DOI:10.16208/j.issn1000-7024.2010.17.014

• 软件与算法 •

基于 Pi 演算和 EECA 规则的跨组织工作流建模研究

钟晓雄， 张远平

(兰州理工大学 计算机与通信学院，甘肃 兰州 730050)

摘 要：针对跨组织环境下的工作流形式化建模技术的不足问题，提出了一种基于 Pi 演算和 EECA(Extended-ECA)规则的跨组织工作流建模方法。将 ECA 规则在时间上进行扩展，解决了 ECA 规则在描述与时间相关的复杂过程时能力不足问题。利用 Pi 演算技术，对跨组织的本地工作流的协同方式进行了形式化描述。分析了协同模型及其执行过程，并给出了一个应用实例验证了提出的模型具有较好的柔性，可以有效用于本地工作流间的协同工作，并且适合于对分布式协同环境下的工作流进行建模。

关键词：跨组织工作流；EECA 规则；Pi 演算；本地工作流；柔性

中图法分类号：TP311 **文献标识码**：A **文章编号**：1000-7024 (2010) 17-3831-04

Modeling research for inter-organizational workflow based on Pi-Calculus and EECA rule

ZHONG Xiao-xiong, ZHANG Yuan-ping

(School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China)

Abstract: After analyzing some defects in the existing modeling methods for inter-organizational processes, a formal modeling method for inter-organizationa1 workflow based on pi-calculus and EECA (Extended-ECA) rule is proposed. Firstly, the EECA rule, which extends in time, resolves the problems of ECA rule's deficient in describing complex process related to time, is presented. At the same time, the collaborated manner of the local workflow in the inter-organizational workflow is described by using the pi-calculus technology. Finally, the implementation process of the proposed model is analyzed and an example is provided to prove the flexibility of the proposed method. The proposed method works well in the collaboration among local workflow and appropriates for the modeling of distributed cooperative workflow systems.

Key words: inter-organization workflow; EECA rule; Pi-calculus; local workflow; flexibility

0 引 言

由于Internet网络应用和分布式计算环境的不断发展，跨组织的业务越来越频繁，对组织之间进行协同工作能力的要求也越来越高，跨组织工作流技术已经成为新一代业务集成的迫切需求，跨组织工作流具有分布性、自治性，不确定性因素多，控制难度大，而跨组织工作流建模技术又要求组织之间共同参与协同协作。现有的工作流技术虽然能够较好地支持简单协同工作模型，然而在处理较为复杂的跨组织环境下的业务流程时，还存在很多的问题[1]。Pi 演算是一种移动进程代数，具有严格的形式化语义，具备描述和解释并发流程互通信行为的能力，是一种描述并发和动态变化系统的计算模型，非常适合用于跨组织工作流建模。文献[2-5]利用 Pi 演算形式化描述了工作流，同时验证模型的正确性，如发现系统的不完整，死锁，缺少同步等，这些研究工作重点仍是关注于同一个组织内部的传统工作流，并未涉及对跨组织工作流的研究。文献[6]中利用 Pi 演算的并发计算操作符，将跨组织业务流程建模为一组自治且并发执行的组织内子流程的组合，并且基于 Pi 演算的弱互相似理论，验证了两个跨组织子流程外部行为的相等性。但是，都没有考虑到现实业务环境复杂多变，确定组织之间及组织内部的工作流管理系统的哪些部分可以和其他组织的工作流交互，什么条件下，什么时候交互通信。因为各个组织往往是为了一个暂时的共同目标进行合作，共同目标实现了，这种合作关系也就不存在了。

综上所述，目前基于 Pi 演算的建模技术不能很好地用于跨组织建模，有必要引入新的建模方法，基于这种情况，本文提出了基于 Pi 演算和规则的工作流建模方法，首次利用 Pi 演算对本地工作流之间的协同关系进行了描述，并给出了一个应用实例及分析了其执行过程。所给出的模型可以充分、精确地形式化描述工作流程的实际交互过程，具有较好的柔性。

1 基于 Pi 演算和 EECA 规则的工作流建模

1.1 Pi 演算

Pi 演算是以过程间移动通讯为研究重点的并发理论，是对 CCS(calculus of communication system)的发展[7]。其基本计算实体为名字和过程，过程之间的通讯通过名字来完成。Pi 演

收稿日期：2009-10-16；修订日期：2010-01-25。

作者简介：钟晓雄 (1983－)，男，广东韶关人，硕士研究生，研究方向为工作流技术、智能 agent； 张远平 (1966－)，男，安徽无为人，博士，教授，研究方向为算法设计与分析、信息论。E-mail：xiaoxiong_zh@yahoo.cn

算可用来描述结构不断变化的分布、动态、协作、并发的系统。

Pi 演算的基本语法定义如下[7]：

设 N 为无限名字集，用u,v,w,x,y,z等小写字母表示名字集上的名字；过程标识符用A,B,C等大写字母表示；过程表达式用P,Q,R等大写字母表示，过程表达式是以下几种表示之一：

P::=0	空表达式
P_1+P_2	和表达式
$P_1 \mid P_2$	并行表达式
$(x)P$或者$(vx)P$	限制表达式
$[x=y]P$	匹配表达式
$y(x).P$	输入前缀
$\bar{y}<x>.P$	输出前缀
$\tau.P$	哑前缀
! P	重复表达式
$A(y_1,y_2,y_3,\cdots,y_n)$	过程标识符

1.2 跨组织的工作流建模

跨组织工作流是由多个独立组织共同参与协同协作的工作流程，这些组织之间的关系经常发生变化，并且组织内部结构也会发生变化。跨组织工作流具有分布性，不确定性因素多，控制难度大，并且需要不同组织之间进行协同。结合分层思想，跨组织工作流可以分为两个层次实施：全局工作流与本地工作流。一个跨组织工作流的流程由多个组织中的相关业务流程组成，每个组织内部的流程由相关活动序列组成。

定义 1 本地工作流(Local Workflow)：本地工作流是密切相关的活动的序列，实质上是一个子工作流。本地工作流 LWf 可以定义为活动集$A_j(j=1,\cdots,n)$。其执行的细节对于外部组织是隐藏，但本地工作流的执行需要与全局工作流进行交互，从而保证全局工作流的顺利执行。这样就不需要了解整个全局工作流过程，实现了各组织内部工作流执行的自治性，同时也方便流程管理。

定义 2 全局工作流(Global Workflow)：全局工作流是指在一个跨组织工作流中能够被抽取表示全局任务执行序列的过程。组织的应用可以描述为多个不同任务的序列，每个任务由一个本地工作流完成，这些任务是松散耦合的。工作流 GWf 可以定义为本地工作流的集合$LWf_i(i=1,\cdots,m)$。

考虑到现实业务环境复杂多变，要确定组织之间及组织内部的工作流管理系统的哪些部分可以和其他组织的工作流交互，什么条件下，什么时候交互通信，我们引入规则——扩展 ECA 规则。如果我们把业务环境的相关变化抽象成一定的规则，则规则在企业中就是一个个灵活多变的因素，如果将规则直接定义给流程，那么流程的灵活性增加，却不利于流程的重用。因此，将规则分离，可以提高规则的重用性，因而，在我们提出的模型中，需要可以建立一个规则库。

1.3 ECA 规则

ECA(事件-条件-动作)规则，早期广泛用于主动数据库技术中，由于 ECA 规则能灵活地描述系统行为，它的应用已延伸到电子商务，工作流建模等众多领域。在本文中，为了更好地描述本地工作流之间的协同，本文对 ECA 规则进行了扩展。EECA 规则可以表示为一个七元组(E,C,A,NULL,T, N,CWfID)，其中 E 代表发生的事件集合，C 代表响应触发事件的条件集合，A 代表满足条件下对触发事件进行响应时采取的动作集合，NULL 代表空活动，它在执行中不作任何实际操作，T，N 为事先设置好的时间和触发次数参数，CWfID 为协同工作流的编号。

1.4 协同工作流及本地工作流的形式描述

为了更好地协同本地工作流，我们引入一种新的元工作流——协同工作流(CWf)，在文献[8]的基础上，我们对元活动 start 进行修改和添加一个元活动 finish。协同工作流有 6 种特定的活动，称为元活动。元活动是：start，terminate，suspend，wait，resume，finish，用来控制和操作本地工作流。

· start(wf, eventID)：在事件 eventID 触发下启动一个新的业务流程 wf；

· finish(wf)：该流程 wf 所有活动已经执行完毕。

其他活动的具体定义见文献[8]。元活动的状态(state)有：initiated，ready，running，suspended，done，terminated，aborted。

引入协同工作流后，我们可以本地工作流之间的几种关系进行 Pi 演算描述。

假定：A 的后续工作流是 A'，如果本地工作流中任何活动的状态为非 done，则本地工作流为 active。

(1)顺序(Sequence)——在工作流流程中，本地工作流 LWfj 在本地工作流 LWf_i 之后执行。

$PLWf_i$ = finish. [state = done].$\tau_{\cdot PLWfi}.\overline{event(eventID_j)}$ .0

$PLWf_j$ = event ($eventID_j$). [state = ready]. start (LWf_j, event ($eventID_j$)). [state = done].$\tau_{\cdot PLWfj}.\overline{event(eventID_k)}$. LWf_j'

本地工作流 LWf_i 完成之后，发送一个控制 LWf_j 的元工作流的事件，当事件处理模块接收到事件 event($eventID_j$)，启动对应的元工作流，判断 LWf_j 是否准备好，若其为 active，则准备启动 LWf_j。如果 LWf_j 顺利完成之后，继续执行其后续工作流。

(2)并行接合(AND—join)——当两个或多个并行本地工作流都完成后，下一个本地工作流才能开始执行时需要同步。

$PLWf_i$ = finish. [state = done].$\tau_{\cdot PLWfi}.\overline{event(eventID_k)}$.0

$PLWf_j$ = finish. [state = done].$\tau_{\cdot PLWfj}.\overline{event(eventID_h)}$.0

$PLWf_k$ = event ($eventID_k$).event ($eventID_h$). [state = ready].start (LWf_k, (event ($eventID_k$, $eventID_h$))). [state =done]$\tau_{\cdot PLWfk}.\overline{event(event\text{-}ID_1)}$.$LWf_k$'

本地工作流 LWf_i 完成之后，发送一个控制 LWf_k 的元工作流的事件，准备启动 LWf_k；同样 LWf_j 也执行类似的操作；事件处理模块接收到事件 event($eventID_k$)和 event($eventID_h$)后，启动对应的元工作流，控制 LWf_k 的执行。LWf_k 执行完之后，对后续本地工作流进行操作。

(3)或分支(XOR-split)——在一个本地工作流执行完成后，需要依据一个结果或流程控制数据，从多个分支路径中选定一个路径。

$PLWf_i$ = finish. [state = done].$\tau_{\cdot PLWfi}.\overline{event(eventID_j)}$. 0|$\overline{event(eventID_k)}$.0

$PLWf_j$ = event ($eventID_j$). [state = ready]. start (LWf_j, event ($eventID_j$)). [state = done].$\tau_{\cdot PLWfj}.\overline{event(eventID_h)}$. LWf_j'

$PLWf_k$ = event ($eventID_k$). [state = ready]. start (LWf_k, event ($eventID_k$)). [state = done].$\tau_{\cdot PLWfk}.\overline{event(eventID_1)}$. LWf_k'

本地工作流 LWf_i 完成之后，发送事件 event ($eventID_j$) 和 event($eventID_k$)，当事件处理模块接收到事件，启动对应的元工

作流且判断其对应的本地工作流是否为 active，若为 active，则控制其执行。完成之后，对后续的工作流进行操作。

1.5 协同模型及其执行过程

工作流模型是对业务流程结构中各种实体及其约束关系的描述。根据上述 Pi 演算及 EECA 规则，图 1 给出了基于 Pi 演算和规则的跨组织工作流的协同模型。

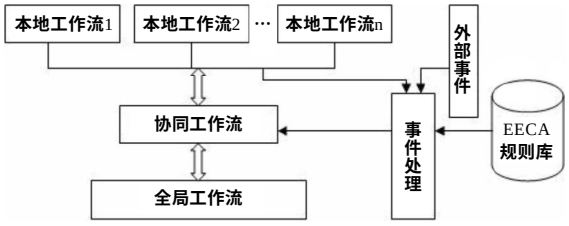

图 1 基于 Pi 演算和 EECA 规则的跨组织工作流的协同模型

为了更好地分析本模型，本文作如下的假设：

(1)我们假设全局工作流的顺利完成都是通过本地工作流有序完成。

(2)即时准则：$\forall goal(goal \in Goal \rightarrow \exists goal(LWf = <goal, A, f_{LWf}, CWfID> \wedge LWf \in GWf))$对应本地工作流的具体目标，EECA 规则一旦匹配就立即找到与之相应的协同工作流。

(3)每个本地工作流都是正确的。因为只有本地工作流都正确，才能在协同工作流下进行协同完成全局工作流。

执行过程如下：一个全局工作流有一个或多个本地工作流组成，因而需要本地工作流之间的协同，有效的配合才能更好地完成一个全局工作流的执行。当本地工作流触发的或外部事件 e，则根据 EECA 规则库进行匹配，若满足条件 c，则会通过规则解析器进行翻译或者解析，执行对应的协同工作流，若本地工作流是 active 的，则启动本地工作流，由协同工作流协同本地工作流的执行，如果经过时间 t，相应的协同工作流未被执行，则撤销此动作，再触发一次并检查条件和时间限制；若连续触发 n 次都未被执行，或者超时了，则撤销。转去执行另一动作 NULL。当一个本地工作流顺利执行后，返回信息给全局工作流，判断这整个流程是否完成执行。多个本地工作流同时执行时，则通过协同工作流来控制执行。全局工作流依赖于各个本地工作流及相互合作，若是其中的一个失败了，那么全局工作流的目标就不能实现，为此，我们可以用下面这个式子来表示

$$G = \sum_{1}^{m} \frac{M[LWf_i \wedge f_{LWf} \propto abilityA_i]}{LWf_i \in GWf} * m \cup CWf$$

此式可以说明，全局工作流的完成不光取决于单个本地工作流的能力，还取决于本地工作流之间的合作程度。

2 应用实例

下面以某个商城网上购物过程为例，对本文提出的跨组织工作流建模方法进行简要说明。购物过程存在 3 个相对独立的工作流流程，即是 client, shop.com, supplier。为便于突出主要问题，我们对工作流流程进行了简化。

LWf_1：顾客登录 shop.com 网站订购所需商品，然后等待商城送货，期间可以取消所订的商品。

LWf_2：商城收到订单后，需要验证顾客身份并联系当地代理商是否有现货，如果两者没有问题，且顾客没有取消订货，就送货上门。

LWf_3：当地代理商收到商城的购物通知，查询是否有现货，如果有就送货至商城，没有就通知商城，拒绝该顾客的订货。

用 Pi 演算表示如下

$LWf_1 = [state=ready].start(LWf_1, orderID).\tau.\overline{client\_oder(orderID)}.(\overline{ok} \mid \overline{not\_ok}).(ok.wait(LWf_2).wait(LWf_3).[state = done].receive\_product.0 | not\_ok.cancel\_order.0)$

$LWf_2 = [state = ready].client\_order(orderID).start(LWf_2, client\_order(orderID)).\tau.(\overline{ok} \mid \overline{not\_ok}).(ok.\tau.\overline{can\_send\_product}.[state = ready].start(LWf_3, can\_send\_product).\tau.(\overline{ok} \mid \overline{not\_ok}).(ok.send\_product.0 | not\_ok.cancel\_order.0) | not\_ok.cancel\_order.0)$

$LWf_3 = [state = ready].start(LWf_3, (client\_order(orderID), can\_send\_product)).\tau.(\overline{ok} \mid \overline{not\_ok}).(ok.send\_product.0 \mid not\_ok.cancel\_order.0)$

这种购物的整个过程要求 3 个本地工作流相互协作，通过协同工作流可以保证它们之间的关系不被破坏。例如：在工作流 LWf_1 中，它要等待 LWf_2 及 LWf_3 的执行结果，此时，我们可以触发相关事件启动对应的协同工作流，对它们进行控制操作。如果将规则直接定义给流程，那么流程的灵活性增加，却不利于流程的重用。如：在工作流 LWf_1 中，假设由于某种情况，在这个购物网上，在时间段 23:00—08:00 不允许订购商品，如果把规则直接定义给流程：$LWf_1 = [state = ready].start(LWf_1, orderID) <from\ 08:00:01\ to\ 22:59:59>.\tau.\overline{client\_order(orderID)}.(\overline{ok} \mid \overline{not\_ok}).(ok.wait(LWf_2).wait(LWf_3).[state = done].receive\_product.\ 0 \mid not\_ok.cancel\_order.\ 0)$，则这个流程不利于重用，为此，我们可以把这种情况认为是一条规则的条件，写入规则库中去，即：$if\ shopping\_time\ in <from\ 08:00:01\ to\ 22:59:59> \wedge [state=ready] \wedge receive\ the\ event\ of\ start\_LWf_1\ then\ start(LWf_1)$，同样，可以加以设置 EECA 规则中的参数 n(例如上述流程 LWf_1 中，n<=5)，限制事件触发的次数，从而一定程度上减少了系统中传输的事件。如果这种情况在将来又发生变化时，业务流程之间的协同也随之发生变化，则与其相关的业务流程不用修改，我们只需根据业务流程的环境变化对应的规则进行添加或者删除即可。将规则与流程分离，可以提高规则的重用性。从实例中可以看到，基于 Pi 演算和规则的建模技术的优势是模块性，灵活性，自适应性。

3 结束语

Pi 演算是一种进程代数，提供了并发操作符，具备描述和解释并发流程互通信行为的能力，是一种描述并发系统的计算模型，非常适合用于跨组织工作流建模。然而业务环境复杂多变，为了更好地描述跨组织工作流的执行，本文引入 ECA 规则，并对其进行扩展，增强了系统灵活性，同时一定程度上可以防止事件风暴。同时本文提出了一个基于 Pi 演算和规则的协同工作流模型并描述了其执行过程，最后给出了一个应用实例。跨组织工作流是一个分布，动态的环境，对工作流的异常处理及一致性是进一步需要解决的问题，下一步的研究工作将对模型的验证，事件的调度算法的优化及冲突处理。

参考文献:

[1] Aalst W. Inheritance of business processes: A journey visiting four notorious problems[C].Petri Net Technology for Communication-based Systems,2003:383-408.

[2] Förster M.Theory of business process modeling: The Pi-calculus [C]. Potsdam, Germany: Seminar Process-Oriented Information Systems,2003.

[3] Smith H, Fingar P. Workflow is just a Pi process [OL].http://www.bpm3.com/picalculus, 2003-12-08.

[4] Yang D,Zhang S S.Approach for workflow modeling using π-calculus[J].Journal of Zhejiang University Science,2003,4(6):643-650.

[5] 胡庆成,刑春晓,杨吉江,等.基于 PI-演算的网上并联审批业务流程建模及验证[J].计算机应用研究,2007,24(12):47-50.

[6] 张静, 王海洋, 崔立真. 基于 Pi 演算的跨组织工作流建模研究[J].计算机研究与发展,2007,44(7):1243-1251.

[7] Smith H,Fingar P.Business process management-the third wave [M].Tampa:Meghan-Kiffer Press,2002.

[8] 杜彦华,范玉顺.基于事件——状态——过程规则的跨组织工作流协同方法 [J]. 计算机集成制造系统, 2008,14 (7): 1342-1348.

(上接第 3742 页)

(3)设置接收 CRC 坏包 :设置 CTR 寄存器 RCV_BAD 位；

(4)PHY 环回时,设置 PHY 环回模式 :调用 PHY 寄存器写函数设置 PHY CONTROL 寄存器的 LPBK 位；

(5)调用发送函数,发送一个数据包；

(6)调用接收函数,接收数据包。

(4)接收 Ping 测试

在通过了以上测试之后，可基本保证以太网系统正常工作,但可能存在数据包丢失现象影响,而接收Ping测试正是用于测量以太网传输性能的。

Ping 接收测试主要程序流程见图 6，图中仅详细给出 ARP、IP 和 ICMP 协议判断和包验证部分,回复包构造需要具体参考 3 种协议的数据包格式[8]。在进行该测试时,将开发板与 PC 机相连,开发板运行该程序,PC 端 ping 开发板 IP,在 IP 端即可看到传输性能相关统计结果。

如果测试出丢包现象，应及时调整软硬件设计。导致传输性能下降可能的原因有 :①终端阻抗匹配电阻、共模噪声滤波电阻、发送电流配置偏置电阻阻止不合理 ;②处理器总线时序配置过快或过慢,导致发送或接受数据阻塞。

4 实验结果与分析

在以太网系统设计调试阶段，将以上功能测试和系统评估程序应用于第一、二批实验板，验证了评估程序的正确性，并验证了硬件方案设计的正确性,也加快了系统调试速度。在开发板小批量试产时,应用以上评估测试程序,也得到了很好的效果，检验出了相关器件质量问题。应用测试评估程序的实验结果详见表 1。实验也充分说明了测试评估程序的实用性和广泛适用性，遵循以上方法设计的测试程序可以应用于所有嵌入式以太网系统的开发、调试、生产等所有环节中。

5 结束语

本文所提出的基于 LAN91C111 的嵌入式以太网系统使用了 MN103E 为主控处理器，已经应用于批量生产的嵌入式开发板中,并且还用于构建了软件无线电实验平台,证明了本文所提出的系统的可行性、稳定性和可靠性。同时,本方案具有一定代表性和广泛的适用性,以太网控制器 LAN91C111 可与市面绝大多数嵌入式处理器搭配使用，在采用其它处理器时可依照本方案简单调整接口电路即可完成设计。

本文创新地提出了在板评估测试方法，改进了以往的测试流程,提高了工作效率,通过在方案验证、系统调试、批量生产和故障检测中的应用,证明所设计的评估方法完备可行,具有很大的应用价值。另外，该评估方法也普遍适用于各种包含以太网的嵌入式和计算机系统，在不同的平台下遵循以上方法并简单修改驱动,设计出的测试亦可得到很好的效果。

参考文献:

[1] SMSC.LAN91C111 data sheet[EB/OL].http://www.smsc.com/:2009-06-01.

[2] Panasonic. MN103E010H/040H LSI user's manual [EB/OL]. https://www.semicon.panasonic.co.jp/micom/manual/pdf/23301-020e.pdf, 2002-08.

[3] IEEE Computer Society.IEEE 802.3-2008,LAN/MAN CSMA/CD(Ethernet) access method[S].2008-09-26.

[4] 张嵩.嵌入式系统硬件设计与调试[M].北京:机械工业出版社,2006:262-264.

[5] 聂启忠,田建生,康勇.高频地波雷达以太网传输系统设计[J].计算机测量与控,2006,14(11):1535-1538.

[6] 刘暾东, 谢维盛, 余齐齐. 通信处理器的 Boot Loader 及 I/O 接口驱动程序[J].计算机工程,2008,34(3):280-282.

[7] 崔强,徐春荣,彭钢锋.嵌入式存储器在板测试软件的设计和实现[J].测控技术,2008,27:325-328.

[8] Gary R Wright,Richard Stevens W.TCP/IP illustrated Volume 2: The implementation[M].Beijing:China Machine Press,2002.

表 1 测试评估程序实验结果

实验名称	MAC 寄存器测试	PHY 识别测试	MAC 环回测试	PHY 环回测试	外部环回测试	接收 Ping 测试	结果分析
第 1 批实验板	通过	失败	通过	失败	失败	失败	LAN91C111 芯片中 PHY 部分损坏
第 2 批实验板 1	失败	失败	失败	失败	失败	失败	LAN91C111 虚焊造成
第 2 批实验板 2	通过	通过	通过	通过	通过	通过	验证了系统设计的正确性
小批量试产	通过	通过	通过	通过	失败	失败	隔离变压器质量不合格