ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.9, September 2007, pp.2141−2152
DOI: 10.1360/jos182141

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于SPEM2XPDL模型转换的软件过程执行*

袁 峰[1,2], 李明树[1,3+]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100080)

Towards Software Process Enactment Based on the SPEM2XPDL Model Transformation

YUAN Feng[1,2], LI Ming-Shu[1,3+]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-10-62661800, E-mail: mingshu@iscas.ac.cn, http://www.iscas.ac.cn

Yuan F, Li MS. Towards software process enactment based on the SPEM2XPDL model transformation. *Journal of Software*, 2007,18(9):2141−2152. http://www.jos.org.cn/1000-9825/18/2141.htm

Abstract: SPEM (software process engineering metamodel) is the standard metamodel put forward by OMG (object management group) and widely accepted in industry. However, its model enactment is still at the early stage. In this paper, software process is viewed as a specialization of the more general kind of process known as workflow. SPEM2XPDL is presented as a software process enactment approach based on the model transformation. The well-defined mapping rules from SPEM to XPDL (XML process definition language) metamodels are provided and the corresponding transformation algorithm and engine are developed. This approach is implemented in the SoftPM project. After transformed to XPDL format, the SPEM models are successfully enacted by Shark, a famous XPDL engine.

Key words: software process enactment; SPEM(software process engineering metamodel); workflow; XPDL(XML process definition language)

摘 要: SPEM(software process engineering metamodel)是国际标准化组织制定的标准元模型,正日益成为软件过程建模领域的行业标准,但在过程执行方面,SPEM还存在不足.将软件过程看作是一种特殊的工作流,提出了一种应用工作流运行机制支持软件过程执行的方法.通过将SPEM模型转换为XPDL(XML process definition language)模型,利用XPDL引擎支持SPEM模型的执行.制定了SPEM和XPDL之间的映射规则,设计了转换算法并开发了转换引擎.该方法被应用在SoftPM项目中,成功地基于XPDL引擎Shark实现了对软件过程模型的执行支持.

关键词: 软件过程执行;SPEM(software process engineering metamodel);工作流;XPDL(XML process definition language)

* Supported by the National Natural Science Foundation of China under Grant No.60273026 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA116060 (国家高技术研究发展计划(863))

Received 2006-01-24; Accepted 2006-06-09

中图法分类号: TP311 文献标识码: A

近年来,软件过程的执行开始成为软件过程领域重要的研究课题.通过自动化任务分配、流程控制、产品管理等方式,软件过程执行可以减少人为错误的发生,从而提高生产效率,降低开发成本[1].SPEM(software process engineering metamodel)是软件过程建模领域的重要标准,它由国际标准化组织(Object Management Group,简写OMG)于 2002 年提出,并致力于将其发展成为软件过程领域的行业标准.自问世以来,SPEM已被广泛用于各种软件过程的建模,如IBM Rational的RUP、DMR的Macroscope以及Unisys的QuadCycle等.基于UML的扩展,SPEM具有优秀的建模能力和方法论无关等优点[2],但是,由于制定的初衷是用于过程描述,SPEM在对软件过程执行的支持方面还存在着不足[3].

20 世纪 90 年代以来,学者们对软件过程和工作流的共性和特点进行了深入研究.Chroust和Barnes等人指出,软件过程和工作流遵循同样的基本范式,它们关注一些共同的对象,如活动、角色、工作产品以及(各自不同的)业务规则,并且都利用工具来支持过程的建模、分析、执行、跟踪和度量[4,5].与软件过程相比,工作流的执行机制以及工具实现目前已经相当成熟.因此,基于工作流来支持软件过程的执行是一个可行而且有益的研究方向[5,6].

工作流是一类能够完全或者部分自动执行的业务过程[7].支持工作流执行的软件系统被称为WFMS(workflow management system),其中包括工作流的定义、管理以及工作流实例的执行.为实现不同WFMS产品之间的协同,工作流管理联盟(workflow management coalition,简写WfMC)定义了5个统一的接口,为WFMS中的各种功能提供了公共标准.其中,接口——XPDL(XML process definition language)是XML格式的过程定义交换语言,以支持工作流定义与执行的分离[8].符合XPDL规范的工作流模型可以被任何XPDL引擎执行.目前, XPDL引擎的发展已经相当成熟,商业软件如IBM的WebSphere MQ Workflow,开源软件如Shark[9]和Obe等都已有广泛的应用.

本文提出一种结合 SPEM 和 XPDL 标准支持软件过程执行的方法——SPEM2XPDL.通过将 SPEM 形式表示的软件过程转换为对应的 XPDL 模型,来实现软件过程基于 XPDL 引擎的执行.第 1 节介绍相关工作.第 2 节中对 SPEM 和 XPDL 的元模型作简单介绍.第 3 节中分 3 个部分介绍 SPEM2XPDL 方法,第 3.1 节给出从 SPEM 到 XPDL 的映射规则定义,并分析其中语义不一致的情况及影响;第 3.2 节和第 3.3 节分别给出 SPEM2XPDL 的转换算法和引擎实现.第 4 节介绍 SPEM2XPDL 方法在 SoftPM 项目中的应用实践情况,给出一个评审过程在 Shark 引擎上的执行实例.最后,总结本文的工作并展望下一步的研究方向.

1 相关工作

文献[5]中,Barnes 等人应用关系数据库和存储过程来作为软件过程管理系统,在数据库的 meta-schema 中定义了软件过程的核心概念,通过建立从 meta-schema 到 XPDL 的映射,实现基于 WFMS 的软件过程执行.文献[10]中,Chan 和 Leung 扩展了传统的工作流模型,在 WIDE 模型中针对软件过程的特点加强了精确的流程控制、动态过程演化以及对复杂过程结构的支持,然后用 WIDE 模型对软件过程进行建模,支持过程的执行.文献[11,12]中采用 UML 作为软件过程的建模语言,得到的过程模型分别被转换为 BPEL(business process execution language)和 XPDL 的形式以支持执行.由于 UML 并不是面向软件过程领域的特定建模语言,因此文献[11,12]中基于 UML 的 Profile 机制分别定义了不同的软件过程建模扩展.此外,文献[13]中以 WFMS-KAOS 为核心开发软件过程管理系统;文献[14]中基于工作流引擎 WebDeploy 搭建软件过程支持环境 PIEnviroment,这些都是在利用 WFMS 支持软件过程执行这个思路下的研究和应用.

在过去的研究中,一方面,正如Barnes等人所指出的,从软件过程到工作流的转换规则还有待进一步地细化定义,需要考虑更多的复杂情况,并在实际应用中不断完善[5].另一方面,以上所有研究中,无论是显式还是隐式地,都定义了不同的软件过程元模型.这些元模型和具体的过程方法论甚至建模环境绑定在一起,导致各自定义的从软件过程元模型到工作流之间的映射规则,以及转换引擎的实现都难以在不同的环境中重用.

在 SPEM2XPDL 方法中,我们选择 SPEM 作为软件过程建模的元模型,选择 XPDL 作为转换后工作流执行模型的标准,并定义了两个元模型之间详细的映射规则.SPEM 和 XPDL 都是国际标准化组织制定的行业标准,基于这两个标准,SPEM2XPDL 具有良好的重用性支持,另外还具有以下优点:1) 方法论无关性.应用 SPEM 作为软件过程的建模元模型,可以表达各种方法论指导的软件过程;2) 系统实现的开放性和灵活性,基于 XPDL 标准,其他的支持工具可以通过 WFMS 的接口定义方便地集成进来;同时,基于标准的实现也保证了系统的灵活性;3) 实现成本的极大降低.与完全重新构建新的系统相比,基于成熟的 XPDL 引擎进行开发极大地降低了实现的时间和工作量成本.

2 SPEM 和 XPDL 元模型

SPEM 是描述软件过程领域的特定建模语言,其中定义了各种软件过程领域的概念结构,因此具有简洁清晰、表现力强等特点;同时,SPEM 与过程方法论以及具体执行环境无关,不涉及任何过程执行的细节定义.而 XPDL 是支持工作流执行的定义语言,其中包含了与执行相关的各种概念结构.这些导致了两个元模型内容上的差异.

2.1 SPEM元模型

SPEM中核心的 3 个元类是角色(ProcessRole)、活动(Activity)和工作产品(WorkProduct)[2].针对软件过程的特定描述需求,SPEM围绕这 3 个元类定义了其他概念结构.如科目(discipline)和过程(process)用于过程元素的组织;生命周期(lifecycle)、阶段(phase)、迭代(iteration)用于过程生命周期的描述;Step用于进一步说明Activity的实现步骤.另外,SPEM中还给出约束条件的定义结构,如目的(goal)和后置条件(procondition);活动的输入输出产品可以通过活动参数(ActivityParameter)来进行说明.

此外,SPEM 还提供了丰富的关系定义.除组织各种元素的 Package 等名字空间关系之外,SPEM 中还定义了活动之间的前驱(precedes)关系,工作产品之间的跟踪(trace)、影响(impact)关系和分类(categorize)关系,工作产品和活动之间的指导(guide)关系,另外,还有与名字空间相关的引入(import)和涉及(referto)关系.

2.2 XPDL元模型

图 1 所示为XPDL的元模型[8],其中,主要元类包括:过程(process)、活动(activity)、参与者(participant)、应用程序声明(application declaration)、工作流相关数据(relevant data)和转移(transition)等.

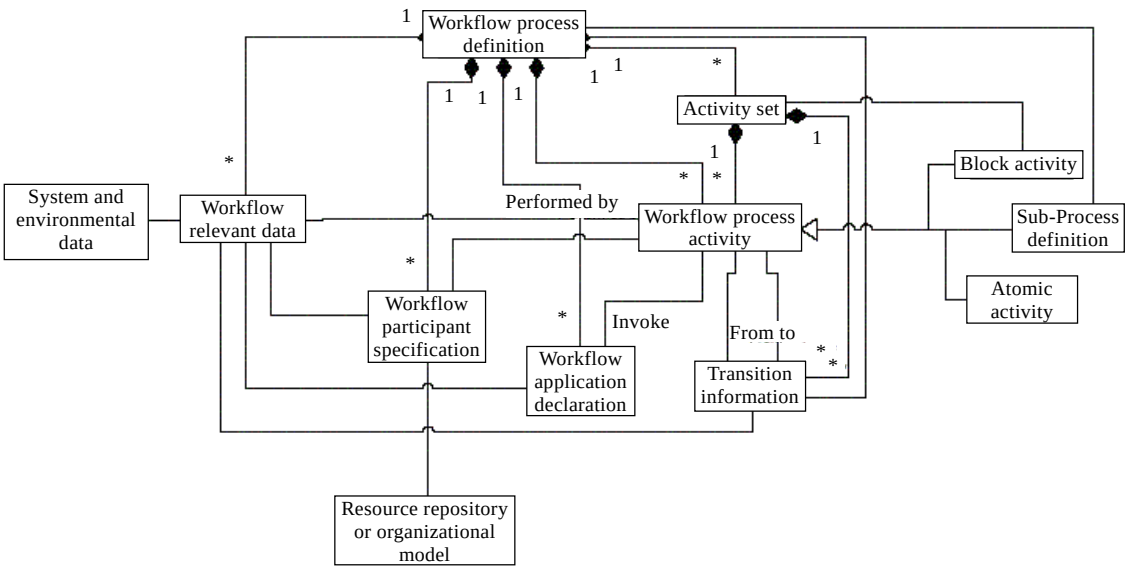

Fig.1 XPDL metamodel

图 1 XPDL 元模型

这些元类都面向于工作流的执行,同时,每个元类中都定义了各种执行相关的信息.以活动为例,它具有活动执行者、执行活动的应用程序、执行优先级等属性.

与 SPEM 相比,XPDL 中的概念结构更加通用和简单.以关系为例,XPDL 中只定义了一种活动之间的关系:Transition.另外需要注意的是,XPDL 中没有定义与工作产品对应的元类.

3 SPEM2XPDL 方法

为结合 SPEM 在软件过程建模方面的表达能力以及 XPDL 对过程执行的支持,SPEM2XPDL 方法定义了 SPEM 元模型到 XPDL 元模型之间的映射规则,通过将 SPEM 转换为对应的 XPDL 模型,支持软件过程的执行.方法内容主要包括 3 个部分:

1) SPEM 元模型到 XPDL 元模型之间的映射规则定义;
2) 基于以上规则,从 SPEM 模型到对应的 XPDL 模型的转换算法;
3) 对应转换算法的引擎实现.

3.1 SPEM2XPDL映射规则

映射规则分为两类:元素的映射规则和关系的映射规则.

3.1.1 元素的映射规则

表 1 中列出了 SPEM2XPDL 中主要的元素映射规则,这些规则覆盖了 SPEM 中的主要元类.

Table 1 The major element mapping rules in SPEM2XPDL

表 1 SPEM2XPDL 中主要的元素映射规则

No.	SPEM element	XPDL element	
[R1]	Package	Package	Different sources are distinguished by setting *ExtendedAttribute*, for example: ⟨ExtendedAttribute Name="type" Value="Phase"/⟩
[R2]	Discipline		
[R3]	Phase		
[R4]	Lifecycle		
[R5]	Process	Process	Transformed to *Process* with the same name.
[R6]	Iteration	Activity	*Iteration* is a composite WorkDefinition with a minor milestone. It is transformed to *Activity*, which implementation type is set to "subflow" and points to corresponding sub flow.
[R7]	Activity	Activity	An *Activity* with *Step* included is transformed to a *Block Activity* which points to an ActivitySet; An *Activity* without *Step* included is transformed to an *Activity* with the same name.
[R8]	Step	BlockActivity; ActivitySet; Activity	
[R9]	WorkProduct	ExtendedAttribute	Transformed to the extended attributes of corresponding *Activity*.
[R10]	WorkProductKind	ExtendedAttribute	Transformed to the lower extended attributes of *WorkProduct*.
[R11]	ProcessPerformer	Participant	Transformed to *Participant* which *ParticipantType* is "Role".
[R12]	ProcessRole	Participant & ExtendedAttribute	Transformed to *Participant*. In the mean time, add extended attribute "assistant" for corresponding *Activity*, the assistant points to this *Participant*.
[R13]	Guidance	ExtendedAttribute	Transformed to the extended attribute points to the *Activity*, for example: ⟨ExtendedAttribute Name="Guidance" Value="XXX"/⟩
[R14]	Goal	ExtendedAttribute	The same as *Guidance*.
[R15]	Precondition	Transition::Condition	The condition is set to the *Condition* attribute of corresponding *Transition*.

元类的属性在转换中也要映射为对应元类的对应属性.以 Activity 为例,其属性的映射规则见表 2.

可以看到,表 1 中多个 SPEM 元素对应到一个 XPDL 元素的情况出现了多次,这是因为这里的软件过程被看作是一种特殊的工作流.它导致的另一个结果就是转换过程中的部分语义丢失.SPEM2XPDL 中最主要的语

义丢失与工作产品相关.在 SPEM 中,工作产品、角色和活动是 3 个重要的核心概念;但在 XPDL 中,并没有直接对应工作产品的顶层元类,因此转换中会丢失相关的一些语义,包括:

1) WorkProduct 属性的一致性.在 SPEM 模型中,WorkProduct 的属性从属于 ID 唯一的 WorkProduct;而在转换到 XPDL 模型中后,可能对应到不同元素的属性.例如,某个 WorkProduct 的属性会同时转换为某个活动的扩展属性 InputWorkProduct 的扩展属性,以及另一个活动的扩展属性 OutputWorkProduct 的扩展属性.在转换后的 XPDL 模型中,这两个扩展属性的一致性需要额外的维护.

2) WorkProduct 与其他元类的关系.例如在 SPEM 模型中,WorkProduct 和 ProcessRole 之间的关联关系 WorkProduct::responsibleRole;WorkProduct 之间的 trace,impact 等关系.

3) 其他元类中与 WorkProduct 相关的属性.例如,WorkDefinition 的属性 hasWorkPerArtifact.

除此之外,SPEM2XPDL 转换中丢失的语义还包括 PrecedeKind,Goal,Guidance 等.需要指出的是,这里的语义丢失指的是在 XPDL 元模型没有对应的元类,从而相关语义无法在 XPDL 引擎中得到运行时支持.对此有如下处理方法:

1) 对于转换中丢失的大部分语义,由于并不影响转换后 XPDL 模型的执行,可不做处理.

2) 对于模型执行中希望支持的部分语义,可按自定义格式转换为 XPDL 中的扩展属性,并通过在 XPDL 引擎中集成对应的处理工具进行支持.例如,为了在过程执行中对工作产品进行管理,我们开发了符合 WfMC 标准的工作产品管理工具,读取和操作 XPDL 模型中对应工作产品的活动的相关扩展属性.

除了语义丢失之外,SPEM2XPDL 转换中也为 XPDL 模型增加了部分语义.见表 2,我们为部分属性定义了默认赋值.在 SPEM2XPDL 方法中这些都是允许的,因为方法的目的是提供一种支持基于 SPEM 建模的软件过程的执行方法,而并不是要得到与转换前 SPEM 模型语义完全一致的 XPDL 模型.

Table 2 The attribute mapping rules of the metaclass Activity

表 2 Activity 元类的属性映射规则

No.	Activity (SPEM)	Activity (XPDL)	
[RA-1]	Name	Name	Set to the same name.
[RA-2]	Performer	Performer	Set to the ID of corresponding *Participant*.
[RA-3]	Assistant	Extended attributes	Add extended attribute, such as: 〈ExtendedAttribute name="Assistant"〉 〈Participant Id="xxxx"〉 〈/ExtendedAttribute〉
[RA-4]	Step	BlockActivity; ActivitySet; Activity	Reference to Table 1 [R8];
[RA-5]	—	Id	Set to unified value during transformation.
[RA-6]	State	Description	Set corresponding attribute *Description*.
[RA-7]	—	Start mode	Set to "Manual" as default, because the activity executors in software engineering are normally human beings.
[RA-8]	—	Finish mode	The same as above, set to "Manual" as default.
[RA-9]	—	Implementation	Set to "NO" as default.
[RA-10]	—	Route	Not a route activity as default.
[RA-11]	—	Transition restrictions	The default transition restrictions are "AND Join" and "AND Split".
[RA-12]	—	Documentation, deadline, icon, limit, priority, simulation, information	All are execution related parameters, it is ok to not set.

3.1.2 关系的映射

在 XPDL 中,活动之间只有一种关系:Transition,而在 SPEM 中则存在诸如 Package,precedes,trace 等各种关系.按照转换的需要,我们将 SPEM 中的关系分为以下 4 类:

1) 层次关系(layer relationship,简称 LR)

我们根据层次关系来区分模型元素所处的不同名字空间(namespace).SPEM 中的层次关系包括 Package 及

Package 的子类 Process,Discipline 和 ModelElement 之间的 ownedElement 关联关系,以及 Activity 和 Step 之间的关系.在转换后的 XPDL 模型中,层次关系对应为 Package,subflow 关系或 ActivitySet 和 Activity 之间的关系.

定义 1. $LR=\{r|(r\in R)\wedge(r.oclIsKindOf(Association))\wedge P(r,CLR)\}$.$R$ 是 SPEM 模型中所有关系的全集,*oclIsKindOf*是对象约束语言OCL[15]中的关键字,$r.oclIsKindOf(Association)$用于判断$r$的类型是否为关联关系,谓词$P(r,CLR)$用于判断模型元素$r$是否满足约束条件CLR.

约束条件 CLR 用 OCL 描述如下:

[CLR] **context** *r* **inv**:

self. *associationEnd*→*size*=2

and

((*associationEnd*→*forAll*(*a*1,*a*2|*a*1.*oclIsKindOf*(*Package*) **implies** *a*2.*oclIsKindOf*(*ModelElement*)))

or ((*associationEnd*→*forAll*(*a*1,*a*2|*a*1.*oclIsKindOf*(*Activity*) **implies** *a*2.*oclIsKindOf*(*Step*)))

)

2) 角色-活动关系(role-activity relationship,简称 RAR)

SPEM 模型中角色和活动之间的关系包括:WorkDefinition 和 ProcessPerformer 之间的 Perform 关系以及 WorkDefinition 和 ProcessRole 之间的 Assistant 关系.转换后的 XPDL 模型中,根据 Perform 关系为每个活动指定参与者;对 Assistant 关系,则通过活动的扩展属性进行标注.

定义 2. *RAR*={*WorkDefinition*::*Performer*,*Activity*::*Assistant*}.

3) 转移关系(transition relationship,简称 TR)

同一层次活动之间的关系在转换之后对应于 XPDL 模型中的 Transition.转移关系包括 precede 关系,以及 SPEM 模型中常见的“活动-工作产品-活动”关系(activity-workproduct-activity relationship,简称 AWAR)组合.AWAR 关系通过输出/输入工作产品将两个活动连接起来.

定义 3. *AWAR*={(*a*1,*w*,*a*2)|*P*((*a*1,*w*,*a*2),*CAWAR*)}.

[CAWAR] **context** (*a*1,*w*,*a*2) **inv**:

(*a*1.*oclIsKindOf*(*Activity*))

and (*a*2.*oclIsKindOf*(*Activity*))

and (*w*.*oclIsKindOf*(*WorkProduct*))

and ((*a*1→*exitsts*(*OutputWorkProduct*=*w*)) **and** (*a*2→*exitsts* (*InputWorkProduct*=*w*)))

定义 4. *TR*={*Precedes*,*AWAR*}.

4) 其他关系(other relationship,简称 OTR)

除了以上关系,SPEM 中所有其他关系都被定义为 OTR.

定义 5. *OTR*={*impact*,*trace*,*referto*,*categorizes*,*govern*}.

由于 XPDL 模型中不同活动之间只有 Transition 关系,因此,OTR 在 SPEM 模型转换到 XPDL 模型时将会丢失.同样,如第 3.1.1 节中所述,这部分语义丢失并不影响对模型的执行支持.

3.2 SPEM2XPDL算法

算法的主要思想是:首先创建一个空的 XPDL 模型,其中,初始化默认的 Package Header 和 Redefinable Header 信息;然后创建一个列表 m_list,用以记录转换过程中的所有元素映射.该列表将用于后续步骤以及回滚;第 4 步,将所有的角色转换为对应的 Participant;第 5 步,创建空的 WorkflowProcesses;第 6 步,遍历 SPEM 模型中的关系,按照名字空间的不同层次组织所有的模型元素;在第 7 步中,转换位于同一名字空间的模型元素,第 7.1 步转换所有的 Activity,然后遍历其中的所有关系.根据第 3.1.2 节中的不同关系定义,触发不同的转换动作,并根据第 3.1.1 节中定义的映射规则转换对应的模型元素.第 7.4 步中,为每个名字空间对应的 ActivitySet 创建 Start/End 扩展属性,并增加需要的虚活动.由于受篇幅所限,虚活动的判断和增加算法本文不作详细介绍.第 8 步中,由转换人员手工将对应 SPEM 模型中的判断条件添加到转换后的 XPDL 模型中,最后进行过程合法性检查.

SPEM2XPDL 算法.

输入:SPEM 模型;

输出:XPDL 模型.

(1) 创建一个空的 XPDL 模型;

(2) 创建 Package Header 和 Redefinable Header;

(3) 创建 m_list; //创建映射列表

(4) 根据[R11][R12]转换所有的角色:{*process performers*,*process roles*}→{*Participant*}

(5) 创建空的 WorkflowProcesses;

(6) 遍历所有的关系 *r*

(6.1) If *r*∈*LR* //匹配{*Package*,*Discipline*,*Phase*,*Lifecycle*,*Process*,*Step*}

(6.1.1) 根据 namespace 组织 SPEM 元素;

(6.1.2) 遍历所有的 namespace

(6.1.2.1) If *r* 不是顶层 namespace,则在 ActivitySets 中创建对应的 ActivitySet;

(6.1.2.2) Goto (7);

(7) 遍历一个 namespace 中的所有元素

(7.1) 根据[R5][R6][R7][R8]转换所有的 *Activity*:{*Iteration*,*Activity*,*Step*}→{*Activity*};

(7.2) 遍历所有的关系 *r*

(7.2.1) if *r*∈*RAR*

(7.2.1.1) if *r*∈(*WorkDefinition*::*Performer*),设置对应 activity 的 Performer 属性;

(7.2.1.2) if *r*∈(*Activity*:*Asssitant*:),根据[R12]设置对应 activity 的扩展属性;

(7.2.2) if *r*∈*TR* //匹配{*precedes*,*ActivityParameters*{*kind*:*input*/*output*}}

(7.2.2.1) 创建 *Transitions*: {*TR*}→{*Transition*};

(7.2.3) if *r*∈*OTR*

(7.2.3.1) if *r*∈{*govern*},创建对应 activity 的扩展属性;

(7.3) 遍历所有的元素 *e*

(7.3.1) if *e*∈{*Guidance*,*Goal*,*Precondition*},根据[R13][R14][R15]设置对应 activity 的扩展属性;

(7.4) 增加 Start/End 扩展属性,以及虚活动;

(8) 手工添加约束条件;

(9) 过程合法性检查;

(10) 算法结束.

3.3 SPEM2XPDL引擎

根据上面的转换算法,我们开发了 SPEM2XPDL 转换引擎.该引擎读入 XML 文件形式的 SPEM 模型,转换后得到的 XPDL 模型也以 XML 文件的形式给出.

图 2 是 SPEM2XPDL 转换引擎的应用示意.首先,建模人员应用 SPEM 建模工具——如 Enterprise Architect——进行软件过程的建模工作,得到 XML 形式的模型文件.SPEM2XPDL 转换引擎读入该模型文件,根据映射规则进行转换,输出一个符合 XPDL 标准的工作流模型文件(这是以工作流模型形式表示的软件过程).它可以在支持 XPDL 的 WFMS,如 Shark 上执行.这样,转换引擎和 WFMS 的组合实际上充当了 SPEM 执行引擎的角色.

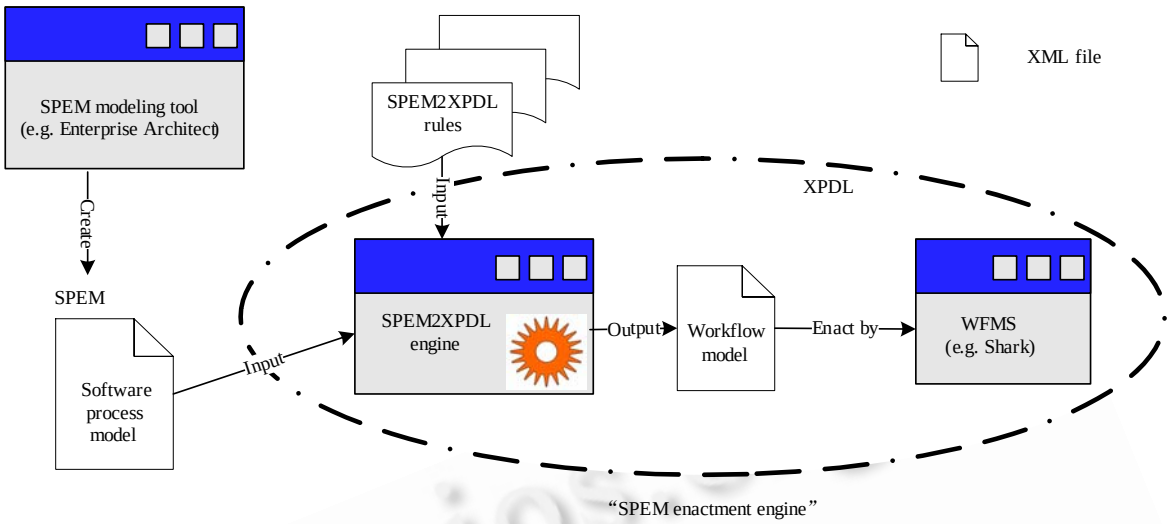

Fig.2 The illustration of SPEM2XPDL engine application

图 2 SPEM2XDPL 引擎的应用示意

4 应用实践

我们将SPEM2XPDL方法应用在中国科学院软件研究所的SoftPM项目中.SoftPM是基于CMM/CMM的软件质量管理平台[16],其中定义了一系列组织级软件过程,同时,这些过程也应用于SoftPM的开发指导.下面以其中的评审过程SoftPM_RV为例来介绍SPEM2XPDL方法的使用.

图3是在Enterprise Architecture中建模得到SoftPM_RV的部分视图.其中定义了4种角色:项目经理(project manager)、评审组长(review leader)、评审成员(review member)和产品组长(product leader).过程的核心部分为:首先,由项目经理负责,评审组长协助制定评审计划,然后由评审组长负责组织预评审(PreReview)和评审会议(review meeting),根据评审得到的问题记录,产品组长负责相关返工(rework)工作;最后,由评审组长负责跟踪问题解决情况.过程之间通过输出/输入的工作产品或者 precedes 关系直接连接.其中,活动 PreReview 通过嵌套的活动视图来进行更细致的定义说明.对于其中的每项活动,也可以定义执行的步骤,例如,独立评审(independent review)活动中定义了 3 个具体的步骤:确定评审时间表(confirm review schedule)、评审(review)和报告评审问题(report review question).

以上模型在 Enterprise Architecture 中导出为 XML 文件形式,经 SPEM2XPDL 引擎转换之后,得到对应的 XPDL 文件,附录中给出了对应 Reviewment Meeting 活动的 XPDL 文件内容片断,在 Shark 中读入后的执行视图如图 4 所示.其中,SPEM 模型中的子过程和步骤都被处理为 ActivitySet 的形式,所有的角色都被转换为对应的 Participant,并且,因 XPDL 的约束要求,增加了 Start,End 活动以及表达结构关系的虚活动.

在执行之前还需要进行一些配置工作:1) 定义用户和角色之间的映射.例如,将用户 Feng 映射为项目经理;2) 应用程序声明.在这里,声明自动调用的一些应用程序,同时也可以支持一些工具的集成.例如,在 SoftPM 中开发的工作产品管理工具和数据记录工具.后者用以记录用户每项任务的完成时间,通过应用程序声明集成到 Shark 中后,可以自动记录每个任务的实际工作量.配置完成之后,SoftPM_RV 模型的一个典型执行路径是:系统启动后,Shark 根据过程模型的定义进行条件判断,自动进行任务分配.不同角色登录后,会得到系统分配的任务列表,图 5 所示是项目经理 Feng 登录系统后得到的第 1 项任务 Make Review Plan.用户完成每项任务时,数据记录程序都会被自动调用,收集相关工作量数据.当 Feng 完成这个任务时,后续活动 PreReview 被启动,此时,如果评审组长登录系统,则会得到新的任务列表.

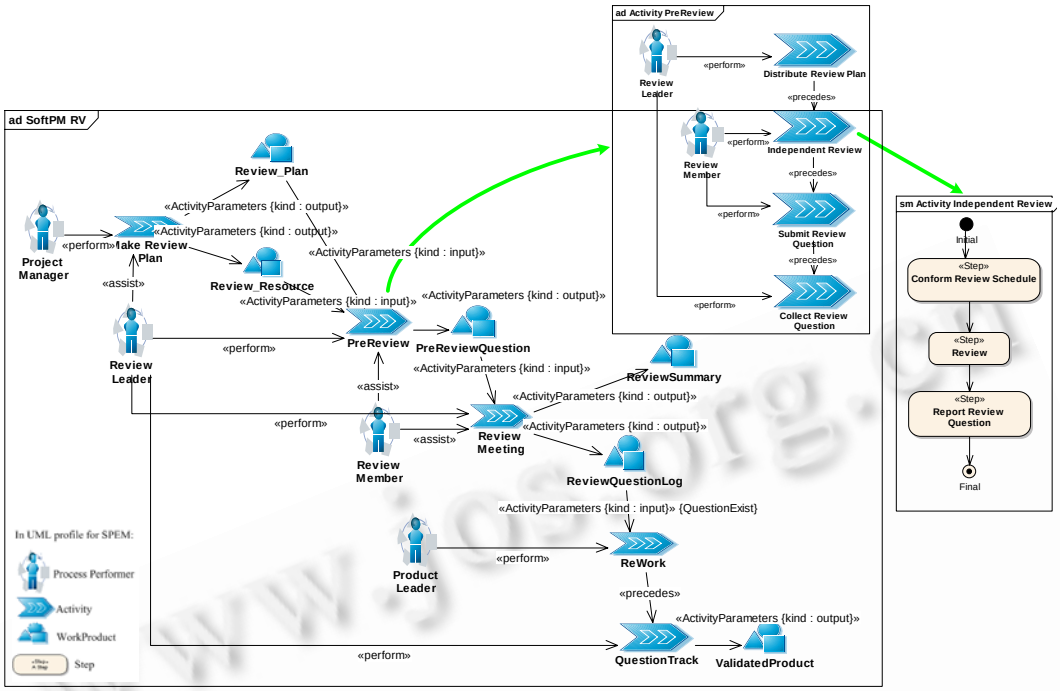

Fig.3 Process model SoftPM_RV (partial)

图 3 SoftPM_RV 过程模型(部分)

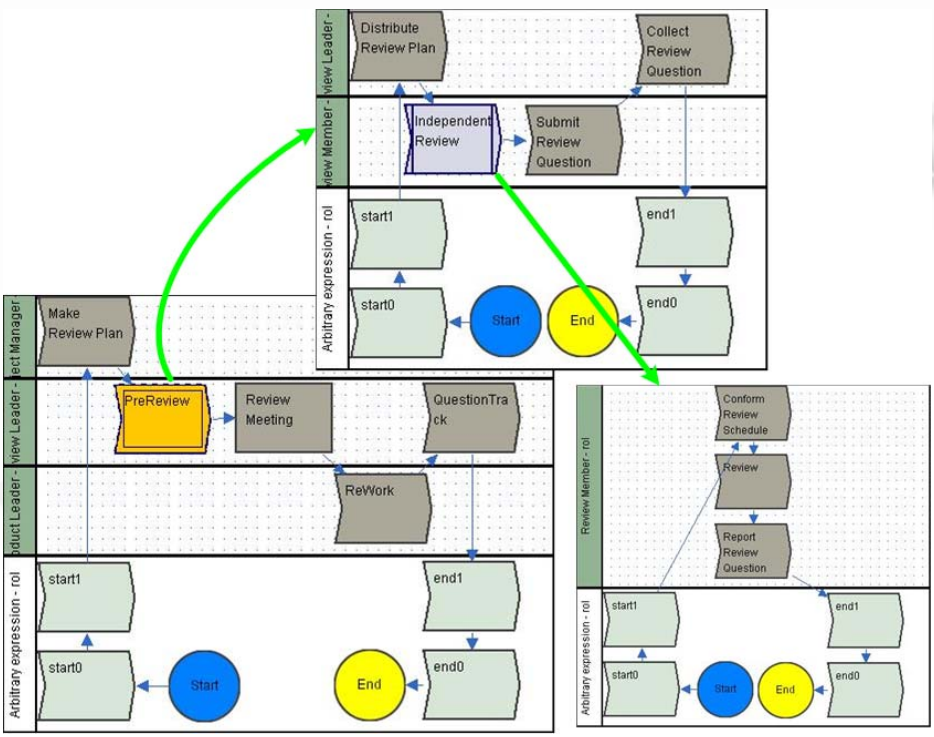

Fig.4 The execution view of SoftPM_RV in Shark

图 4 SoftPM_RV 在 Shark 中的执行视图

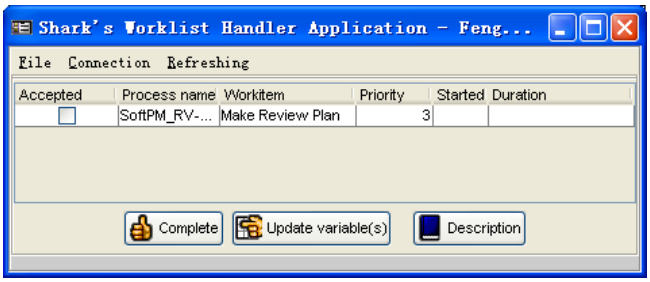

Fig.5 The task list in Shark

图 5 在 Shark 中的任务列表

可以看到,依靠工作流引擎的执行机制,软件过程的流程管理可以得到良好的自动化实现.同时,WfMC 的标准也为各种支持工具的集成提供了良好的接口,这为更加丰富的软件过程特定的管理支持提供了基础.上文中提到的工作产品管理工具和数据记录工具通过 WfMC 的标准接口和 Shark 很好地集成在一起,在 SoftPM 项目中,为软件过程的执行提供了有力的支持.除了 SoftPM 项目之外,我们也将 SPEM2XPDL 方法应用在了著名的过程模型 RUP 上,并取得了良好的效果.

5 结束语

本文提出了 SPEM2XPDL 方法,通过工作流的执行机制来支持软件过程的执行.我们定义了从 SPEM 元模型到 XPDL 模型的映射规则,设计了 SPEM2XPDL 转换算法,并开发了引擎实现.SPEM2XPDL 方法被应用在 SoftPM 项目中,在 XPDL 引擎 Shark 上成功地支持了 SPEM 模型的执行.

下一步的工作将集中在以下方向:

1) 在SPEM当前版本中没有明确地给出约束条件的定义规范.目前的SPEM2XPDL算法中相关的条件判断是手工添加的.定义中的SPEM2.0 计划采用OCL作为约束定义语言[3].因此,我们将致力于OCL表达式的解析和转换,以提高转换的自动化程度;

2) 为满足更多的软件过程特定的管理需求,将改进现有的工作产品管理工具和数据记录工具,同时开发更多其他工具,对软件过程执行中涉及的各个方面提供更好的支持;

3) SPEM2XPDL 方法也将被应用于更多、更复杂的情况.从中得到的实践经验一方面将用于方法的改进,另一方面,也将为 SPEM 新标准的制定提供参考.

References:

[1] Christie AM. Software Process Automation: The Technology and its Adoption. Berlin, New York: Springer-Verlag, 1995.

[2] OMG. Software process engineering metamodel specification. Version 1.1, 2005. http://www.omg.org/cgi-bin/doc?formal/2005-01-06

[3] OMG. Software process engineering metamodel (SPEM) 2.0 RFP. 2004. http://www.omg.org/cgi-bin/doc?formal/2004-11-04

[4] Chroust G. Interpretable process model for software development and workflow. In: Proc. of the European Workshop on Software Process Technology. LNCS 913, Springer-Verlag, 1995. 144−153.

[5] Barnes A, Gray J. COTS workflow and software process management: An exploration of software engineering tool development. In: Proc. of the 2000 Australian Software Engineering Conf. Gold Coast, 2000. 221−232. http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/6798/18243/00844579.pdf

[6] Chan DKC, Leung KRPH. Software development as a workflow process. In: Proc. of the ICSC. 1997. 282−291.

[7] The Workflow Management Coalition. The workflow reference model. WfMC-TC-1003, Version 1.1. 1995. http://www.wfmc.org/standards/docs/tc003v11-16.pdf

[8] Workflow Management Coalition. XML process definition language specification (Draft 1.0). Document Number WfMC-TC-1025. 2002. http://www.wfmc.org

[9] Shark. http://shark.enhydra.org

[10] Chan DKC, Leung KRPH. A workflow vista of the software process. In: Proc. of the Int'l Workshop on Database and Expert Systems Applications. IEEE Press, 1997.

[11] Keith M. From UML to BPEL, model driven architecture in a Web services world. 2003. http://www-128.ibm.com/developerworks/cn/webservices/wsuml2bpel/index.html

[12] Ping Jiang , Quentin Mair , Julian Newman. Using UML to design distributed collaborative workflows: From UML to XPDL. In: Proc. of the 12th Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. 2003. 71−76. http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/wetice/2003/1963/00/1963toc.xml&DOI=10.1109/ENABL.2003.1231385

[13] Penadés MC, Canós JH, Carsí JA. Workflow support to the software process. In: Proc. of the 4th MENHIR Workshop. Sedano, 1999. http://giro.infor.uva.es/workshop4/Presentaciones/Sesion2/Workflow.ppt

[14] Araujo RM, Borges MRS. Extending the software process culture—An approach based on groupware and workflow. In: Proc. of the 3rd Int'l Conf. on Product Focused Software Process Improvement PROFES 2001. LNCS, 2001. 297−310.

[15] OMG. OCL 2.0 final adopted specification. 2003. http://www.omg.org/cgi-bin/doc?formal/2003-10-14

[16] Wang Q, Li MS. Software process management practices in China. In: Proc. of the Software Process Workshop (SPW 2005). 2005.

附录:XPDL 模型内容片断

以下是 XPDL 模型形式的 SoftPM_RV 过程中关于活动 Reviewment Meeting 部分的内容片断:

```
〈Activity Id="EAID_CC6D544C_CDBF_4112_85CD_8E1A34783478"Name="Reviewment Meeting"〉
    〈Implementation〉
        〈No/〉
    〈/Implementation〉
    〈Performer〉EAID_5FD88815_8233_4b86_87B4_C573BDD41CA0〈/Performer〉
    〈StartMode〉
        〈Automatic/〉
    〈/StartMode〉
    〈FinishMode〉
        〈Automatic/〉
    〈/FinishMode〉
    〈TransitionRestrictions〉
        〈TransitionRestriction〉
        〈Join Type="AND"/〉
        〈Split Type="AND"〉
            〈TransitionRefs〉
            〈TransitionRef Id="id_Transition3"/〉
            〈/TransitionRefs〉
          〈/Split〉
          〈/TransitionRestriction〉
     〈/TransitionRestrictions〉
     〈ExtendedAttributes〉
       〈ExtendedAttribute
        name="ParticipantID"
```

```
  Value="EAID_5FD88815_8233_4b86_87B4_C573BDD41CA0"/>
 <ExtendedAttribute
  Name="assistId"
  Value="EAID_5FD88815_8233_4b86_87B4_C573BDD41CA0"/>
 <ExtendedAttribute Name="InputWorkProducts">
        <ExtendedAttribute Name="InputWorkProduct">
        <WorkProductName>Review_Plan</ WorkProductName>
        <Description>The Review Plan</Description>
        </ExtendedAttribute>
        <ExtendedAttribute Name="InputWorkProduct">
        <WorkProductName>Review_Resource</ WorkProductName>
        <Description>The Review Resource</Description>
        </ExtendedAttribute>
 </ExtendedAttribute>
 <ExtendedAttribute Name="OutputWorkProducts">
        <ExtendedAttribute Name="OutputWorkProduct">
        <WorkProductName>PreReview_Question</WorkProductName>
        <Description>The Question gathered in the prereview activity</Description>
        </ExtendedAttribute>
 </ExtendedAttribute>
</ExtendedAttributes>
</Activity>
```

袁峰(1977－),男,湖北咸宁人,博士生,主要研究领域为模型驱动架构,过程建模.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为智能软件工程,实时系统.