基于π演算的工作流建模方法研究

黄贤明[1,2]，李长云[1]，梁爱南[1]

（1.湖南工业大学 计算机与通信学院，湖南 株洲 412008；2.中南大学 信息科学与工程学院，湖南 长沙 410083）

摘 要：工作流模型缺乏一种支持过程定义以及过程分析的形式化数学模型，而π演算是一种移动进程代数运算，可用于对并发和动态变化的系统进行建模。首先提出了基于π演算的工作流建模方法，然后经过对多种建模工具的比较和分析后，利用π演算对业务流程结构进行了形式化定义，详细地阐述了各种活动和依赖关系在π演算中的表示。该方法是完全形式化的方法，具有较强的语义表达能力，便于工作流的执行、推理和仿真等。

关键词：业务流程；工作流建模；π演算

中图分类号：TP391 **文献标志码**：A **文章编号**：1673-9833(2009)04-0045-04

Research on Modeling Method of Workflow Based on π-Calculus

Huang Xianming[1,2]，Li Changyun[1]，Liang Ainan[1]

（1. School of Computer and Communication，Hunan University of Technology，Zhuzhou Hunan 412008，China；
2. School of Information Science and Engineering，Central South University，Changsha 410083，China）

Abstract: Workflow model is short of a kind of formal mathematics model which support process definition and process analysis. The π-calculus is a kind of mobile process algebra which can be used to model concurrent and dynamic systems. Firstly, proposes a modeling method for workflow based on the π-calculus. Furthermore, after comparing and analyzing a variety of modeling tools, makes a formal definition for workflow structure by using π-calculus, and clarifies the expression of activities and dependency relations in π-calculus. The method is a complete formal method, which has a strong capability in semantics expression and is easy to implement, reason and simulate workflows.

Keywords: business process; workflow modeling; π-calculus

0 引言

目前，工作流建模理论基础的研究不是很成熟，还没有统一的建模方法标准。近年来，国内外学者提出了许多有关工作流建模的方法。Hommes在文献[1]中指出，到目前为止大约有350多种支持BPR（Business Process Reengineering）的流程建模技术和工具被提出并实现，这些建模技术各有特色，但是在进行业务流程建模时都存在一定的缺陷[2]。总体上讲，大多数工作流过程模型在形式化语义、可视化、易修改性和描述的全面性等方面难以均衡，健壮性和柔性不足，工作流过程模型还缺乏一种支持过程定义以及过程分析的形式化数学模型。

π演算（π-calculus）是Robin Milner提出的以进程间的移动通信为研究重点的并发理论，是对CCS (Calculus of Communication System)的发展[3-4]。π演算是系统交互行为建模的理论基础，适合描述动态系统。在工作流建模阶段，使用π演算有助于清楚地描述工作流；在模型建立后，可利用π演算来推演系统的行为，同时验证模型的正确性，如发现系统行为不完整、死锁、缺少同步等；另外，π演算作为一种强大和成熟

收稿日期：2008-12-05

基金项目：国家自然科学基金资助项目（60773110）

作者简介：黄贤明（1976-），男，湖南汉寿人，湖南工业大学讲师，中南大学硕士研究生，主要研究方向为网格计算，工作流管理等，E-mail：shjshxm@21cn.com

的形式化方法，具有支持其正确性验证和相关应用的工具。此外，文献[5]还讨论了用π演算作为BPM（Business Process Management）的1个形式化基础的适用性问题，即π演算适用于BPM领域发生的3个主要的变化：BPM从基于状态的系统到基于消息的系统；随着“集成”变成BPM的1个核心活动，BPM从集中式引擎到分布式系统；从封闭式环境到开放环境。文献[6]说明了π演算适合描述服务编制（Service Orchestrations）和服务编排（Service Choreographies），而它们将是未来业务流程管理的核心基础。因此，π演算是对工作流作形式化研究的有力武器，具有良好的研究前景。例如，文献[7]使用π演算对工作流进行了初步的形式化。

1 π演算语法

在π演算中，进程是并发运行实体的单位，并以名字来统一定义通道名以及在通道中传送的消息，每个进程都有若干与其它进程联系的通道，数据结构在这里被封装为与特定通道相关的进程，外部进程通过这些通道来操作相关结构。其基本计算实体为名字和进程，进程之间的通信是通过传递名字来完成。由于π演算不但可以传递CCS中的变量和值等，还可以传递通道名，并将这几种实体都统称为名字而不再作区分，这使得π演算具有了建立新通道的能力，因此π演算可以用来描述结构不断变化的并发系统。

π演算有几种不同的符号表示，下面介绍π演算的基本语法，可由以下BNF（Backus Normal Form）范式给出：

$P ::= M \mid P \mid P \mid vzP \mid !P$，

$M ::= 0 \mid \pi.P \mid M+M$，

$\pi ::= \langle y \rangle \mid x(z) \mid \tau \mid [x=y]\pi$。

π演算中最简单的实体是名字，进程通过名字进行交互，并在交互中传递名字，名字的语法定义是标识符。上述π演算语法定义中，x是单个名字，$\bar{x}$与x互为对偶名字（Co-names），进程通过对偶名字进行交互。P是π演算中另一实体进程的语法定义，而M是π演算中“和”(summations)的语法定义，表达选择关系。π的语法给出进程能够执行的4种动作称为前缀，进程通过执行这些动作而进行演化。

π演算语法的形式语义运用约简关系和变迁关系定义，下面运用自然语言给出它的直观定义。

1）0表示非活动进程，即不做任何工作的进程。

2）前缀$\pi.P$表示具有π表示的单一行为能力，该能力执行后，执行进程P。

输出前缀$\bar{x}\langle y \rangle.P$表示通过名字$x$输出名字$y$，然后执行进程$P$；

输入前缀$x(z).P$表示通过名字x输入名字，并用输入的名字替换进程P中的z，然后执行替换后的进程P；

哑前缀$\tau.P$表示做哑动作τ，然后执行P，一般来说，τ用于表示进程外部不可见的内部动作；

匹配前缀$[x=y]\pi.P$表示当x和y是同一名字时，执行$\pi.P$，否则不做任何工作。

3）“和”——M_1+M_2表示选择执行M_1、M_2中的一个（其中M_1、M_2为进程表达式）。

4）组装$P_1|P_2$表示并行执行P_1和P_2。

5）限制vzP表示名字z是P的局部名字，P在名字z上的外部动作被禁止，但P通过名字z的内部通信是允许的。

6）复制$!P$表示无限多个P的组装。

2 π演算工作流建模

在利用π演算对工作流进行建模时，各个流程被分解成许多不同类型的活动，这些活动依据它们之间的依赖关系进行组合。根据WFMC（Workflow Management Coalition）有关工作流过程定义接口规范的描述，对业务流程进行如下形式化定义。

2.1 不同类型活动的定义

活动在工作流模型中是一个重要的、原子的、不能再分的概念。根据活动被触发的类型，可分成4种类型：自动活动、人工活动、消息触发活动和时间触发活动。自动活动指的是不需要人的参与，活动被使能的同时触发；人工活动要求人的参与，活动的执行由任务执行者触发；消息触发活动指的是活动的激活需要某一事件或者是消息的到来才能完成，比如说是E-mail；时间触发活动指的是活动由控制时间的定时器来触发。

在活动执行过程中需要调用系统或者外部的资源，这些资源包括人、应用程序、引擎内部的过程等等。工作流管理系统认为流程就是由一个或多个活动按照一定顺序组合而成，系统只关心活动的交互和管理。例如，当一个活动被激活的时候，系统会分配资源执行这个活动，当这个活动执行完毕，系统被通知结束，然后释放分配给这个活动的资源。这样，活动就充当了一个“黑盒”，通过端口（通道）与外部环境交互。因此，可以用π演算中的进程来表示活动。下面介绍几种不同类型的活动在π演算中的表示。

定义1 一个活动定义为π演算中的一个进程。

定义2 自动活动在π演算中的表示如下（假设A是自动活动）：

$$A =_{def} \overline{start.require\_resource} < case\_id,\ activity\_id,\ resource\_id,\ assigned\_resource > .$$
$$assigned\_resource(case\_id,\ activity\_id,\ resource).\bar{a}.finish.\overline{release\_resource} < resource > .\overline{done}。$$

表达式中大写字母代表1个进程，也就是活动，小写字母表示正处于执行状态的活动。当自动活动被激活之后，进程通过通道require_resource申请资源。申请资源的参数包括：工作流实例的标识（case_id），活动实例的标识（activity_id），申请资源的标识（resource_id）及用来接收资源的通道assigned_resource。私有通道assigned_resource用来接收分配给活动的资源（resource）、工作流实例的标识（case_id）、活动实例的标识（activity_id）。活动通过通道assigned_resource得到系统分配的资源后，便开始执行活动。当活动接收到通道finish发出执行完毕的确认信息后，由通道release_resource释放它所占用的资源（resource），最后进程向系统发出活动执行完毕的信号并激活下一个活动。

定义3　人工活动在π演算中的表示如下（假设A是人工活动）：

$$A =_{\text{def}} \text{start}.\overline{\text{require\_resource}} < \text{case\_id, activity\_id, resource\_id, assigned\_resource} > .$$
$$\text{assigned\_resource(case\_id, activity\_id, resource)}.\text{wait\_user(role\_id)}.\bar{a}.\text{finish}.\overline{\text{release\_resource}} < \text{resource} > .\overline{\text{done}}。$$

定义3与定义2类似，不同之处是：定义3中，当系统分配资源后，人工活动必须使用通道wait_user等待用户完成任务，只有在用户完成操作以后，才会继续运行过程。

定义4　消息触发活动在π演算中的表示如下（假设A是消息触发活动）：

$$A =_{\text{def}} \text{start.event(event\_id)}.\overline{\text{require\_resource}} < \text{case\_id, activity\_id, resource\_id, assigned\_resource} > .$$
$$\text{assigned\_resource(case\_id, activity\_id, resource)}.\bar{a}.\text{finish}.\overline{\text{release\_resource}} < \text{resource} > .\overline{\text{done}}。$$

定义4与定义2类似。不同之处是：定义4中，活动必须等待通道event的事件或者是消息的到来才能激活该活动，通道event中有输入名字event_id。

定义5　时间触发活动在π演算中的表示如下（假设A是时间触发活动）：

$$A =_{\text{def}} \text{start}.\overline{\text{time}} < \text{from time, to time, } t > .t().\overline{\text{require\_resource}} < \text{case\_id, activity\_id, resource\_id,}$$
$$\text{assigned\_resource} > .\text{assigned\_resource(case\_id, activity\_id, resource)}.\bar{a}.\text{finish}.\overline{\text{release\_resource}} < \text{resource} > .\overline{\text{done}}。$$

定义5与定义2类似。不同之处是：定义5中，当系统分配资源后，活动要求从通道time中输出3个名字：from time，to time，t。from time代表活动执行的开始时间，to time代表活动执行的结束时间，t代表1个私有通道，它用来接受时钟发出的触发信号。当活动接收到时钟发出的信号之后，才能激活该活动。

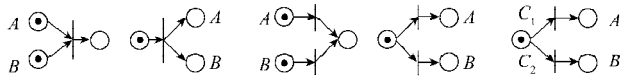
2.2　依赖的定义

一个工作流是由活动之间的复杂的依赖关系和活动组成的。为了可表达更加高级的关系，工作流管理联盟定义了几种基本的控制结构，包括AND-分支（AND-split），AND-聚合（AND-join），OR-分支（OR-split）和OR-聚合（OR-join）。AND-分支节点表示从1个节点分裂出多个并发的线程，AND-聚合表示多个同步的线程组成1个单独的线程，OR-分支节点在有多个可选分支的情况下，只能选择其中的1个分支。存在一种特殊的OR-分支，它可以选择多个可选的分支，但它的选择是建立在显性的条件的基础上，我们可以称它为条件依赖选择。OR-聚合节点表示多个分支在没有同步的情况下汇流成1个单独的线程。在Petri网中对控制结构的语义描述如图1所示。

定义6　如果活动A和活动B之间有一种顺序依赖关系，它们有各自对应的进程表达式，它们之间的顺序依赖关系可以被定义为：

$$A \text{ Before } B =_{\text{def}} (vp)(\{p/\text{done}\}A|\{p/\text{start}\}B)。$$

这个进程表达式说明：在进程A完成后，它通过一对互补的通道和B交互，然后进程B被执行。



a）AND-join b）AND-split c）OR- join d）OR-split e）condition

图1　Perti net对依赖关系的表示图

Fig. 1　Representation of perti net to dependencies

定义7　如果活动A和活动B是一种OR-分支的依赖关系（如图1中的d）），可以这样被定义：

$$A \text{ OR-split } B =_{\text{def}} (vp)((\text{start}.\bar{p})|\{p/\text{start}\}A+\{p/\text{start}\}B))。$$

从上面的表达式可以看出，当控制元素OR-分支节点使能之后，通过一对互补的通道$(\bar{p},p)$触发进程A或进程B的执行。

定义8　如果活动A和活动B存在着OR-聚合依赖关系（如图1中的c）），可以这样被定义：

$$A \text{ OR-join } B =_{\text{def}} (vq)((\{q/\text{done}\}A+\{q/\text{done}\}B)|q.\overline{\text{done}})。$$

从上面的表达式可以看出，当进程A或者进程B被执行完之后，它会触发done动作，从而触发下一个活动。

定义9　如果活动A和活动B存在着AND-分支

依赖关系（如图1中的b）），可以这样被定义：

$$A\ \text{AND}-\text{split}\ B=_{\text{def}}(vp)((\text{start}.\overline{p}|\overline{p})|(\{p/\text{start}\}A|\{p/\text{start}\}B))。$$

从上面的表达式可以看出，当控制元素AND-分支节点使能之后，进程A和进程B将并发的开始，私有通道P用来触发进程A和B。

定义10 如果活动A和活动B存在着AND-聚合依赖关系（如图1中的a）），可以这样被定义：

$$A\ \text{AND}-\text{join}\ B=_{\text{def}}(vq)((\{q/\text{done}\}\text{A}|\{q/\text{done}\}B)|(q.\overline{w}|q.\overline{w}|w.\overline{\text{done}}))。$$

在上面的表达式中，进程A和B通过一对互补的通道$(\overline{q},q)$各自交互，其中私有通道q用来处理2个进程之间的同步。

定义1~10描述的是2个活动存在依赖关系时的情形，对于多个活动存在上述依赖关系时，可以进行类似的描述。

定义11 如果条件表达式φ为真就执行活动A，否则就执行活动B。对于这种情况通常这样定义：

$$\text{If}\ \varphi\ \text{then}\ A\ \text{else}\ B=_{\text{def}}(vxd_1d_2)\text{start}.\overline{\text{condition}}<x,\varphi>.x(\text{result})([\text{result}=\text{True}]\overline{d}_1+[\text{result}=\text{False}]\overline{d}_2))|(\{d_1/\text{start}\}A|\{d_2/\text{start}\}B)。$$

在上面的定义式中，condition通道输出1个私有通道x和1个条件表达式φ，其中通道x是用来接受条件式φ的值。

子流程是一种可重用的流程片断，它与调用它的父流程具有同样的上下文数据和状态。子流程大大增强了模型的表达能力，使模型具有了层次化的概念。

定义12 如果A是1个子流程，在π演算中用P_{sub}表示，假设它的父流程是B，那么可以这样定义：

$$A\ \text{IS}-\text{SUB}-\text{OF}\ B=_{\text{def}}(v\ \text{not if}\ y\ \text{ack})(\text{start}.\text{not if}\ y.\text{ack.done}|\{\text{not if}\ y/\text{start},\ \text{ack/done}\})P_{\text{sub}}。$$

在上面的定义式中，当父流程B从其它的流程中接受到1个启动信息之后，由它发出1个消息通知子流程，这时子流程被触发并执行。当子流程执行完毕之后它又需要发送1个ack信息给父流程，然后父流程B发送消息done给其它的流程并告知B已经结束。

3 结语

本文利用π演算对业务流程结构进行了形式化方法描述，详细地阐述了各种活动和依赖关系在π演算中的表示。使用基于π演算的建模方法能清晰地表达业务流程行为，提出的方法是完全形式化的，具有较强的语义表达能力，便于工作流的执行、推理和仿真等。未来的工作将使用π演算来研究工作流中的数据流、资源配置模式等问题。基于π演算理论的业务流程的证明与互模拟机制也是进一步要解决的问题。

参考文献：

[1] Hommes B J. Overview of Business Modeling Tools[EB/OL]. [2001-06-21]. http://is.twi.tudelft.nl/homm/tools.html.

[2] Yu hongyan，Alex Bejan. Modeling Workflow within Distributed Systems[C]//Weiming Shen，Zongkai Lin，Jean-Paul A，et al. Proceedings of International CSCW Conference. London：[s.n.], 2001：433-439.

[3] Milner R. Communicating and Mobile Systems：The π-Calculus[M]. Cambridge：Cambridge University Press，1999：70-75.

[4] Smith H，Fingar P. Business Process Management - The Third Wave[M]. Tampa：Meghan- Kiffer Press，2002.

[5] Puhlmann F. Why Do We Actually Need the Pi-Calculus for Business Process Management?[C] //Abramowicz W，Mayr H. Proceedings of the 9th International Conference on Business Information Systems. Klagenfurt：[s.n.]，2006：77-89.

[6] Overdick H，Puhlmann F, Weske M. Towards a Formal Model for Agile Service Discovery and Integration[C]//Kunal Verma，Amit Sheth，Michal Zaremba，et al. International Workshop on Dynamic Web Processes， associated with the 3rd International Conference on Service Oriented Computing. Amsterdam：[s.n.]，2005：25-37.

[7] Wil van der Aalst. Pi Calculus Versus Petri Nets：Let Us Eat “Humble Pie” Rather Than Further Inflate the “Pi Ype”[EB/OL].[2005-05-31]. http://is.tm.tue.nl/research/patterns/download/ pi-hype.pdf.

（责任编辑：罗立宇）