文章编号:1001 - 9081(2007)02 - 0409 - 04

支持服务协作的工作流元模型和建模语言

马 华[1],张红宇[2],李建华[3]
(1. 湖南涉外经济学院 计算机系, 湖南 长沙 410205; 2. 中南大学 商学院, 湖南 长沙 410083;
3. 中南大学 信息科学与工程学院, 湖南 长沙 410083)
(mohel. ma@163. com)

摘 要:当前支持 Web 服务的建模语言存在着复杂度高、互操作困难和缺乏对服务协作关系的有效支持等不足。结合面向服务工作流的服务协作需求,通过扩展工作流管理联盟的相关理论,定义了面向服务工作流的过程定义元模型,并建立了其形式化模型。通过扩展 XPDL 定义了一种互操作性好、支持服务协作的面向服务工作流建模语言(SoXPDL)。

关键词:工作流; 服务协作; 元模型; 建模语言
中图分类号: TP311 **文献标识码**:A

Meta model and modeling language for workflow supporting service collaboration

MA Hua[1], ZHANG Hong-yu[2], LI Jian-hua[3]
(1. *Department of Computer, Hunan College of International Economics, Changsha Hunan* 410205, *China*;
2. *School of Business, Central South University, Changsha Hunan* 410083, *China*;
3. *College of Information Science and Engineering, Central South University, Changsha Hunan* 410083, *China*)

Abstract: There are limitations in currently existing modeling language supporting web services, such as high complexity, poor interoperability and lack of effective support for service collaboration relationship. Process definition meta model of service-oriented workflow was presented on the basis of the analysis of service collaboration requirements, and the corresponding formal model was introduced. A model language for Service-oriented XML Process Definition Language (SoXPDL) supporting service collaboration was defined by extending XPDL.

Key words: service-oriented workflow; service collaboration; mete model; modeling language

面向服务计算环境下,面对分布式应用系统和资源的异构性、自治性和分布性,传统的工作流技术已经凸现出较大的局限性,如互操作困难;缺乏一个标准化的集成框架等等。Web 服务与工作流技术的结合,有助于克服传统工作流技术的不足,从而最终实现面向服务计算环境下动态、灵活的分布式应用集成。然而,现有的支持 Web 服务的工作流建模语言[1~4]存在着复杂度高、互操作困难和缺乏对服务协作关系的有效支持等不足。HeavenStarFlow[1] 中使用的建模语言 S-XPDL虽然实现了对 Web 服务的建模支持,但仅限于外部应用调用的层次上,仍无法满足面向服务工作流对服务协同的实际需求。文献[2]提出了一种确保面向服务工作流中交互式应用的健壮性的工作流建模语言和模型,但这种基于事件的编程模型无法满足业务过程建模的复杂需求。文献[3]设计的新的流程建模语言较为复杂,且存在互操作困难的问题。WfMC(Workflow Management Coalition)定义的 XPDL(XML Process Definition Language)[4] 也不具备对 Web 服务进行有效建模的能力,从而制约了面向服务工作流技术的发展。

1 工作流的服务协作需求

面向服务计算环境下,工作流中的服务已经成为了一种业务协作的单位,所以,面向服务工作流的过程建模必须要满足以下服务协作需求:1)异步活动。在面向服务计算环境下,业务流程中引用的某个 Web 服务可能是一个需要进行大量复杂的运算或者人工处理的周期很长的长事务服务,为避免过多地消耗本机资源,这类 Web 服务应该建模为异步方式调用。2)同步消息。面向服务计算环境下,多个业务伙伴之间的流程协作将日益频繁,同步消息的建模支持正是工作协同的需求。3)服务的事务管理与异常处理。Web 本身可靠性较低,一些对过程执行要求非常严格的场景势必要求服务的事务性保证,因此需要提供一定的异常处理策略。4)其他需求。包括:Web 服务的 QoS 要求,如:服务费用、执行时延、服务的可靠性和服务可信度等方面指标的要求;定义服务的安全策略;定义服务的并发控制机制。

2 面向服务工作流的建模

2.1 过程定义元模型

针对面向服务工作流的服务协作需求,WfMC 所定义的传统工作流过程定义元模型[5] 需要增加一些新的元素,并扩展原有部分元素的语义,如图 1 所示。

1)流程定义(Process Definition):描述业务流程的活动以及活动之间的控制流和数据流逻辑。在 SoWfMS 中,存在 public 和 private 两种访问级别的业务流程,public 型业务流程

收稿日期:2006 - 08 - 07;**修订日期**:2006 - 10 - 12

作者简介:马华(1979 -),男,湖南株洲人,硕士,主要研究方向:工作流技术、Web 服务、企业应用集成; 张红宇(1979 -),女,安徽亳州人,博士研究生,主要研究方向:虚拟企业、语义 Web、本体; 李建华(1963 -),男,湖南湘乡人,教授,主要研究方向:分布式计算、工作流技术.

可以被发布为一个公开的 Web 服务。

2)活动(Activity):是完成业务流程的一个逻辑步骤,在面向服务计算环境下,活动可能由一个 Web 服务来实现。

3)服务(Service):在逻辑上可能是某种原子操作,也可能是业务流程实体的封装。服务属于某个服务提供者,面向服务工作流中的服务提供者被作为合作伙伴,这体现了一种业务协作关系。同时,服务也将使用一些服务相关数据。

4)业务伙伴(Parnter):是 Internet 上与本流程所属组织存在业务协作关系的实体。它作为服务提供者,通过提供 Web 服务来参与当前的业务流程。

5)服务相关数据(Service Relevant Data):是用于约束 Web 服务的非功能性特性的数据,如服务费用、执行时延、服务的可靠性等,以及约束服务执行的参数,包括服务的同/异步调用类型、事务处理和异常处理策略等。

6)同步条件(Synchronization Conditions):定义活动开始和结束时所必需的消息约束。

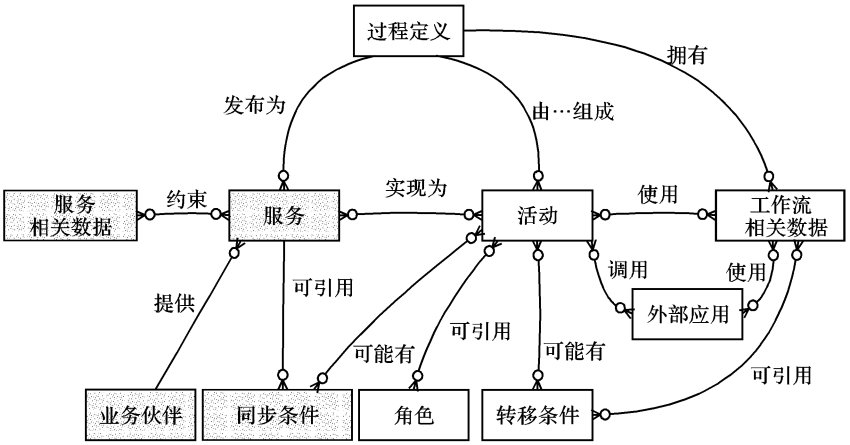

图 1 面向服务工作流的过程定义元模型

2.2 形式化建模

基于以上对面向服务工作流的过程定义元模型的分析,我们建立了相应的形式化模型:

定义 1 流程定义 p,是实际业务过程的数学描述,它是一个六元组 $< id, A, T, P, FP, C, Attr >$,$A = \{a_1, a_2, \cdots, a_n\}$,表示流程定义中所有活动构成的集合。$T$ 是所有活动间的转移的集合。FP 是形式参数集合。C 是流程容器,它是 p 中活动所使用的公共元素的集合,C 是一个五元组 $< PA, AP, RD, GS, PT >$,分别定义了资源/角色、应用、工作流相关数据、服务和业务伙伴的集合。$Attr$ 是五元组 $< name, accesslevel, version, author, created >$,它是流程属性集合,分别定义了 p 的名称、访问级别、版本号、创建者和创建时间。

定义 2 活动定义 a,表示业务过程中的一个实际或抽象的工作步骤,它是一个八元组 $< name, desc, exectrl, type, implementation, performer, transrestrict, CTX >$。$executectrl$ 是 a 的执行控制模式,决定活动以何种方式开始和结束。$type$ 是 a 的类型,可以定义为人工交互活动、自动化活动、路由活动、子流程活动和服务活动。$implementation$ 为 a 的实现方式。$ActP$ 是实际参数集合,$ActP = \{actp_1, actp_2, \cdots, actp_n\}$,$\forall actp_i(\exists rd_j(rd_j \in RD \wedge rd_j.name = actp_i))$。$performerper$ 是 a 的执行者,$\exists pa_i(pa_i \in PA \wedge pa_i.id = performer)$。$transitionrestriction$ 是 a 的转移限制,定义 a 的合并类型和分支类型。CTX 是活动上下文。

定义 3 转移 t,是一个五元组 $< id, a_{from}, a_{to}, type, exp >$,表示活动间的控制关系与数据流动关系。$a_{from}$ 是 t 的起始活动。a_{to} 是 t 的终止活动。$type$ 是转移的类型,可以实现条件分支的转移处理。exp 是转移条件表达式。

定义 4 Web 服务 ws,是一个七元组 $< id, pt, desc, location, operation, FP, gsrd >$,表示 p 中活动可能引用的服务。pt 指示了 ws 的提供者。$description$ 是对 ws 的语义描述。$location$ 是 ws 的 URL。$operation$ 指明了要调用的操作名。FP 的是形式参数集合。$wsrd$ 是一个四元组 $< execution, exception, issubtransaction, coordinator >$,它定义了服务相关数据。$execution$ 指示了 ws 的同/异步执行方式。$exception$ 定义了 ws 的异常处理策略。$issubtransaction$ 指示 ws 是否要求可靠事务保证。$coordinator$ 是服务协调器的 URL。

定义 5 业务伙伴 pt,是一个五元组 $< name, address, email, telephone, type >$,表示参与 p 的业务伙伴,分别定义了 pt 的名字、地理位置、联系邮箱、联系电话和服务类型。

3 面向服务工作流的建模语言

3.1 XPDL 的优点与不足

XPDL 是 WfMC 提出的一种基于 XML 的工作流过程定义语言[4],严格的说,XPDL 是不具备 Web 服务建模能力的。

1)过程控制结构 XPDL 的语言结构是有环有向图。XPDL 支持的过程控制结构包括:顺序、并发、选择、循环、时间控制和嵌套等。相对于 BPEL4WS 而言,XPDL 没有提供对基于消息的同步控制支持。同时这种依赖于“transition”的控制流机制无法象 BPEL4WS 一样显式地表达诸如 flow, while, switch 等结构。

2)活动建模 XPDL 是目前唯一一种显式地将用户交互引入业务流程建模的标准(BPEL4WS 等均不支持)。自动型活动调用的应用组件可以是 Web 服务。然而,XPDL 的自动型活动和应用还不足以对真实的消息流进行建模。XPDL 中的活动缺乏开始/结束条件的定义,所以 XPDL 中的所有自动型活动基本上都仅相当于 BPEL4WS 中的“invoke”类型。

3)事务及异常处理 XPDL 中不包含事务或异常处理的定义。虽然它提供了 exception 类型的 transition 的定义,但是仅支持对超时等简单异常的处理,不支持复杂的异常,而且额外定义的异常处理分支流程导致了流程的复杂化。BPEL4WS 中提供了多层次的异常定义和预定义异常类型,可以利用 fault 元素定义异常处理,它也提供了对捕获异常机制的支持。

4)人工交互 WfMC 主要关注人工型活动,所以定义了活动的执行者。XPDL 中,可以静态指定一个执行者,或通过一个组织结构和特定的业务规则在运行时动态地指定一个执行者。遗憾的是,这个概念基本上在 BPEL4WS 中已经消失了。

5)工作流仿真当流程定义变得复杂时,XPDL 提供了“simulation”元素来验证流程正确性。但是,目前基本上还没有工作流引擎实现了对这种内建仿真机制的支持。

总的来说,XPDL 侧重于业务流程描述。面向服务计算环境下,XPDL 已无法满足工作流中服务协作的各种需求,从而迫切需要对它进行一定的扩展,继承 XPDL 在业务过程建模方面的优点的同时,实现对分布式服务协作的支持。

3.2　SoXPDL 的定义

通过扩展 XPDL,我们定义了一种支持服务协作的面向服务工作流的建模语言(Service-oriented XML Process Definition Language,SoXPDL),它定义了新的“服务”元素,并对 XPDL 的活动实现方式和自动化模式等语法进行了扩展,提供了对异步服务、同步消息、服务事务和异常处理、服务质量参数等支持。

1)服务的定义

XPDL 定义了 5 种活动类型,包括:普通(人工型)活动、普通(自动型)活动、路由活动、块活动和子流程活动。SoXPDL 仍然是一种完全由活动和转移构成的基于条件化有向图的建模语言。所以,SoXPDL 并不新添加节点类型,而是扩展了 XPDL 中原有的活动类型,新定义了一个标记“Service”,它定义了:服务提供者、服务 URL、操作名、入/出口参数集、异常处理策略和事务执行方式等内容。为容纳“Service”元素,还要补充定义服务集。其他涉及的待扩展的标记包括:“WorkflowProcess”和“Package”等。

2)活动实现方式的扩展

XPDL 中仅路由活动和块活动定义了单独的标记,其余三种类型定义为活动的不同实现方式。默认为普通(人工型)活动,如果实现方式定义为“tool”,则对应为普通(自动型)活动,如果定义为“subflow”,则对应为子流程活动。XPDL 中简单地将 Web 服务作为普通(自动型)活动中一种被调用 application 来定义,这样无法满足服务协作的建模需求。所以,SoXPDL 定义了第三种活动的实现方式——Web 服务。

3)活动自动化模式的扩展

XPDL 通过自动化模式来定义启动和结束一个活动的自动化程度。目前,它定义了两种自动化模式[4]。其中,自动模式完全由工作流引擎进行控制,手工模式,要求用户明确地进行干预,促使活动的启动或完成。在面向服务计算环境下,以上两种自动化模式已经不能完全满足服务协作的需求了。为实现对异步活动和同步消息的支持,SoXPDL 定义了第三种活动的自动化模式——同步模式。

3.3　SoXPDL 的语法规范

通过定义 SoXPDL 的 XML Schema,我们给出了该建模语言的详细语法规范,以下是 SoXPDL 的关键语法定义。

1)服务(“Service”)的相关定义

“Service”定义的内容包括:业务伙伴、描述、服务 URL、操作名、入/出口参数集、同/异步执行方式、异常处理策略、事务执行方式、QoS 参数、协调器 URL 等,如图下代码所示。业务伙伴(Partner)的定义的内容包括:Id、Name、地址、联系方式(电子邮箱、联系电话)和提供的服务类型或所属行业。QoS 的定义内容包括:服务费用、执行时延、服务的可靠性和服务可信度。

SoXPDL 中服务的定义如下:

```
<xsd: element name = "Service" >
  <xsd: complexType >
    <xsd: sequence >
      <xsd: element ref = "xpdl: Partner" minOccurs = "0"/ >
      <xsd: element ref = "xpdl: Description"minOccurs = "0"/ >
      <xsd: element ref = "xpdl: Operation" minOccurs = "0"/ >
      <xsd: element ref = "xpdl: FormalParameters" minOccurs
      = "0"/ >
      <xsd: element ref = "xpdl: QoS" minOccurs = "0"/ >
      <xsd: element ref = "xpdl: Coordinator"minOccurs = "0"/ >
      < xsd: element ref = " xpdl: ExtendedAttributes "
      minOccurs = "0"/ >
    </xsd: sequence >
    <xsd: attribute name = "Id" type = "xsd: NMTOKEN" use = "
    required"/ >
    <xsd: attribute name = "Location" type = "xsd: anyURI" use
    = "required"/ >
    <xsd: attribute name = "Execution" >
      <xsd: simpleType >
        <xsd: restriction base = "xsd: NMTOKEN" >
          <xsd: enumeration value = "ASYNCHR"/ >
          <xsd: enumeration value = "SYNCHR"/ >
        </xsd: restriction >
      </xsd: simpleType >
    </xsd: attribute >
    <xsd: attribute name = "Exception" default = "FALSE" >
      <xsd: simpleType >
        <xsd: restriction base = "xsd: NMTOKEN" >
          <xsd: enumeration value = "CANCEL"/ >
          <xsd: enumeration value = "ABORT"/ >
          <xsd: enumeration value = "RETRY"/ >
          <xsd: enumeration value = "ALTERNATIVE"/ >
        </xsd: restriction >
      </xsd: simpleType >
    </xsd: attribute >
    <xsd: attribute name = "IsSubTransaction"
      default = "FALSE" >
    </xsd: attribute >
  </xsd: complexType >
</xsd: element >
```

2)扩展活动的实现方式

扩展后的活动实现定义如下:

```
<xsd: element name = "Implementation" >
  <xsd: complexType >
    <xsd: choice >
      <xsd: element ref = "xpdl: No"/ >
      <xsd: element ref = "xpdl: WebService"
      maxOccurs = "unbounded"/ >
      <xsd: element ref = "xpdl: Tool"
      maxOccurs = "unbounded"/ >
      <xsd: element ref = "xpdl: SubFlow"/ >
    </xsd: choice >
  </xsd: complexType >
</xsd: element >
```

3)扩展活动的自动化模式

扩展后活动启动与完成模式定义如下:

```
<xsd: element name = "StartMode" >
  <xsd: complexType >
    <xsd: choice >
      <xsd: element ref = "xpdl: Automatic"/ >
      <xsd: element ref = "xpdl: Manual"/ >
      <xsd: element ref = "xpdl: Synchronous"/ >
    </xsd: choice >
  </xsd: complexType >
</xsd: element >
```

4)其他新添或需扩展的语法

其他新添加的语法包括:Partners(Partner 元素的容器)和

Services(Service 元素的容器),以及 Operation、Coordinator 等简单数据类型的定义。另外,XPDL 语法中的 Package 和 WorkflowProcess 语法需要扩展对 Service 等元素的支持。

3.4 XPDL 与 SoXPDL 的语言特征对比

表 1 XPDL 与 SoXPDL 的语言特征对比

语言特征		XPDL	SoXPDL
图语言否		是	是
人工交互		支持	支持
工作流仿真		支持	支持
事务定义		支持(无显式语法)	同 XPDL
异常处理		很弱支持	同 XPDL
控制结构	顺序结构	支持(无显式语法)	同 XPDL
	并发结构	Split-And;Join-And	同 XPDL
	选择结构	Split-XOR;Join-XOR	同 XPDL
	循环结构	支持(无显式语法)	同 XPDL
	时间控制	Deadline 和 Limit	同 XPDL
	消息同步	不支持	支持(Synchronous 自动化)
Web 服务支持能力	支持方式	由 Tool 属性引用	支持(定义了 Service)
	集成 WSDL	不支持	支持(引用 PortType)
	同/异步支持	不支持	支持(Execution 属性)
	事务支持	不支持	支持(IsSubTransaction 属性)
	异常处理	不支持	支持(Exception 属性)
服务提供者定义		不支持	支持(定义了 Partner)
QoS 参数定义		不支持	支持(定义了 QoS)

XPDL 与 SoXPDL 的语言特点对比如表 1 所示。我们从该表可看到:SoXPDL 仍然是一种基于图的语言,同时它新增加了对消息同步控制的支持,另外,由于提供了与 WSDL 的集成、同/异步调用、事务和异常处理等支持,相对于 XPDL 而言,SoXPDL 对服务协作的支持能力增强。

4 结语

通过对面向服务计算环境下服务协作需求的分析,通过扩展 WfMC 的参考模型,提出了一个支持服务协作的面向服务流的过程定义元模型,在其形式化模型的基础上,通过扩展 XPDL 定义了一种面向服务工作流建模语言 SoXPDL,SoXPDL 不是一门全新的语言,由于保持了对 XPDL 的兼容性,所以,它不会带来 WfMS 之间互操作困难的问题。SoXPDL 满足了面向服务工作流的服务协作需求,支持定义异步服务、同步消息、服务事务管理与异常处理、服务质量参数等。目前它已经应用于基于面向服务工作流技术的电子政务协同系统中,满足了面向服务计算环境下动态、灵活的分布式应用集成中的工作流建模需求。

参考文献:

[1] WU ZH, SHUI G, DENG YL. Introducing EAI and service components into process management[A]. In: proceedings of the 2004 IEEE international conference on services computing[C]. 2004. 1-6.

[2] NEPAL S, FEKETE A, GREENFIELD P, *et al*. A service-oriented workflow language for robust interacting applications[A]. In: proceedings of the international conference on cooperative information systems[C]. 2005. 926-928.

[3] BERFIELD A, CHRYSANTHIS PK, *et al*. A scheme for integrating e-services in establishing virtual enterprises[A]. In: proceedings of the 12th international workshop on engineering e-commerce/e-business systems[C]. 2002.

[4] WFMC-TC-1025-2002. Workflow process definition interface——XML process definition language[R].

[5] WFMC-TC00-1003-1995. The reference model and API specification[R].

(上接第 405 页)

PDTool 的系统架构如图 2 所示,该图也反映了使用这种基于模板的过程定义工具的基本流程。

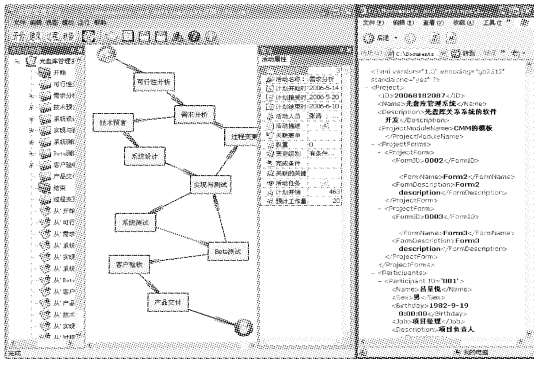

图 3 实验结果

系统的设计充分体现了面向对象和组件重用的思想。用户可以通过在图形界面上的点击和拖动实现流程的定义,然后再填写和选择某些数据,完成对整个过程的定义。图 3 是利用此过程定义工具对某一实际项目的开发流程的定义以及生成的 XML。

4 结语

本文提出了一种基于模板的过程定义语言。主要讨论了作为该语言基础的模板机制以及这种过程定义语言对已经存在的过程定义语言 WPDL 和 XPDL 某些部分的改进。一方面,基于模板机制的过程定义语言的使用非常简单。只要有了定义好的模板,剩下的过程就好比搭积木一样,降低了过程定义的复杂度。另一方面,由于使用了模板,将定义出来的过程中的业务数据和流程数据分离开来,使结构更加清晰。

当然,在这其中还存在着一些问题。比如关于过程中牵涉的某些数据到底应该放在模板中还是实际的过程定义中,还有关于某些情况下过程动态变更的控制等等。这些问题现在都还没有得到很好的解决,是下一步研究的主要方向。

参考文献:

[1] AMBLER SW, 王海鹏. 过程模式[M]. 北京: 人民邮电出版社, 2005.

[2] WAERN V. Co-Operative Process Management[Z]. Taylor & Francis, Feb 23, 1998.

[3] 罗运模. 软件过程改进与评估[M]. 北京: 电子工业出版社 2004.

[4] 余金山. 试论软件过程模型及其重要性[J]. 华侨大学学报, 1994, 15(1): 112-116.

[5] 柳军飞, 唐稚松. 软件过程建模语言研究[J]. 软件学报, 1996, 7(8): 449-457.

[6] 李军怀, 张彤, 张景, 等. 一种基于 XML 的工作流过程定义语言研究与应用[J]. 计算机工程, 2005, 31(15): 53-55.