990, 16(11):1293−1306. [doi: 10.1109/32.60317]

[82] Vandewoude Y, Ebraert P, Berbers Y, D'Hondt T. Tranquility: A low disruptive alternative to quiescence for ensuring safe dynamic updates. IEEE Trans. on Software Engineering, 2007,33(12):856−868. [doi: 10.1109/TSE.2007.70733]

[83] Yang FQ, Lü J, Mei H. Technical framework for internetware: An architecture centric approach. Science in China (E), 2008,38(6): 818−828 (in Chinese with English abstract). [doi: 10.1007/s11432-008-0051-z]

[84] Lü J, Ma XX, Huang Y, Cao C, Xu F. Internetware: A shift of software paradigm. In: Proc. of the of the 1st Asia-Pacific Symp. on Internetware. New York: ACM Press, 2009. Article No.7. http://portal.acm.org/citation.cfm?id=1640213 [doi: 10.1145/1640206.1640213]

[85] Li C, Reichert M, Wombacher A. Discovering reference process models by mining process variants. In: Proc. of the 6th IEEE Int'l Conf. on Web Service. Washington: IEEE Computer Society Press, 2008. 45−53. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4670158 [doi: 10.1109/ICWS.2008.13]

[86] Sirin E, Parsia B, Wu D, Hendler JA, Nau DS. HTN planning for Web service composition using SHOP2. Web Semantics: Science, Services and Agents on the World Wide Web, 2004,1(4):377−396. [doi: 10.1016/j.websem.2004.06.005]

[87] Rinderle-Ma S, Reichert M. Adjustment strategies for non-compliant process instances. Technical Report, Ulm: Ulm University, 2009.

[88] Rinderle S, Wombacher A, Reichert M. Evolution of process choreographies in DYCHOR. In: Meersman R, *et al.*, eds. Proc. of the OTM Confederated Int'l Conf. LNCS 4275, Berlin, Heidelberg: Springer-Verlag, 2006. 273−290. [doi: 10.1007/11914853_17]

[89] Wombacher A. Alignment of choreography changes in BPEL processes. In: Proc. of the 7th IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society Press, 2009. 1−8. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5284036 [doi: 10.1109/SCC.2009.11]

附中文参考文献:

[7] Dumas M, van der Aalst WMP, ter Hofstede AHM,著;王建民,闻立杰,译.过程感知的信息系统.北京:清华大学出版社,2009.

[8] 范玉顺.工作流管理技术基础.北京:清华大学出版社,2001.

[9] 吕建,陶先平,马晓星,胡昊,徐锋,曹春.基于 Agent 的网构软件模型研究.中国科学(E 辑),2005,35(12):1233−1253.

[15] 宋巍,马晓星,吕建.Web 服务组合动态演化的实例可迁移性.计算机学报,2009,32(9):1816−1831.

[40] 宋巍,窦万春,刘茜萍.时间约束 Petri 网及其可调度性分析与验证.软件学报,2007,18(1):11−21. http://www.jos.org.cn/1000-9825/18/11.htm [doi: 10.1360/jos180011]

[79] 黄罡,王千祥,梅宏,杨芙清.基于软件体系结构的反射式中间件研究.软件学报,2003,14(11):1819−1826. http://www.jos.org.cn/1000-9825/14/1819.htm

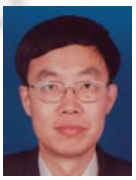
[80] 马晓星,余萍,陶先平,吕建.一种面向服务的动态协同架构及其支撑平台.计算机学报,2005,4(28):467−477.

[83] 杨芙清,吕建,梅宏.网构软件技术体系:一种以体系结构为中心的途径.中国科学(E 辑),2008,38(6):818−828.

宋巍(1981－),男,山东日照人,博士,讲师,CCF 会员,主要研究领域为服务计算,形式化方法,软件工程与方法学.

胡昊(1975－),男,博士,副教授,CCF 会员,主要研究领域为面向开放软件环境的过程技术,软件过程,软件质量保障技术.

马晓星(1975－),男,博士,教授,CCF 会员,主要研究领域为 Internet 软件技术,软件体系结构和软件构件,软件工程与方法学.



吕建(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件形式化与自动化,面向对象方法与技术,软件工程与方法学.

© 中国科学院软件研究所 http://www.jos.org.cn