支持快速构建领域应用系统的业务过程模型研究 *

杜栓柱，池成忠，赵会娟
（太原理工大学，太原 030024）

摘　要：针对基于业务基础平台快速构建领域应用系统，建立了业务过程模型 BPM4DBL（business process model for dynamic business logic）。模型中包括活动和规则元素，可以描述领域应用中宏观实体和微观知识结构；给出了模型中元素的构建步骤和原则，并定义了模型向 XPDL（XML process definition language）和 BPEL4WS（bussiness process execution language for Web service）的映射规则，从而支持领域应用系统的快速构建。最后给出了一个具体的应用实例。

关键词：业务过程模型；领域应用系统；业务基础平台
中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1001-3695（2008）03-0744-05

Study on business process model for rapidly constructing domain application system

DU Shuan-zhu，CHI Cheng-zhong，ZHAO Hui-juan
（*Taiyuan University of Technology*，*Taiyuan* 030024，*China*）

Abstract：Aiming at rapidly constructing application systems based on the domain application infrastructure platform，BPM4DBL（business process model for dynamic business logic）was built. BPM4DBL included rule activity and rule elements，and thus it could describe macro and micro domain applications. After that，the construction steps and principles of the elements of BPM4DBL were given，and then the mapping rules from BPM4DBL to XPDL（XML process definition language）and BPEL4WS（business process execution language for Web service）were defined. Finally，a case study was presented.

Key words：business process model；domain application system；business infrastructure platform

0　引言

在动态多变的市场环境中，如何在成本和质量指标约束下快速构建满足用户个性化需求的应用系统，是当前领域应用系统开发中面临的挑战。业务基础平台[1,2]一方面提供可直接适用于不同领域应用系统的基础模块，如组织结构管理、日志管理和安全管理模块，以及与中间件和数据库的连接模块；另一方面提供了业务过程模型驱动的领域应用系统开发框架，从而可以提高系统的开发效率及适应用户需求的动态变化。因此，业务基础平台的研究目前得到学术界和工业界的广泛重视。本文研究如何构建适应领域应用基础平台中的业务过程模型。

目前研究者已经建立了各种业务过程模型，而每种模型一般具有特定的适用范围。文献[3,4]通过总结近 20 年的业务过程模型研究，根据构建目标将其分为三类：应用于业务过程的一般性描述，如甘特图和 IDEF3 模型；应用于业务过程重组，如流程图和 IDEF0 模型；应用于软件系统开发和执行，如工作流、OMT 和着色 Petri 网模型。本文所研究的业务过程模型属于第三类。业务基础平台中业务过程模型需要适应两类领域应用的动态性变化：a）业务功能的动态性，即业务应用中领域知识或领域逻辑的变化；b）业务结构的动态性，即业务应用中任务的数目或执行顺序等发生变化。因此，如何在统一的模型框架下反映上述两类动态性，成为支持业务基础平台中业务过程模型的关键[5,6]。

本文在给出业务基础平台概念的基础上，定义了支持领域应用系统快速构造的业务过程模型 BPM4DBL。

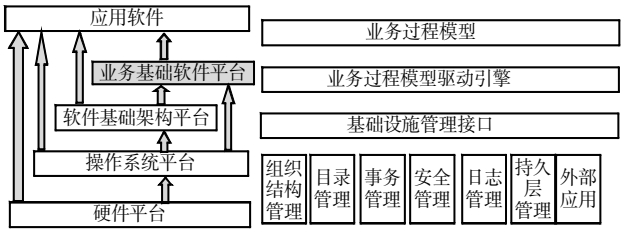
1　领域应用系统的快速构造

本文中领域应用系统主要面向业务密集型领域，如金融、保险、通信和电子政务等业务系统。一般来说，该类系统的业务层具有复杂多变的业务逻辑，技术层架构于数据库和中间件平台之上；同时，业务层表现为领域相关的业务实体，技术层表现为数据结构和函数等程序对象。因此，业务层和技术层之间的语义“鸿沟”以及技术层实现的复杂性使技术人员很难快速构造领域应用系统。业务基础平台试图缩小这种“鸿沟”以及降低技术层开发的复杂度，它用业务过程模型表示业务逻辑，通过业务过程引擎实现模型执行的自动化，并且封装引擎与中间件、数据库和操作系统的基础设施接口模块，从而屏蔽技术层的实现细节。本文将业务过程引擎及各个公共接口模块统称为业务过程执行环境。

图 1（a）为业务基础平台在软件系统中的位置[2]；（b）为业务基础平台的结构。包括业务基础平台的软件系统结构从底向上依次分为四层，即操作系统、数据库、中间件和业务基础平台。其中：业务过程模型既是面向不同的领域应用、领域用

收稿日期：2007-01-06；**修回日期**：2007-04-25　　**基金项目**：国家自然科学基金资助项目（60673121）；太原理工大学青年基金资助项目
作者简介：杜栓柱（1971-），男，博士，主要研究方向为知识管理、业务过程管理技术及理论（shuanzhu@163.com）；池成忠（1963-），男，教授，博士，主要研究方向为计算机辅助设计；赵会娟（1982-），女，硕士研究生，主要研究方向为计算机辅助设计.

户可理解的业务逻辑表示,又是系统开发技术人员可以通过业务过程执行环境驱动的可执行程序对象。业务过程模型一方面适应不同领域应用背景;另一方面又通过业务过程执行环境屏蔽了执行中的技术细节,从而在业务逻辑的可变性和技术的标准性之间获得恰当的平衡,由此提高开发效率。



(a) 位置　　(b) 结构

图 1　软件系统中业务基础平台的位置及结构

基于业务基础平台的系统构建采用业务过程模型驱动的开发方法:领域专家和软件开发技术人员首先构造 BPM4DBL 模型,同时针对具体的领域应用需求设计特定的业务功能模块;技术人员在业务过程执行环境中对 BPM4DBL 模型进行调试,获得正确、合理的模型,同时实现特定的业务功能模块;最后对 BPM4DBL 模型和业务功能模块集成测试,获得可靠的应用系统。需要指出的是,随着业务基础平台的成熟,针对不同领域应用需要订制的业务功能模块会逐渐减少,而公共模块则会逐渐扩充。由此可见,基于业务基础平台的系统构造只需要建立 BPM4DBL 模型和订制少量的业务功能模块,并且通过 BPM4DBL 模型适应业务逻辑的变化。BPM4DBL 模型因此成为业务基础平台成败的关键。

2　BPM4DBL 模型

一般来说,业务过程是活动的有序集合,通过活动之间结构化的逻辑偏序和依赖关系实现企业或组织的预定目标。由于本文着眼于业务密集型领域,为了适应领域应用基础平台业务过程的动态性,BPM4DBL 模型应该具有下述特点:a)可以从微观层面描述领域知识或领域逻辑的内容,如政策和法规约束等,即领域应用是做什么(what)的;b)可以从宏观层面描述领域应用中各个实体之间的关系,如前驱和后继等依赖,即领域应用是如何做(how)的。同时从应用层来说,BPM4DBL 模型需要便于领域专家和开发技术人员的定义,从技术层来说 BPM4DBL 模型需要支持分布式计算环境的动态执行。

2.1　BPM4DBL 概念模型

BPM4DBL 中概念模型既需要表达微观层面的领域知识或逻辑,又需要描述宏观层面实体之间的关系。业务规则可以方便地描述动态多变的领域知识或逻辑,但缺乏对业务实体结构间关系的宏观表达,而业务实体结构间的关系则可以通过业务实体语义来表达。为方便描述及突出主题,下面的概念模型定义中只给出了过程的基本属性,其他属性如活动的名称、相关数据和活动的时间限制等则忽略。

定义 1　BPM4DBL 模型为四元组,BPM4DBL = {AS, RS, RuleS, ES}。其中:AS 为活动集;RS 为活动间的结构关系集;RuleS 为规则集;ES 为活动或规则的触发事件集。

上述定义中,AS 表示模型的基本业务实体集,RuleS 表示易变的领域知识或逻辑,同时,RuleS 可以引用 AS 中的元素。

定义 2　活动集 AS = $\{a_i \mid i \in N\}$,a_i 为集合中一项活动,N 为自然数集。

在 a_i 中活动类型包括递阶型和一般型两种。其中:递阶型活动 a' 是指活动是粗粒度的,可细分为一个完整的或部分子 BPM4DBL 模型;一般型活动 a'' 是指活动是细粒度的,不可以再递阶。另外,一般型活动中引入虚活动 a_v,其不执行任何操作,只用于构造复杂活动之间的结构关系。

定义 3　递阶型活动 a' = {AS, RS, RuleS, ES} ∨ {AS, ES} ∨ {RuleS, ES} ∨ {AS, RS, ES} ∨ {AS, RuleS, ES}。其中:{AS, ES}、{RuleS, ES}、{AS, RS, ES} 和 {AS, RuleS, ES} 分别是 BPMDS 的子结构。

定义 4　一般型活动 a'' = {id, state, type, idata, odata, ioexp, impl, performer}。其中:id 为活动标志;type 为活动类型,分为 auto/manual 和 empty/invoke/receive/reply/user 两个聚类,分别用于表示企业内部和企业间活动;state 为活动状态,包括就绪、运行、取消、终止和完成;idata 和 odata 分别为输入和输出数据变量或常量;ioexp 为输入或输出数据变量的赋值表达式;impl 为活动的外部激活应用,属性包括 participant/interface/operation;performer 为活动的执行者。

从定义 3 可以看出,递阶型活动可以只包括活动集和事件集,如{AS, ES}和{AS, RS, ES}结构。其前者结构 AS 集中为单项活动,后者结构 AS 集中活动为偏序关系;同时也可以只包括规则集和事件集,如{RuleS, ES}结构。另外,{AS, RuleS, ES}结构 AS 集中为单项活动。从上述定义也可以看出,每一个结构中都包括事件集 ES,子结构的每次迁入和迁出都由活动的事件触发。由定义 3 可知,虚活动 a_v 中 type、state、idata、odata、iorules 和 performer 都为空。

定义 5　活动结构关系集 RS = $\{r_i \mid i \in N\}$。其中:r_1 和 r_2、r_3 和 r_4、r_5 和 r_6、r_7 和 r_8 分别为与分支/汇聚、或分支/汇聚、异或分支/汇聚、顺序/循环关系;r_9 和 r_{10} 为 *K-N* 分支/汇聚关系。

与分支/汇聚、或分支/汇聚、异或分支/汇聚、顺序/循环关系的定义可参见文献[5],它们称为活动之间的基本结构关系;*K-N* 分支/汇聚结构关系通过引入虚活动可以表达相邻多个活动之间复杂的依赖关系。*K-N* 分支结构关系中,*K* 指活动的相邻后继分支活动中有 *K* 个活动之间是与关系,其余 *N* 个分支中的活动与前述 *K* 个活动之间是或关系。*K-N* 汇聚结构关系中,*K* 指活动的前驱分支活动中有 *K* 个活动之间是与的关系,其余 *N* 个分支中的活动与前述 *K* 个活动之间是或关系。同时,对于活动的分支/汇聚结构关系中存在 *K* 个活动之间的异或结构关系,也归并为 *K-N* 分支/汇聚结构。

活动集合 AS 描述了业务过程中的基本实体对象,RS 描述了相邻实体对象间的结构依赖关系。对于 RS 中 r_3、r_4、r_5、r_6、r_8、r_9 和 r_{10} 结构关系来说,其需要控制变迁的触发或变迁触发的次序;同时,对于 AS 中非相邻实体对象之间的约束依赖,或者单个实体对象控制自身行为方面的约束,都需要通过规则集中的规则进行描述。

定义 6　规则集 RuleS = {StruRuleS ∪ ActioRuleS}。其中:StruRuleS 为结构规则子集;ActioRuleS 为行为规则子集;StruRule、ActioRule 分别表示集合中的结构规则和行为规则。

结构规则描述 r_3、r_4、r_5、r_6、r_8、r_9 和 r_{10} 中变迁触发控制条件,以及 AS 中非相邻实体对象之间的约束依赖关系,如规则“a_i 的输出值大于 0”表示变迁触发控制条件;行为规则描述

AS 单项活动中内部行为或外部资源的约束,如规则“a_i 的执行成员是其所属业务过程启动成员的直接负责人”。

定义 7　结构规则 StruRule = {term, fact}。其中:term 为术语集;fact 为事实集。

术语是业务过程中的基本实体对象、对象属性以及业务过程资源和外部引用的特定表达,如“a_i”和“a_i 的名称”为术语;事实表示术语之间的关系,即 fact = {Rl(term)},Rl 为术语之间的关系符号,其中基本关系包括比较、成员、属性、参与和继承关系,如“a_j 的输入数据是 a_i 的输出数据”为参与关系。另外,根据具体的领域应用可以扩展术语之间的关系。

定义 8　行为规则 ActioRule = {RuleWord(StruRule)}。其中 RuleWord 为规则词,基本的规则词包括如果 - 那么、必须、允许和禁止。

从定义 7 可以看出,行为规则通过规则词和结构规则构建,但其作用对象是 AS 中单项活动。行为规则分为条件、一致性和权属规则。条件规则定义给定条件下某种结果,规则词为如果 - 那么;一致性规则为永真断言,规则词为必须;权属规则定义某主体是否有某行为的权利,又可分为授权规则和禁止规则,授权规则定义主体有权(或被允许)进行某行为,规则词是允许;禁止规则定义主体无权(不允许)进行某行为,规则词是禁止。

定义 9　事件集 ES = {$event_i$ | $i \in N$}。其中:$event_i$ 为集合中单项事件。

定义 10　事件 event = {id, type, tr, ar}。其中:id 为事件标志;type 为事件类型,包括开始、结束和内部;tr 为各类事件的触发点。其中:开始事件触发点包括时间和消息;结束事件触发点包括消息、终止和异常;内部事件触发点包括时间、消息和异常;ar 为触发点的属性。其中:时间触发点属性包括日期和循环周期;消息触发点属性包括实现名称;错误触发点属性包括错误代码名称;终止事件点表示过程的所有实例结束,属性为空。

开始和结束类型的事件一般针对过程层,即每一过程的启动和终止由开始和结束事件表示;内部类型的事件针对具体活动或规则,即活动的状态变迁由相应事件点触发。业务过程执行时,开始事件首先触发流程执行,活动或规则的驱动由事件触发,同时在活动或规则中可以引入相对或绝对时间约束,时间约束的边界则作为时间事件的触发点。需要指出的是,此处规则的时间约束是指规则或规则集作为整体而被启动或执行的时间边界,而非规则内部与时间相关的内容。ES 为活动和规则的触发事件集,事件触发引起活动或规则状态的变化,同时,活动或规则执行又触发 ES 中相应的事件,如此往复直至业务过程执行结束。

在 BPM4DBL 模型中活动集 AS、活动间结构关系集 RS、规则集 RuleS 和事件集 ES 四类元素中,AS 和 RuleS 是 BPM4DBL 模型的实体构成元素。其中:AS 反映了领域应用中的基本实体对象;RuleS 反映了实体对象或实体对象间的业务逻辑。RS 和 ES 是 BPM4DBL 模型的连接元素。其中:RS 描述了领域应用中相邻活动之间结构上的控制依赖;ES 则定义了 AS 和 RuleS、RuleS 和 RS 相互连接的方式。在建立 BPM4DBL 模型时,如何根据具体的领域应用合理定义上述四类元素是构造 BPM4DBL 模型的关键。

2.2　BPM4DBL 概念模型中基本元素构建方法

BPM4DBL 概念模型中基本元素的构建遵循自顶向下、逐步细化的原则,步骤如下:a)识别领域应用中的顶层活动集。该集合中活动一般为递阶型,不包括虚活动。对顶层活动集中递阶型活动进行分解,直至其所有子活动均为原子活动为止,由此得到聚合程度不同的多层次活动集。b)建立各层活动集中活动之间结构上的控制依赖,即确定每一活动的前驱和后继活动,建立邻接和非邻接活动之间的结构规则,并可加入虚活动来满足复杂的结构约束;针对活动集中每项活动分析其内部行为或外部资源约束,构建活动的行为规则。c)对于每项活动和规则确定事件触发点,并且通过时间约束设定其时间触发事件点。

BPM4DBL 概念模型中的活动和规则对象分别是从整体和局部对领域应用的描述,体现了关注点分离原则。恰当的关注点则是模型保持动态性并且适应业务基础平台的关键。模型构建时应遵循以下准则:

a)粒度适中准则。对企业内部的领域应用来说,活动分解为由单个组织、角色或成员负责执行的粒度,而组织内部成员或角色之间的协作方式由规则来定义;对于企业之间的领域应用来说,活动一般以企业对外暴露的单项功能操作为粒度大小。

b)优先级准则。当通过活动粒度的进一步分解或者大粒度活动和规则结合均可以描述领域应用时,优先取后者;当通过引入新活动或增加新规则均可以描述领域应用时,优先取后者。

c)冲突准则。规则加入前,需要检测其与活动结构之间是否存在冲突,存在冲突的规则不被加入到规则集中。活动集中活动的名称、属性和数量修改时,需要检测其与规则集中的规则是否存在冲突,存在冲突的规则应被维护。

d)一致性准则。结构规则应与活动间的结构关系相一致,活动间结构变迁应该关联于恰当的结构规则。

根据上述构建过程和准则,可以确定领域应用的BPM4-DBL 概念模型中的基本元素。然而,业务基础平台中业务过程引擎需要通过驱动可执行的业务过程语言来得到过程的自动执行。同时为了支持引擎之间的互操作,执行层的语言应该遵循一致的标准。因此 BPM4DBL 概念模型需要映射到标准的可执行语言。

2.3　BPM4DBL 执行层语言

工作流管理联盟制定的 XPDL[7] 和 IBM 等制定的 BPEL4WS[8] 分别是面向企业内部和企业间应用事实上的业务过程标准执行语言,得到了研究界和工业界的广泛支持,同时,这两类语言的重点在于领域应用中宏观层面业务实体结构间关系的描述,而未显式给出微观层面领域知识或逻辑的表示。为此,通过对 XPDL 和 BPEL4W 进行扩展,本文将其分别作为 BPM4DBL 面向企业内部和企业间应用的执行层语言。下面给出 XPDL 和 BPEL4W 的扩展方式及具体映射规则。

2.3.1　BPM4DBL 面向企业内部应用的执行语言

XPDL 的基本语义元素包括 activity、transition 和 dataField 等,并且定义了各个元素的属性;同时各个基本元素提供了属性扩展机制。由此可以在不改变 XPDL 语义情况下,表达

BPM4DBL 中规则语义。

BPM4DBL 与 XPDL 之间映射的方法是:根据 BPM4DBL 概念模型中的基本元素,分析 XPDL 的语义相近元素,对语义相近元素进行映射;若 BPM4DBL 中的元素在 XPDL 中不存在相近语义元素,则将其映射到 XPDL 中基本元素的扩展属性或进行语义转换。表 1 为具体映射规则。

表 1　BPM4DBL 与 XPDL 的映射规则

BPM4DBL 元素	所映射的 XPDL 中元素
a''	a''与 activity 语义相近,但存在差异,两者不能进行直接映射,但其中属性值或元素的状态可以进行如下的映射
a''. state	映射为 activity。XPDL 中未显式给出状态概念,若将 activity 理解为活动的当前状态,则 a''中每一不同状态对应于 XPDL 中惟一的 activity
a''. type	映射为 activity 中的 startMode 和 finishMode,值为 automatic 或 manual
a''. idata a''. odata	映射为 dataField,其 dataType 由 a''. odata、a''. odata 的类型确定
a''. ioexp	映射为 dataField 中的 description。其中:description 中可以引用 a''. idata 和 a''. odata,并且根据其表达式确定 Script 元素的 Type 为 Java、Java Script 或 Python 等
a''. impl	属性 Interface 映射为 FormalParameter,属性 Operation 映射为 Application,属性 Participant 不作映射
a''. performer	映射为 activity 中的 performer。其中:description 中可以引用 a''. idata 和 a''. odata,并且根据其表达式确定 Script 元素的 Type 为 Java、JavaScript 或 Python 等
r_1、r_2、r_5、r_6	映射为 transition。其中:transition 属性元素 condition、description 和 extendedAttributes 设置为空;另外,还需要设置各结构关系中 activity 的 transitionRestriction 值。其中:r_1 和 r_5 的值为 split;r_2 和 r_6 的值为 join。同时,r_1 和 r_5 中 split 元素、r_2 和 r_6 中 join 元素的 type 属性分别设置为 AND 和 XOR
r_3、r_4、r_7、r_8	映射为 transition。其中:transition 属性元素 condition 非空,其值由结构规则确定(见下文);另外,需要设置结构关系中 activity 的 transitionRestriction 值。其中:r_3 和 r_7 的值为 split;r_4 和 r_8 的值为 join,并且 split 和 join 元素的 type 属性都为 XOR
StruRuleS	StruRuleS 中 term 类型为比较关系的规则映射为 transition 中的 condition,其余类型的规则映射为 activity 的 extendedAttribute。其中:activity 是 term 直接引用或者是属性相关的 activity
ActioRule	映射为 activity 的 extendedAttribute。其中:activity 是 StruRuleS 中 term 的直接引用或者是属性相关的 activity
event	映射为 deadline。XPDLSchema 中未包括显式的 event 语义,但通过定义活动和规则的各个事件点变量,以及由事件点变量设置 activity 中 deadline 的 deadlineCondition,可以通过 deadline 模拟活动和规则的事件

通过上述映射规则,可以将 BPM4DBL 中基本元素映射为 XPDL 中元素或其扩展属性。对于 BPM4DBL 中复合元素 a',首先将其映射为 XPDL 中的 blockActivity;其次递归分解 a'至基本元素;然后由上述映射规则得到 blockActivity 中各个元素。BPM4DBL 映射后可以得到 XPDL 的基本元素,但映射后 XPDL 中的 processHeader、redefinableHeader 和 version 等描述性属性可以根据实际情况单独设定或者保持为空。

2.3.2　BPM4DBL 面向企业间应用的业务过程执行语言

BPEL4WS 描述了多个 Web service 的复合方法,可以表达顺序、并行、同步和异步复合形式,包括 empty、invoke、receive、reply 基本活动元素,sequence、switch、while、pick 和 flow 结构活动元素以及 partners、variables 和 correlationSets 等上下文元素。

BPM4DBL 的规则元素与 BPEL4WS 中元素映射分为两种方法:a)将 BPM4DBL 中规则映射为 Web service,并且定义 message、portType 和 property 等 Web service 属性;b)在新命名空间中建立规则中各元素和属性的表达式,并且在 BPEL4WS 中通过扩展类型 extensibleElement 引用。第一种方法不引入新命名空间及其元素,简化了两者之间的映射,能以较大的粒度描述 BPM4DBL 中的规则;第二种方法便于通过建立适合规则元素和属性的表达式更加精细地刻画 BPM4DBL 中的规则,同时也增加了 BPEL4WS 模型的复杂度。另外,两种方法对业务过程执行环境也有不同的要求。本文采用第一种方法。表 2 给出了 BPM4DBL 中元素与 BPEL4WS 元素的映射规则。

表 2　BPM4DBL 与 BPEL4WS 的映射规则

BPM4DBL 元素	所映射的 BPEL4WS 中元素
a''	根据 a''. type 映射为不同的基本活动,具体内容见下文
a''. type = empty/invoke/receive/reply/User	分别映射为 empty/invoke/receive/reply/基本活动;各类型基本活动中 idata/odata 属性分别映射为上述基本活动的 inputVariable/output Variable 属性;各类型活动 impl 中的 Web service 属性 Participant/Interface/Operation 分别映射为 parternerLink/portType/operation 元素;活动属性 performer/ioexp 不进行映射
a''. state	不进行映射。BPEL4WS 中单个 Web service 没有状态概念,Web service 的异常或错误通过〈faulthandlers〉〈/ faulthandlers〉描述
r_1、r_2、r_3、r_4、r_5、r_6、r_7 和 r_8	映射为结构化活动〈flow〉〈links〉〈link〉activity〈/link〉〈/links〉〈/flow〉。其中:〈link〉定义了各分支结构中变迁的名称;activity 定义了与各变迁关联的基本活动或块活动,它们并行或者顺序执行。以 r_3、r_5 和 r_7 关系中变迁为起点的基本活动或块活动需要定义变迁条件,即〈source linkName = " ncnames" transitionCondition = " bool-expr" ?〉。其中:linkName 为变迁名;transitionCondition 为变迁条件。以 r_4、r_6 和 r_8 中变迁为终点的基本活动或块活动需要定义汇聚条件,即〈joinCondition = " bool-expr" ?〉Bool-expr 具体定义见下文
StruRule	StruRule 中 term 类型为比较关系的规则映射为 transitionCondition 或 joinCondition 中的 bool-expr,其余类型的规则映射见 ActioRule
ActioRule	映射为 Web service 定义中的各个属性,包括〈message〉、〈portType〉和〈plnk:partnerLinkType〉。其中 term 类型映射为〈message〉〈part〉〈/message〉。part 中 name 为 Term 术语名,type 为 term 的类型;规则词映射为〈portType〉〈operation〉〈input/〉〈output/〉〈/operation〉〈/portType〉。其中 operation 中 name 名为规则词名,input、output 为上述 message;另外,a'中定义的与活动之间关系映射为 plnk:partnerLinkType
event	开始事件映射为〈receive〉基本活动;结束事件的消息、终止、错误、补偿和空事件触发点分别映射为〈invoke〉、〈terminate〉、〈throw〉、〈compensate〉和〈empty〉基本活动;内部事件的消息触发点映射为〈receive〉基本活动,其余触发点的映射与结束事件相同

通过上述映射规则,可以将 BPM4DBL 中基本元素映射为 BPEL4WS 中基本活动和结构活动。对于 BPM4DBL 中复合元素 a',首先将其映射为〈scope〉块结构活动,其次递归分解 a'至基本元素,然后根据上述映射规则得到〈scope〉中各个元素。需要指出的是,BPEL4WS 文件中的 Web service 和 message properties 的定义需要在 WSDL 文件中描述,因此 BPM4DBL 中活动属性及映射应该生成 WSDL 文件;同时,BPEL4WS 本身具有相关的语义约束,如流程的初始活动类型必须为〈receive〉,并且其〈createInstance〉值应为“yes”等,映射后生成的流程文件应该符合上述约束。另外,BPM4DBL 模型映射后所生成的各个元素应当根据 BPEL4WS 语义定义合理设定其位置。

3　应用实例

在一个涉及跨部门的政府审批流程中,部门 1 在接收到审批请求后开始审批,若审批通过,则部门 2 和 3 并行审批,若审批通过后由部门 4 审批,部门 4 审批通过则流程结束。每个部门在一周之内未给出审批结果时,需要给出提醒通知;同时,部门 1 审批不通过则流程结束,部门 2 ~4 中任何部门审批不通过则返回部门 1 重复审批流程。另外,部门 2 的审批为子流程,部门 3 审批中涉及到易变的业务逻辑计算。

下面给出 BPM4DBL 对该流程的描述:AS 包括活动 a_1、a_2、a_3 和 a_4,分别表示部门 1 ~4 的审批活动。其中:a_2 为递阶

型活动;RS 包括 r_1 和 r_2 两类结构关系,即 a_1 与 a_2、a_3 之间为与分支关系,a_2、a_3 与 a_4 之间为与汇聚关系;RuleS 结构规则中的 Term 包括 $a_1.x$,$a_2.x$。其中:$a_1.x$ 为 a_1 的输出变量;$a_2.x$ 为 a_2 的输出变量。AS 活动之间变迁不存在触发控制条件,因此 AS 结构规则中的 Fact 为空。由于 a_3 中涉及到易变的业务逻辑计算通过行为规则表示,即 $ActioRule_1$:if $a_1.x <= A$ then $a_2.x = C$;$ActioRule_2$:if $a_1.x > A$ then $a_2.x = D$。ES 包括开始、结束和内部类型事件。其中 AS 中所有活动的内部事件时间触发点日期属性设置为一周,开始事件的触发点为消息,结束事件的触发点为空。为简化起见,a_2 的子流程不再具体说明。

由于是跨部门的审批流程,BPM4DBL 模型应该映射为 BPEL4WS。下面具体说明映射过程:

a)映射流程的开始事件。开始事件映射为〈receive〉元素,并由消息属性定义〈receive〉的属性。

b)映射活动集中的活动。根据活动的类型映射 a_1、a_2、a_3 和 a_4 为〈invoke〉元素,并且由活动属性定义〈invoke〉的属性;a_2 为递阶活动,所映射的〈invoke〉属性 name 为子流程名称。

c)映射活动间结构关系。r_1 和 r_2 映射为〈flow〉元素,并且定义〈flow〉中的〈link〉元素分别为 r_1 和 r_2 中的连接弧名称。

d)映射 $ActioRule_1$ 和 $ActioRule_2$。$ActioRule_1$ 和 $ActioRule_2$ 组成的规则集名 Ars 映射为〈input〉元素,Ars 的推理结果 Arr 映射为〈output〉元素,并且定义〈operation〉元素名为规则推理引擎名。

根据 BPEL4WS 语义设定映射后生成各个元素的位置,则该审批流程的 BPEL4WS 描述为

```
〈process name = "examineandapproveProcess"〉
〈sequence〉
〈receive partnerLink = "" portType = "" operation = "" variable = "" createInstance = "yes" /〉
〈invoke name = "" partnerLink = "" portType = "" operation = "" inputVariable = "a1x" outputVariable = "a1x" /〉
〈switch name = "Depar1Notion"〉
〈case condition = "bpws:getVariableProperty(a1x) = true"〉
〈flow〉〈links〉
  〈link name = "a1toa2" /〉
  〈link name = "a1toa3" /〉
  〈link name = "a2toa4" /〉
  〈link name = "a3toa4" /〉
〈/links〉
〈scope〉〈faultHandlers〉
〈catch faultName = "7days_Notify"〉
〈invoke name = ""〉〈/catch〉
〈/faultHandlers〉
〈eventHandlers〉
  〈onAlarm for = "tns:oneWeek"〉
  〈throw faultName = "7days_Notify"〉
  〈/onAlarm〉
〈/eventHandlers〉
〈invoke name = "Depa2Examine_Cycle_Process" partnerLink = "Internal" portType = "tns:processPort" operation = "call_Depa2Examine_Cycle_Process" inputVariable = "a2x" outputVariable = "a2x"〉
〈target linkName = "a1toa2" /〉
〈source linkName = "a2toa4" /〉
〈/invoke〉
〈invoke name = "Depa3Examine" partnerLink = "Internal" portType = "tns:processPort" operation = "ruleEngine" inputVariable = "Ars" outputVariable = "Arr"〉
〈target linkName = "a1toa3" /〉
〈source linkName = "a3toa4" /〉
〈/invoke〉
〈invoke name = "Depa4Examine" partnerLink = "Internal" portType = "tns:processPort" operation = "depa4Examine"〉
〈target linkName = "a2toa4" /〉
〈target linkName = "a3toa4" /〉
〈/invoke〉
〈/scope〉
〈/flow〉
〈/case〉
〈otherwise〉
〈empty/〉
〈/otherwise〉
〈/switch〉
〈/sequence〉
〈/process〉
```

需要说明的是,同时映射生成的还有对 Web service 定义的 WSDL 文件,如规则集名 Ars 和其推理结果 Arr 即在 WSDL 文件中定义,此处略去。

4　结束语

业务基础平台作为软件系统的开发环境,可以满足业务密集型领域应用中动态多变的业务逻辑,并且能提高软件系统的开发效率;而创建适合描述业务密集型领域应用中多变业务逻辑的业务过程模型,则是基于业务基础平台快速构建应用系统的关键。本文通过对业务密集型领域应用中业务过程特点的分析,定义了融合微观领域知识和宏观实体结构的业务过程模型 BPM4DBL 的元组表达,并且给出了模型的构建原则,从而适应领域应用中功能的动态性和结构的动态性;同时定义了 BPM4DBL 向 XPDL 和 BPEL4WS 的映射规则,从而使模型在执行层面遵循一致的标准,支持业务基础平台的互操作。如何用适当的形式化方式表示 BPM4DBL ,并且由此对其进行正确性、合理性和性能等分析,是下一步工作的重点。

参考文献:

[1] HELLER D,KRENZELOK L,ORR J. Webtop:Realities in designing a Web-application platform[C]//Proc of Conference on Designing for User Experiences. New York:ACM Press,2004:1-15.

[2] 计世资讯.中国软件平台产业发展战略研究[EB/OL].(2004). http://www.ccwresearch.com.cn/.

[3] AGUILAR-SAVEN R S. Business process modelling: Reviewand framework[J]. Internal Journal of Production Economics,2004,90(2):129-149.

[4] LINDSAY A,DOWNS D,LUNN K. Business processes-attempts to find a definition[J]. Information and Software Technology,2003,45(15):1015-1019.

[5] BERGHOLTZ M,JAYAWEERA P,JOHANNESSON P,*et al*. Process models and business models:a unified framework[C]//Lecture Notes in Computer Science. Berlin:Springer,2003:364-377.

[6] GORDIJN J,AKKERMANS H,VLIET H van. Business modelling is not process modelling[C]//Lecture Notes in Computer Science. Berlin:Springer,2000:40-51.

[7] Workflow Management Coalition. XML process definition language [EB/OL].(2005). http://www.wfmc.org.

[8] Business process execution language for Web services version 1.1 [EB/OL].(2003). http://www-128.ibm.com.