用Pi演算为业务过程建模的生命周期

郭小群 郝克刚 侯 红 丁剑洁
(西北大学信息科学与技术学院 西安 710069)

摘 要 随着企业竞争日趋激烈,业务过程建模技术变得越来越重要。由于形式化方法降低了二义性并为模型的分析和验证提供了可行性,因此形式化的业务过程建模技术在学术界引起了很多人的关注,但到目前为止仍缺乏一套既能方便地进行过程建模,又具有对模型进行形式化分析与验证的整套理论体系。从生命周期的角度入手,探讨如何把形式化方法更有效地应用于商业过程建模。主要工作在于提出了基于Pi演算的生命周期,探讨了生命周期各个阶段使用的技术和工具。

关键词 Pi演算,生命周期,业务过程建模
中图法分类号 TP311.52 **文献标识码** A

Lifecycle of Modeling Business Process Based on Pi Calculus

GUO Xiao-qun HAO Ke-gang HOU Hong DING Jian-jie
(School of Information Science and Technology, Northwestern University, Xi'an 710069, China)

Abstract With the dramatic competition of enterprise, the technology of business modeling gets more and more important. Formal business process model receives a lot of attention in the scientific community because formal methodology is proved effective in avoiding ambiguity and feasible in supporting analysis and verification of model. But at present, there is not suitable systemic theory and methodology to support modeling business process easily and to analyze and verify model formally. This paper discussed how to use formal methodology when modeling business process at the viewpoint of the lifecycle. The contribution of this paper is giving a lifecycle based on Pi calculus, discussing technology and tools used in every stage of lifecycle.

Keywords Pi calculus, Life cycle, Modelling business process

随着网络的出现以及业务复杂度的提高,一个企业或组织无法实现需要的所有业务,在过程执行中需要与其他组织的过程进行协作来交换数据,通知状态变化,请求远程过程调用或者同步执行等。组织内的过程不再是孤立的,它需要跨越组织边界,与其他组织内的过程建立联系。业务过程通过网络而跨组织的方式带来的一个问题就是它的部件是自动的,成员是动态的,没有中央集权。所以在开发支持这种模式的工作流系统时,必须提高系统的健壮性。由于它们在一个不可预知的动态环境中运行,这意味着很难正确地构造它们,也很难用传统中受控制的方式测试它们,因此在这类系统部署前如何确保其具有正确的行为,消除系统的缺陷以及安全性威胁就变得非常重要。而形式化分析技术和相应的验证技术为解决这一问题提供了一种合适的手段。目前业务过程建模技术常用的就是Petri网[1,2]。但遗憾的是,使用Pi演算建模使得即使一个简单的问题也会产生状态爆炸的问题[3]。Pi演算原语非常简单但表达力却非常丰富:通道名可以被创建并被用来通信(这使它具备了动态的重新配置过程参与者的可能性),而且它们受到复杂范围规则的约束,因此Pi演算非常适合作为工作流的理论基础。但是虽然形式化的业务过程建模技术得到学术界的广泛关注,可是到目前为止仍缺乏一套既能方便地进行过程建模,又具有对模型进行形式化分析与验证的整套理论体系。本文在这种背景下提出了业务过程建模技术的框架。

1 过程模型

软件工程的生命周期理论指导人们在开发一个软件系统时应该怎样做才能得到一个更好的软件系统。同理,为了更好地指导对业务过程建模,我们认为一个基于形式化方法的业务过程也有自己的生命周期,而且生命周期方法也提供了一个更为宽广的、系统化的观点去考察形式化建模技术中的各种活动。通过研究业务过程形式化建模时应包括的主要活动,以及这些活动之间的关系,本文给出了一个基于Pi演算的业务过程建模时的过程模型(如图1所示)。当然,这一模型也适合于其他形式化方法。模型中主要包含问题获取、建立BPEL模型、建立Pi演算模型、Pi演算模型的分析和验证以及建立Pi演算模型的执行引擎5个活动。先对图1中包含的活动做简单介绍,详细的介绍将在后面给出。

像软件工程一样,建立模型的第一个阶段是问题获取,即

到稿日期:2008-12-23 返修日期:2009-03-02 本文受国家高可信研究开发计划(2007AA010305)资助。
郭小群 女,博士生,讲师,主要研究方向为软件工程,E-mail:guoxiaoq@nwu.edu.cn;郝克刚 男,博士生导师,主要研究方向为软件工程;侯 红 女,副教授,主要研究方向为软件度量;丁剑洁 女,博士生,讲师,主要研究方向为软件度量。

了解、分析用户的需求。这一阶段的结果为建立模型提供输入。在建立 Pi 演算模型时，由于 Pi 演算理论本身的复杂性，也许只有极少数人能直接从用户的需求中创建一个形式化的模型，因此这一阶段允许用户采用非形式化的语言建立模型，然后通过一个转换工具得到 Pi 演算模型。这里选择的非形式化语言是 BPEL，主要是因为 BPEL 代表了商业过程执行语言，它被看作事实上的工业标准用来描述商业过程。当得到一个 Pi 模型后，可以用相应的工具对 Pi 模型进行验证。如果验证结果不能达到用户要求，则说明过程方案有问题，这时需要修改模型。如果通过，则可以部署过程模型到一个执行引擎上，使过程在实际中执行起来。根据本模型，部署的时候有两种选择：一是把经过分析验证的 BPEL 模型部署到执行引擎上，这一选择不在形式化技术的研究范围之内；另外一种就是把经过验证的 Pi 模型部署到执行引擎上，这一选择要求提供一个能够解释 Pi 模型的执行引擎。

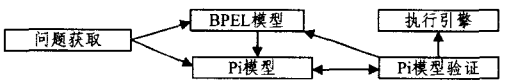

图 1　基于 Pi 演算业务过程的过程模型

2　问题获取

问题获取阶段最主要的目标是理解组织的业务过程，包括了解业务的目标和任务。每一个组织都有一个不同的起始点，也有不同的需求。有些已经有一个定义好的过程，另外一些却可能没有。有些想强调过程的自动化，而另一些却需要更好的可追踪性、可视化和性能度量。不管是什么，第一个目标是必须理解过程。特别是业务过程再造时，修改业务过程时最容易犯的一个错误是没有全面调查和理解当前的过程和目标，没有投入足够的时间去研究、分析、计划。许多商业人员发现他们并没有充分解决原来的问题，或者他们仅简单地把已识别的问题用一个未知的问题替换，所以成功的业务过程再造的关键在于全面理解目前过程的细节并精确预测过程变化的结果。

为了理解组织的业务过程，特别是复杂的过程，要确保范围正确地识别，需要尽量多问为什么。这一阶段业务分析师可以采取访谈与会议的形式，也就是业务分析师以个别访谈或小组会议的形式开始初步的沟通。在访谈与会议过程中，业务分析师应该多问几个为什么，通过对这些问题的回答获取有关流程或存在的问题，并逐步理解过程如何设计才能达到业务目标。

例如业务分析师对于人力资源过程的理解就可以通过问下面的几个为什么[4]：1）人力资源过程无效，为什么？2）花费太长的时间招到一个新员工，为什么？3）招聘人员和人力资源经理在过程中的什么地方决定候选者，为什么？4）没有办法知道最希望的开始时间以及决定什么时候面试，为什么？

问题获取阶段强调的是理解过程，并不是建立一个模型以转换成代码或给出执行过程的定义。这样就能使用户和业务分析人员站在旁观者的角度看待业务过程，并且允许对过程有不同的看法。通过这一阶段的活动，业务分析师需要了解并记录过程的所有关键方面。例如：

1）过程流程是什么？

2）过程中所用的资源是什么？

3）过程处理的过程项是什么？

4）活动的输入和输出是什么？

5）任务的变化及什么时候发生变化？

6）可选择的任务有哪些？

7）完成任务的描述是什么？

8）和任务相关的角色有哪些？

在这一阶段产生的问题是人们总是想知道所有可能潜在的路径、所有异常、所有潜在的活动，这种情况容易导致分析瘫痪。分析人员应该明白，当以计算机实现业务流程的管理时，其中一个主要的目的是证明这种方式对业务的有效性，而不是所有的可能情况都需要计算机实现。只有让用户先明白这种有效性，才可能去实现更多功能。

为了更好地和用户沟通，可能需要一些流程定义工具的支持。目前支持这一阶段的工具有很多，例如 BM 的 WebSphere，Activebpel 组织开发的开源工作流引擎等，它们都提供了图形化的流程定义工具。

3　创建 Pi 演算模型

形式化业务过程模型能够模拟真实世界业务操作的目标。一个合适的业务过程模拟会清楚了解资源使用情况以及组织的性能，业务过程将被部署并且完成它的功能。有效的过程模型能够通过模拟评价业务过程，通过如果则什么的分析方法以及再工程提高过程设计。这一阶段的主要任务就是在前一阶段的结果的基础上利用 Pi 演算创建业务过程的形式化模型。

Pi 演算[2,5]起源于 1991 年，是由图灵奖得主 Robin Milner 参照物理学大统一理论提出的。它是一种描述和分析并发系统的计算模型，用动态演化结构表示过程间的间歇性的相互作用，名称是基本的概念，值、变量和管道通过名称引用。

对于很多需求者及商业分析人员和开发人员来说，由于非形式化语言容易理解和阅读，因此在工业界和学术界有很多人研究基于非形式化语言的建模技术，例如 UML AD，BPMN，BPEL。而直接基于形式化建模的技术相对研究较少，而且大多局限于学术界，例如 Petri 网。但是由于非形式化语言的不精确性、二义性，因此为了弥补二者之间的差别，许多人研究非形式化语言到形式化语言的转换。所以根据前一节的分析结果，创建形式化模型时，有两种选择：一是直接创建形式化模型，另一种方式是先创建非形式化模型，这里指的是 BPEL 模型，然后把它转化为形式化模型。通过转化过程，消除非形式化语言的二义性和不精确性。

为了方便创建 Pi 演算模型，可以用模式的概念。一个模式是一个具体形式的抽象，它们在特定的非任意环境中重复出现。Gamma 等人首先系统地区分了 23 个设计模式[6]，这些设计模式描述了面向对象系统中较少重复出现的交互方式。设计模式提供了和具体实现技术无关的独立性以及和它们需要解决的领域的基本需求无关的特性。

图 2 描述了创建一个 Pi 演算模型的过程。如果选择直接创建 Pi 演算模型，首先从第一阶段的用户需求中提取工作流模式，再把这些工作流模式描述成 Pi 演算，然后用这些 Pi 演算模式组合成模板。模板是业务模型在一些小的问题域的参考模型（例如货物的接收、投诉处理），其目的是处理特定问

题领域并给出可被直接重用的更复杂的工作流程。模板比模式更为具体。模式从更一般的角度描述了工作流元素(分支、顺序、同步),与模板相比较,它对流程部件的直接复用的支持力还不够大。一个用户流程可以通过组合模板来获得。

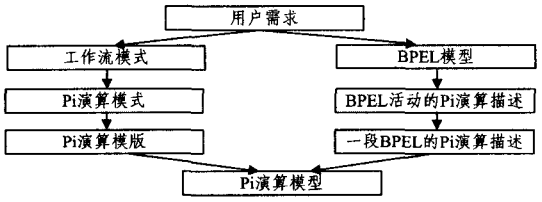

图 2 建模过程

在创建 Pi 演算模型时,另外一种选择是采用 BPEL 语言去描述第一阶段产生的结果,然后把 BPEL 语言的描述转换为 Pi 演算,以进行形式化的分析与验证。转换时首先将 BPEL 的活动转换为 Pi 演算,然后把描述一个特定问题领域的 BPEL 语言转换为一个 Pi 演算模板,最终通过模板组合得到用户业务流程的 Pi 演算模型。

这一阶段主要的技术就是要研究如何得到一个 Pi 演算模型,其中包括 Pi 演算是否有足够的表达能力去描述所有可能的模式? 如何描述? BPEL 语言中所有的成分都能被 Pi 演算描述吗? 如何描述? 实际上这些问题已经引起了学术界的广泛关注,很多学者都在研究它们[7,8]。

虽然有很多人在研究这些技术,但到目前为止,这一阶段的支持工具几乎没有。那么本阶段需要的支持工具有哪些? 我们认为至少需要如下 3 个工具:1)建立基于 Pi 演算的模式库;2)建立基于 Pi 演算的模板库;3)提供 BPEL 语言中转换为 Pi 演算的规则库。

当有了工具的支持以后,本阶段的流程可被修改为如图 3 所示。建模者首先根据用户的需求描述判断某一段工作流包含哪些模式,然后查找模式库中是否已有这些模式的 Pi 演算模型。如果有,则使用;如果没有,则创建它,并把它放入模式库。如果所包含的所有模式已经存在于模板库中,则根据用户需求描述,查看某一段工作流是否已在模板库中。如果在,则提取、修改,以满足用户需求;如果不在,利用已有的模式进行组合,形成新的模板并把它放入库中。如果用户开始提供的是一个 BPEL 文件,则可用转换规则直接转换为 Pi 演算模板,这时也可考虑是否把它作为模板放入模板库中。

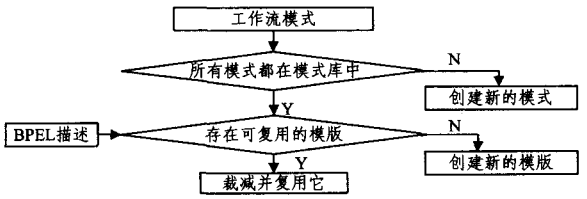

图 3 工具支持的建模过程

4 模型的验证与确认测试

在建立了模型之后,需要对模型进行验证与确认测试。验证测试就是检查模型是否正确以及解决方案是否合适。确认测试就是检查是否有正确的模型以及解决方案的行为是否和实际系统一致。

Pi 演算根据是否和外界环境的交互,其行为可通过两种方式来模拟:一个是规约,主要描述了系统自身的演化;另外一个是标签转移系统,它描述了系统通过和外界环境的交互而不断演化。规约和标签转移系统通常被定义为规则的集合。基于规约和标签转移系统,Pi 演算在状态集合上定义了许多不同的等价关系,这些行为等价关系用来捕获哪些状态具有相同的行为。

根据前一阶段得到 Pi 演算模型的方式的不同,可以从两个不同方面进行模型的确认与验证测试:一是对工作流系统的测试。可以通过 Pi 演算模型检查工作流系统的一些特性,例如:

1)可达性。通过可达性的分析,不但可以检查希望到达的状态是否可达,而且可以检查流程是否到达了一个不希望的状态。

2)等价性:检查两个流程是否在行为上是等价的。

3)安全性:主要检查信息在两个过程中传递时是否会泄露。

4)死锁:检查多个流程并发推进时是否会有死锁的发生。

5)缺乏同步性:主要检查几个过程是否同步完成。

另一个就是对 BPEL 语言的验证。像其他语言一样,BPEL 用结构化的英语定义,这种定义形式从外观上看也许很清晰,但实际上它很容易带来不一致性、二义性、不完整性的问题。所以为了保证 BPEL 描述的行为的正确性,必须采用一些技术和工具对 BPEL 进行验证。通过 Pi 演算模型可以验证 BPEL 语言的如下性质。

1)可用性:所描述的过程能通过和外界交互正常终止。

2)兼容性:两个 BPEL 语言描述的过程组合起来是可用的。

3)模拟性:一个 BPEL 语言描述的过程可以模拟另外一个。

4)等价性:两个 BPEL 语言描述的过程是等价的。

目前,已有一些基于 Pi 演算的确认与验证工具,例如 MWB,PICNIC,ABC,MIHDA,PIET[9] 等。通过模型的确认与验证,可以不断改进过程模型,使得过程在执行阶段得到更好的保证。

5 模型的执行

一个工作流模型必须部署到能够解释它的工作流引擎上,运行时引擎根据工作流模型派遣新的任务到工作列表,那些已完成的任务从它所处的环境中返回信号给工作流。引擎的一个主要功能是根据环境做出决策。

Pi 演算除了能够描述工作流的一些基本结构外,其他的一些描述能力,如对数据结构的描述、对于并发性的描述、移动性的描述、同步和异步通信的描述、与外界环境交互的描述等决定了 Pi 演算完全可以作为一种实现工作流引擎的工作流语言。

所以在这一阶段,为了能使基于 Pi 演算的模型运行起来,需要开发一个能够解释 Pi 演算模型的引擎。

结束语 关于业务过程的形式化建模技术,已经有很多人研究。本文主要从生命周期的角度给出了基于 Pi 演算的业务过程的建模以及分析与验证的框架,给出了主要的框架活动以及每一框架活动的主要目标。同时指出了为了更好地完成每一阶段的目标,在这一阶段可采用的工具和技术应包

(下转第 195 页)

本文采用以下 3 个基准函数进行测试。

(1) Rosenbrock 函数

$$f_1=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2] \quad -100\leqslant x_i\leqslant 100$$

(2) Rastrigrin 函数

$$f_2=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x)+10] \quad -100\leqslant x_i\leqslant 100$$

(3) Griewank 函数

$$f_3=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \quad -100\leqslant x_i\leqslant 100$$

4.2 实验参数

惯性因子最小值为 0，最大值为 0.9，种群大小为 100、进化代数取 100，3 个基准函数的自变量维数均取 10，即 $n=10$。

4.3 实验环境

算法使用 matlab7.1 代码，运行于 AMD 1.79GHz，384MB 内存的普通 PC 机上。

4.4 实验结果

3 个函数的惯性因子变化曲线分别如图 2、图 3、图 4 所示。

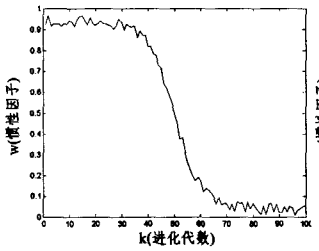

图 2 Rosenbrock 函数的惯性因子变化曲线

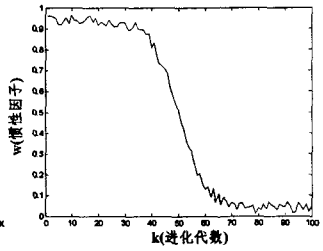

图 3 Rastrigrin 函数的惯性因子变化曲线

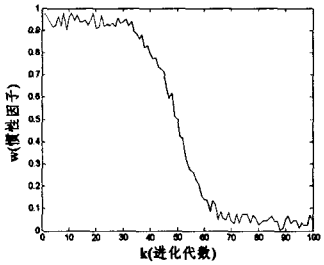

图 4 Griewank 函数的惯性因子变化曲线

表 1 分别给出了本文算法与 APSO 的 3 个函数的 20 次运行的平均最小适应值(均值±方差)。

表 1 两类算法运行 20 次的均值与方差

函数	APSO	本文算法
f_1	28.7665±31.2046	14.6624±12.3836
f_2	8.3372±6.2498	6.1126±4.1532
f_3	0.1136±0.0694	0.0561±0.0231

4.5 实验结果分析

从实验结果来看，本文算法较 APSO 优化性能有了较显著的提升。但由于每进化一代就要对粒子进行聚类，因此，本文算法耗时相对较长。其实，本文算法对聚类精度要求不高，因此，在保证一定精度的前提下，采用更加高速的聚类算法是本文算法进一步改进的一个方向。

结束语 本文在分析惯性因子在算法中的作用机理的基础上，设计了一个根据种群多样性和进化代数自适应调节的惯性因子，并运用试探法，通过变换搜索步长，提高了算法的局部搜索能力。实验证明，与 APSO 相比，本文的改进极大提高了算法的优化精度。进一步的工作将是提高聚类速度，以缩短寻优时间。

参 考 文 献

[1] Kennedy J, Eberhart R. Particle Swarm Optimization[A]//Proceedings of IEEE International Conference on Neural Networks [C]. Perth: IEEE Press, 1995

[2] 黄岚，王康平，周春光，等. 粒子群优化算法求解旅行商问题[J]. 吉林大学学报：理学版，2003，41(4)：477-480

[3] 张鹏帅，霍红卫. 多序列比对问题的粒子群优化算法求解[J]. 计算机工程与应用，2005，41(18)：84-87

[4] 黄友锐，等. 智能优化算法及其应用[M]. 北京：国防工业出版社，2008：93-100

[5] Shi Y H. Experimental Study of Partiele Swann Optimization [C]//Proceedings of SCI Conference. Orlando, FL, 2000

[6] Mo Y R, Xue X P. On stability of delayed cellular neural networks with sigmoid output functions[J]. Science in China, 2003, 46(5): 371-380

[7] Mutihac R, Cicuttin A, Mutihac R C. Entropic approach to information coding in DNA molecules[J]. Materials Science and Engineering, 2001, 12: 51-60

[8] Ivasm S I, Ikl P M. Adaptive probabilities of crossover and mutation in genetic algorithm[J]. IEEE Transactions on System, Man and Cybernetics, 1994, 24(4): 656-667

(上接第 159 页)

括哪些。当然这个框架也可用于其他的形式化建模技术。将来的工作就是在这一框架下开发支持这些框架活动的工具，并最终建立一个集成化的支持环境。

参 考 文 献

[1] van der Aalst W, van Hee K. Workflow Management Models, methods and systems[M]. The MIT Press, 2000

[2] Li Shuyou, Song Binheng. Normalized workflow net (NWF-net): Its definition and properties. doi: 10.1016/j.future.2005.02.003

[3] Milner R, Parrow J, Walker D. A Calculus of Mobile Processes Information and Computation. 1992

[4] Derek M E. Best Practice B P M. ACM QUEUE, March 2006

[5] Milner R. Communicating and Mobile Systems: The Pi Calculus [M]. Cambridge, UK: Cambridge University Press, 1999

[6] Gamma, et al. Design pattern[M]. 北京：机械工业出版社，2002

[7] Lucchi R, Mazzara M. A pi-calculus based semantics for WS-BPEL[J]. The Journal of Logic and Algebraic Programming, 2007, 70: 96-118

[8] Abouzaid F, Mullins J. A Calculus for Generation, Verification and Refinement of BPEL Specifications[J]. Electronic Notes in Theoretical Computer Science, 2008, 200: 43-65

[9] Crafa S, Mio M, Miculan M, et al. PicNIc Pi-calculus Non-Interference Checker[C]//29th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency. Xi'an, China, June 2008