第 35 卷 第 2 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.2 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)02—0074—02 文献标识码：A 中图分类号：TP311.5

基于对象 Petri 网的 BPEL 建模技术

林 强，胡 昊，吕 建
(南京大学软件新技术国家重点实验室，南京 210093)

摘 要：讨论了通过对象 Petri 网对业务过程执行语言(BPEL)进行建模，提供一种从抽象 BPEL 过程扩展生成可执行 BPEL 过程的方法，该方法保证生成的可执行 BPEL 过程遵循抽象 BPEL 过程定义的业务协议，并通过 BPEL2OPN 对建模结果进行仿真。
关键词：对象 Petri 网；业务过程执行语言；行为继承

BPEL Modeling Technique Based on Object Petri Net

LIN Qiang, HU Hao, LV Jian
(State Key Laboratory of Software New Technology, Nanjing University, Nanjing 210093)

【Abstract】 BPEL modeling technique based on object Petri net is proposed in this paper. An approach to extend abstract BPEL processes to executable ones is also provided, which guarantees the conformance to the business protocol specified by the abstract BPEL process. BPEL2OPN is used to simulate running the resulting model.
【Key words】 object Petri net; BPEL; behavior inheritance

1 概述

BPEL4WS(或简写为BPEL)能够用于定义业务过程以及如何与Web服务关联。这其中包括了如何利用Web服务实现业务过程的目标，以及如何通过业务过程对Web服务进行组合从而实现新的、增值的Web服务[1]。

BPEL可以对 2 种过程进行建模：抽象过程与可执行过程。抽象过程用于定义交互协议，它描述了不同业务伙伴之间消息交换的行为而不涉及到每个业务伙伴内部的行为。而一个可执行的业务过程则用于定义其构成活动的执行顺序，涉及到的业务伙伴以及这些业务合作伙伴之间的消息交互，还包括一些如异常及错误处理机制[2]。正是基于这一点，Leymann[1]指出，既可以从抽象BPEL过程定义的交互协议出发对其进行扩展生成可执行BPEL过程，也可以从一个可执行BPEL过程出发进行抽象得出一个用抽象BPEL过程描述的业务协议。

本文就是基于前者，即通过抽象BPEL过程定义的交互协议最终生成可执行的BPEL过程。然而由于BPEL本身并没有给出如何将抽象过程扩展为可执行过程，也无法直接从BPEL本身确定可执行BPEL过程是否遵循抽象过程所定义的交互协议。所以本文将首先将抽象过程转换为Petri网。在将BPEL或Web服务组合转换为Petri网方面已经出现了许多研究成果，如Rachid Hamadi将Web服务组合利用基于Petri网的代数来进行建模[3]。Aalst将BPEL转换为Petri网，定义了BPEL构成元素与Petri网之间的映射，从而利用Petri网的性质和工具对BPEL进行验证[2]。但他们都没有给出如何通过一个抽象BPEL生成一个可执行BPEL的方法，也没有给出如何保证一个可执行BPEL满足预定义的业务协议。

本文将引入对象Petri网对BPEL进行建模，对象Petri网是对传统Petri网的一种扩展[4]，从结构上讲，对象Petri网包含 2 层：系统网和对象网。它们都是Petri网，它们之间存在交互集。本文提到的对象网主要基于Valk在文献[4]中的定义。利用该方法，将抽象BPEL过程描述的交互协议利用对象Petri网中的系统网来定义，并且自动生成可执行BPEL过程的框架(用对象Petri网中的对象网来描述)，并通过交互集将两者关联起来。最后利用行为继承对生成的可执行BPEL过程框架进行扩展，从而保证了最终生成的可执行BPEL过程遵循抽象过程描述的交互协议。

2 对象 Petri 网简介

对象 Petri 网的最核心的思想之一就是将托肯(Token)看作是一个 Petri 网，从而实现了一种嵌套的思想。也就是说利用 Petri 网来替代 Token。

本文将利用 WF-Net 来描述对象 Petri 网中的系统网与子网，所以这里先给出 WF-Net 的定义。

2.1 WF-Net:

一个 Petri 网 $PN=\{P,T,F\}$ 是 WF-Net(WorkFlow Net 的缩写)的充分必要条件是：

(1)PN 有 2 个特殊的库所：i 和 o。其中，i 是一个源库所，$\bullet i=\varnothing$；o 是一个目标库所，且 $o\bullet=\varnothing$。

(2)如果向 PN 中加入一个变迁 t^*，并使得 $\bullet t^*=\{o\}\wedge t^*\bullet=\{i\}$，则 PN 变为强连通。

下面开始介绍对象Petri网，其中最简单的一种称为UEOS[4](Unary Elementary Object System)。

2.2 EOS——Elementary Object System

一个 EOS 是一个三元组，$EOS=\{SN,ON,\rho\}$。其中：

基金项目：国家“973”计划基金资助项目(2002CB312002)；国家“863”计划基金资助项目(2007AA01Z178, 2007AA01Z140, 2006AA01Z159)；国家自然科学基金资助项目(60736015, 60721002, 60603034, 60403014)；江苏省自然科学基金资助项目(BK2006712)
作者简介：林 强(1979－)，男，硕士，主研方向：工作流，软件过程，Petri 网；胡 昊，讲师；吕 建，教授、博士生导师
收稿日期：2008-10-14 **E-mail：**simon.linq@gmail.com

(1) $SN=(P,T,W,M_0)$ 是一个基本网系统，且 $|M_0|=1$，称为 EOS 的系统网。

(2) $ON=(B,E,F,m_0)$ 是一个基本网系统，称为 EOS 的对象网。

(3) $\rho \subseteq T\times E$ 是 T 与 E 之间的交互集。交互集 ρ 用于描述 SN 与 ON 之间的同步关系。

下面给出EOS的点火规则[4]。

2.3 EOS 的点火规则

EOS 系统的标识(marking)是一个有序对(M,m)，其中，M 是系统网 SN 的一个标识，而 m 则是对象网 ON 的一个标识。

系统网自主点火：对于 EOS 的标识(M,m)，对于 $t\in T$，如果 $\rho(t)=\varnothing$，且 t 在 M 中被激活，则 t 在(M,m)中被激活。且(M,m)的后继标识 (M',m') 可定义为：$M\xrightarrow{t}M'\ (SN),\ m=m'$。可以写为：$(M,m)\xrightarrow{[t,\lambda]}(M',m')$。

对象网自主点火： EOS 的标识(M,m)，对于 $e\in E$，如果 $\rho(e)=\varnothing$，且 e 在 m 中被激活，则 e 在(M,m)中被激活。且(M,m)的后继标识 (M',m') 可定义为：$m\xrightarrow{e}m'\ (ON),\ M=M'$。可以写为：$(M,m)\xrightarrow{[\lambda,e]}(M',m')$。

同步点火：对于 EOS 的标识(M,m)，对于$[t,e]$，如果 $[t,e]\in\rho$，并且 t 与 e 分别在 M 与 m 中被激活，则$[t,e]$在(M,m)中被激活。(M,m)的后继标识 (M',m') 可定义为：$M\xrightarrow{t}M'\ (SN),\ m\xrightarrow{e}m'\ (ON)$。可以写为：$(M,m)\xrightarrow{[t,e]}(M',m')$。

在以上定义中，λ 表示空发生序列，即没有任何变迁被点火：

$$\rho(t)=\{e\in E\mid(t,e)\in\rho\}$$
$$\rho(e)=\{t\in T\mid(t,e)\in\rho\}$$

3 BPEL 与对象 Petri 网的转换

Leymann 在文献[1]中指出，一个业务协议给出了对于一个特定的业务伙伴而言，它与其他业务伙伴的消息交换序列，从而实现最终的业务目标。业务协议不必给出消息的所有细节，隐藏了一些内部决策、消息构成以及一些内部处理活动。所以一个业务协议可以看作是其内部私有业务过程的一个外部视图。在 BPEL 中，可以通过抽象 BPEL 过程来描述一个业务协议。

本文给出的根据抽象 BPEL 过程生成可执行 BPEL 过程，并保证所生成的 BPEL 可执行过程遵循业务协议的方法主要包含以下步骤：

(1)将利用 Aalst 在文献[2]中的工作，将一个抽象 BPEL 过程生成一个带标注的 WF-Net，并将它作为对象 Petri 网中的系统网。

(2)根据(1)中生成的系统网生成对象网框架(也是一个 WF-Net)，并建立系统网与对象网之间的同步关系。

(3)根据(2)中生成的对象网框架利用行为继承进行扩展，从而最终生成完整的对象网。

(4)根据(3)中生成的带标注的对象网生成可执行 BPEL 过程。

下面是一个旅行代理帮助顾客订票的例子，首先代理将等待顾客的订票信息，处理后联系航空公司的售票系统，最后等待拿票。

```
<process name="ticketOrder">
    <receive name="processItinerary"
            variable="itinerary">
    </receive>
    <invoke name="contactAirline"
            variable="itinerary">
    </invoke>
    <receive name="receiveTickets"
            variable="tickets">
    </receive>
</process>
```

图 1 是根据该抽象 BPEL 过程生成的系统网。

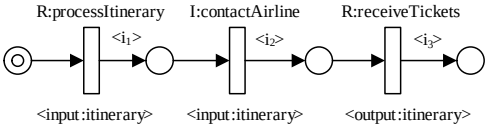

图 1　订票系统对象 Petri 网的系统网

图 1 是根据上文的 BPEL 描述利用 Aalst 的方法生成的 WF-Net。对此做了扩展，即将 BPEL 过程描述中的一些信息以标注的形式带到 WF-Net 中，从而保证了信息的完整。如：将 BPEL 中的活动类别分别以首字母简写的形式，如 R 代表 Receive、I 代表 Invoke 等。另外，变量则以输入输出的形式进行了区分。

图 1 中的Petri网将作为本文对象Petri网模型中的系统网。为保证最终扩展出来的可执行BPEL过程遵循图 1 中所描述的交互协议，将在图 1 的基础上利用行为继承(即过程协议一致性)[5-6]来对图 1 进行扩展。

例如在图 1 的基础上要加入用户登录环节，以及代理在联系航空公司的同时从客户信用卡上扣除费用的步骤，则扩展后的对象网如图 2 所示。

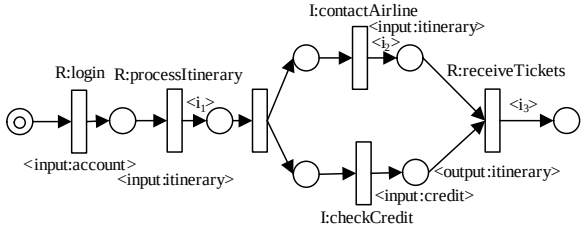

图 2　订票系统对象 Petri 网的对象网

根据对象Petri网关于同步集的定义，在图 1 和图 2 中具有相同标签的活动定义为具有同步关系。例如图 2 中的processItinerary与图 1 中的processItinerary具有相同的标签$<i_1>$，所以它们之间存在同步关系。根据对象Petri网的点火规则，具有同步关系的变迁必须同时点火才能够通过，从而可以验证图 2 是遵循图 1 所定义的交互规则的。下面将介绍BPEL2OPN仿真环境及仿真执行上述对象Petri网的结果。

4 BPEL2OPN 及仿真结果

BPEL2OPN 是基于开源软件 PIPE。PIPE 是一个针对传统 Petri 网的工具。它的功能包括绘图、仿真、Petri 网的分析(包括静态和动态分析)等。在此基础上引入了对象 Petri 网模型和 BPEL 分析器。对原来 PIPE 的 Petri 网模型做了修改。

将抽象 BPEL 过程输入进去之后，首先通过 BPEL 分析器生成 OPN，然后通过行为继承扩展生成对象网，最后仿真运行该模型。

图 3 和图 4 分别给出了仿真运行的结果。在图 3 与图 4 中，深色表示的变迁表示属于同步集合的。仿真运行的结果是系统网与对象网都正常结束。 (下转第 78 页)