

# 基于 MDA 的 Web 服务合成技术研究

党伟超,白尚旺

(太原科技大学 计算机科学与技术学院,山西 太原 030024)

**摘要:** Web 服务合成越来越引起人们的广泛关注。对基于 MDA 的 Web 服务合成技术进行了研究:描述了 Web 服务合成开发方法;提出利用 UML 类图对 Web 服务合成进行结构建模,对 UML 活动图进行行为建模,并给出 UML 类图和活动图到接口描述语言(WSDL)和业务流程执行语言(BPEL4WS)的转换规则;给出一个订单管理案例证明该方法的可行性。

**关键词:** Web 服务合成;MDA;UML

**中图分类号:** TP311

**文献标识码:** A

**文章编号:** 1673-629X(2007)06-0133-04

## Study of Web Services Composition Based on MDA

DANG Wei-chao, BAI Shang-wang

(School of Computer Sci. and Tech., Taiyuan Univ. of Sci. and Tech., Taiyuan 030024, China)

**Abstract:** There is a growing interest for Web services composition. Describes a model-driven Web services composition development approach: using UML class diagram to model structure PIM and UML activity diagram to model behavior PIM, then the conversion rules between UML PIM and PSM are introduced, by which the UML class diagram is converted to WSDL document and UML activity diagram is converted to BPEL4WS. At last the approach is illustrated with an example from a purchase order management scenario.

**Key words:** Web services composition;MDA;UML

## 0 引言

随着基于 XML 的 Web 服务标准如 SOAP, WSDL 等为广大研究人员和厂商所认同,出现了大量支持这些标准的产品和服务,因此如何有效地对这些 Web 服务进行重用、合成以组建新的、更复杂的 Web 服务组件,也自然成为人们进一步关注的热点问题。不同组织发表了各种 Web 服务合成语言:BPML, BPMN, BPEL4WS, BPSS, WSCI,但还没有一种 Web 服务合成语言成为事实上的标准。文献[1]对各种语言进行了深入比较并认为由于这些语言和产品的差异更增加 Web 服务合成的复杂性,提出要谨慎采用这些标准进行 Web 服务合成。

所有这些 Web 服务合成语言都基于可扩展标记语言(XML)。XML 是一种元标记语言,它使用简单

灵活的标准格式,为基于 Web 的程序提供了一个描述数据和交换数据的有效手段。从本质上讲,XML 只适合机器处理,对于不了解 XML 的人来说很难理解。因此利用上述语言进行 Web 服务合成就更加复杂。1997 年 OMG 制订了 UML1.1(目前最新标准为 UML2.0),经过几年的发展已经成为软件工业界事实上的标准,UML 作为一种可视化、统一的用于描述、构造软件系统的建模语言,得到了世界大多数软件厂商的支持。更进一步,2002 年,OMG 提出了模型驱动架构(MDA)这一新的软件系统开发方法,它强调整个系统开发过程由对软件系统的建模行为驱动,完成系统需求分析、架构设计、构建、测试、部署和运行维护工作。MDA 能够创建出机器可读和高度抽象的模型,然后通过转换(Transformation)技术把模型自动转换为代码、测试脚本、数据库定义以及各种平台部署描述。文献[2]对 MDA 在 B2B 中的应用进行研究,重点分析了各种元模型的转换机制;文献[3]描述了 MDA 在 Web 服务开发中的应用;文献[4~6]也提出利用 UML 对 Web 服务合成进行了建模,但都没有给出具体的实现方法。

文中就 MDA 在 Web 服务合成领域的应用进行研究,提出利用 UML 类图对 Web 服务合成组件进行结

收稿日期:2006-09-13

基金项目:山西省自然科学基金资助项目(20021047);太原科技大学青年基金资助项目(2005019)

作者简介:党伟超(1974-),男,山西芮城人,硕士,讲师,研究方向为数据库与软件工程技术、分布式计算、企业应用集成等;白尚旺,硕士,教授,研究方向为数据库与软件工程技术、分布式计算、网络计算等。

构建模, UML 活动图对 Web 服务合成过程进行行为建模。通过模型转换, 把 UML 类图转换成特定的 Web 服务描述语言如 WSDL, 活动图转换成特定的 Web 服务合成语言如 BPEL4WS。

## 1 Web 服务合成开发方法

Web 服务合成主要目的就是通过通过对已有的服务进行重用, 把多个服务合成为一个粒度更大、功能更复杂的服务组件, 最终的合成服务也要在 Internet 上发布, 并提供接口让基于 Web 的应用程序调用, 因此 Web 服务的合成开发需要解决两方面的问题: 其一就是接口的分析与建模; 其二就是如何实现接口所提供的功能, 在这方面和通常实现 Web 服务的方法不同, Web 服务合成组件的功能是通过调用其它的 Web 服务的相应操作而实现的。自然地, 可以通过 UML 类图解决上述第一个问题(具体描述见 2.1 节), 而第二个问题可以通过 UML 活动图来建模(见 2.2 节)。开发过程如下所述:

1) 用 UML 类图对 Web 服务合成组件的接口进行初步建模, 主要描述接口所提供的功能;

2) 从 Web 服务注册库中查找相关的原子服务;

3) 根据第 2 步找到的原子服务提供的功能对合成服务的接口进一步细化, 这一步的结果就是描述服务接口 PIM 的 UML 类图;

4) 利用模型转换把 UML 类图转换成某种特定的接口描述语言如 WSDL, IDL 或 Java 文件;

5) 用 UML 活动图对 Web 服务合成组件功能的内部流程进行描述, 重点是对第 2 步找到的基本 Web 服务所提供的操作进行分析, 把若干个基本操作通过特定的控制流、数据流组合成完成 Web 服务合成组件所提供的操作, 也就是 UML 接口类的每个操作都代表 UML 活动图中的一个流程, 流程正确执行就提供了操作所提供的功能;

6) 利用模型转换把 UML 活动图转换成某种特定的流程执行语言如 BPEL4WS;

7) 通过对应于特定流程执行语言的流程执行引擎运行第 6 步生成的流程执行配置文件来实现第 5 步所描述的功能;

8) 在相应的注册库中对新生成的 Web 服务合成组件进行注册, 这样其它的应用程序就能通过 Internet 对其进行调用, 其调用信息通过第 4 步生成的接口描述语言表达。

## 2 模型转换机制

上述步骤中主要思想就是对 Web 服务合成组件

的功能和行为进行 UML 建模, 然后利用模型转换机制把平台无关模型(PIM)转换成平台相关模型(PSM), 对于 UML 类图, 需要把它转换成特定的接口描述文件, 这样别的应用程序就能通过解析其中的信息, 完成对 Web 服务组件的调用, 由于 WSDL 在 Web 服务描述方面的广泛应用, 本节描述 UML 类图到 WSDL 的转换; 而把 UML 活动图转换成 BPEL4WS。

### 2.1 类图到 WSDL 的映射

UML 类图中主要元素包括: 类(class)描述拥有相同属性(attribute)和操作(operate)的一组对象; 类之间的关系用关联、聚合和组成描述; 类提供接口(interface)向外公开自己的操作, 类实现接口所提供的功能; 接口与类之间的关联用端口(port)来表达。

WSDL 规范<sup>[7]</sup>如图 1 所示, 主要由 6 个元素组成: definitions 是所有 WSDL 文档的根元素, 它定义 Web 服务的名称, 声明文档其它部分使用的多个名称空间, 并包含所有其它的元素; types 元素描述在客户端和服务端之间使用的所有数据类型; message 元素描述一个单向的请求或响应消息; porttype 元素结合多个 message 元素, 形成一个完整的单向或往返操作, 一个 porttype 可以定义多个操作; binding 元素描述了在 Internet 上实现服务的具体细节; service 元素定义调用指定服务的地址(URL)。

通过对 UML 类图和 WSDL 元数据的分析, 表 1 给出了 UML 类图元素到 WSDL 元素的映射关系。

|                                |
|--------------------------------|
| <definitions>: 根 WSDL 元素       |
| <types>: 传输哪种类型的数据?            |
| <message>: 传输什么消息?             |
| <porttype>: 支持什么操作(功能)?        |
| <binding>: 如何通过 Internet 传输消息? |
| <service>: 服务在哪里?              |

图 1 简洁的 WSDL 规范

表 1 UML 类图元素到 WSDL 元素的映射关系

| UML 类图元素  | WSDL 元素        |
|-----------|----------------|
| package   | definitions    |
| class     | complex type   |
| attribute | (part) element |
| operation | operation      |
| parameter | message(part)  |
| interface | porttype       |

### 2.2 活动图到 BPEL4WS 的映射

活动图用于显示组成复杂过程的步骤序列。活动图的主要元素有: 活动(activity)表示复杂过程的步骤, 通常是一个操作; 活动的执行顺序用控制流(control

flow) 描述,控制流的模式有顺序(sequence)、选择(choice)、同步(synchronization)、分叉(fork)和合并(merge)等;活动的输入和输出用数据对象(data object)表达;对活动图中的活动进行分区或责任分配称作泳道(swimlane)。

BPEL4WS<sup>[8]</sup>是基于 Web 服务的业务流程执行语言。通过与合作伙伴(partner)的交互定义 BPEL4WS 执行流程。合作伙伴可以将服务提供给流程(invoke),也可以向流程请求服务(receive),或者参与到流程的双向交互中。BPEL4WS 通过指定顺序来编排 Web 服务。BPEL4WS 还针对每个服务分配了合作伙伴的责任。

BPEL4WS 流程本身基本上类似于用来表达算法的流程图。流程的每一步称为一个活动。活动模式有以下几种:调用某个 Web 服务上的操作(<invoke>),外部服务请求流程(<receive>),生成输入/输出操作的响应(<reply>),等待一段时间(<wait>),把数据从一个地方复制到另一个地方(<assign>),指明某个地方出错了(<throw>),终止整个服务实例(<terminate>),或者什么也不做(<empty>)。通过结构化控制,可以将上述原语活动组合成更复杂的算法,这些算法表示了服务的实现。主要有:定义一组步骤的有序序列(<sequence>),使用“case-statement”产生分支(<switch>),定义一个循环(<while>),执行几条可选路径中的一条(<pick>),以及指明一组步骤应该并行地执行(<flow>)等。UML 活动图到 BPEL4WS 的转换规则如表 2 所示。

表 2 UML 活动图元素到

BPEL4WS 元素的映射关系

| UML 活动图元素                              | BPEL4WS 元素             |
|--|------------------------|
| activity graph                         | process                |
| swimlane                               | partner                |
| control flow                           | sequence and flow      |
| activity(<invoke>, <receive>, <reply>) | invoke, receive, relay |

### 3 一个 Web 服务合成案例

本节通过一个订单管理的案例说明上述 Web 服务合成组件的设计过程。该 Web 服务组件通过一系列与原子 Web 服务的交互完成订单管理的功能。其内部流程主要包括以下步骤:顾客代理向该服务(根据服务接口描述)提出订

单请求,该请求启动 Web 服务的内部流程,对获得的订单信息进行解析,同时触发三个并行的任务:计算订单价格、选择货运人和生成货运计划。当三个任务正常完成之后,顾客就会最终得到订单所请求的物品。

通过 UML 类图对该 Web 服务合成组件进行结构建模,向外提供接口(PurchaseOrderInterface),如图 2(左)所示,通过 2.1 所述的模型转换机制转换成如图 2(右)所示的接口描述文件(WSDL)。

同传统的 Web 服务开发技术不同,我们并不是一句句地从头开始写代码实现该接口功能,而是从 Web 服务注册库找到完成上述功能的原子服务,通过分析这些原子服务所提供的接口描述文件,定义与其端口的交互方式(在活动图中用 activity 表示),交互模式有 invoke, receive, relay, 其中 invoke 表示调用 Web 服务的操作, receive 表示 Web 服务向流程提出请求,而 relay 表示流程对请求的响应,活动图中通过 UML 扩展机制如 Stereotype 来区分不同的活动类型。对这些活动进行分区(或划分泳道)以区分不同的活动参与者,如图 3(左)所示,该案例共有三个原子服务(invoiceProvider, shippingProvider, schedulingProvider)。图 3(右)表示对应的 BPEL4WS 文件,三个原子服务表示为 Partner 元素,三个并行的控制流表示为 flow,每个分区(或泳道)的控制流被转换成 sequence,三种类型的 activity 分别转换成 invoke, receive 和 relay,三种模式都具有 partner, portType, operation 属性,通过给三者取值可以表示对伙伴某个端口特定操作的调用。

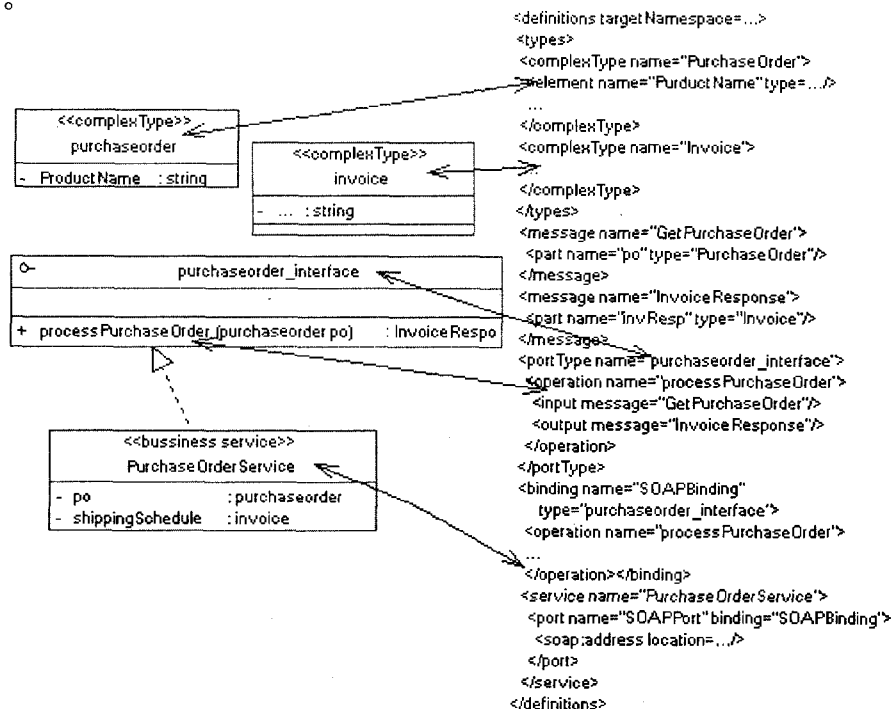
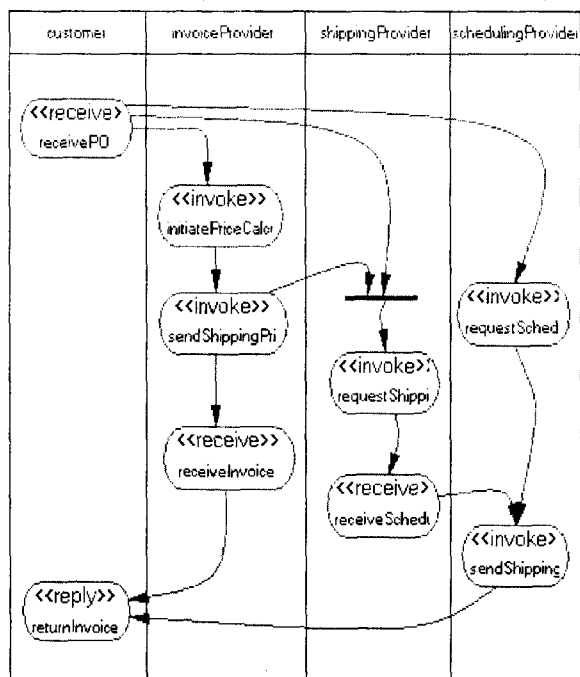


图 2 订单管理 UML 类图与对应的 WSDL 文件



```

<process name="purchaseOrderProcess">
  <partnerLinks>
    <partnerLink name="purchasing" .../>
    <partnerLink name="invoiceprovider" .../>
    <partnerLink name="shippingprovider" .../>
    <partnerLink name="schedulingprovider" .../>
  </partnerLinks>
  <variables>
    <variable name="PO" .../>
  </variables>
  <sequence>
    <receive partnerLink="purchasing" portType="purchaseOrderPT"
      operation="sendPurchaseOrder" variable="PO"> </receive>
    <flow>
      <sequence>
        <invoke partnerLink="shippingprovider" portType="shippingPT"
          operation="requestShipping" inputVariable="shippingRequest"
          outputVariable="shippingInfo">...</invoke>
        <receive partnerLink="shippingprovider" portType="shippingCallbackPT"
          operation="sendSchedule" variable="shippingSchedule"></receive>
      </sequence>
      <sequence>
        <invoke partnerLink="invoiceprovider" portType="computePricePT"
          operation="initiatePriceCalculation" inputVariable="PO"></invoke>
        <invoke partnerLink="invoiceprovider" portType="computePricePT"
          operation="sendShippingPrice"
          inputVariable="shippingInfo"></invoke>
        <receive partnerLink="invoiceprovider" portType="invoiceCallbackPT"
          operation="sendInvoice" variable="Invoice">
        </sequence>
        <sequence>
          <invoke partnerLink="schedulingprovider" portType="schedulingPT"
            operation="requestProductionScheduling" inputVariable="PO"></invoke>
          <invoke partnerLink="schedulingprovider" portType="schedulingPT"
            operation="sendShippingSchedule" inputVariable="shippingSchedule"></invoke>
        </sequence>
      </flow>
      <reply partnerLink="purchasing" portType="purchaseOrderPT"
        operation="sendPurchaseOrder" variable="Invoice">
    </sequence>
  </process>

```

图 3 订单管理 UML 活动图与  
对应的 BPEL4WS 文件

## 4 结 语

Web 服务技术已从基础设施的构建与概念推广阶段向大规模商业应用阶段快速发展。前者主要通过制定基于 XML 的 SOAP, WSDL 与 UDDI 等标准化通信协议与数据描述方式解决了 Web 服务定义、接口描述、服务查找以及松耦合异构环境下的远程调用与通信等基础问题;后者是解决在商业应用过程中所涉及的服务的重用与合成、安全、QoS 以及基于长事务的服务管理与调度等更为复杂的应用问题。文中就 MDA 在 Web 服务合成方面进行研究,从开发方法、模型转换规则以及案例分析等方面进行了全面描述,为 Web 服务的合成提供了新的思路。今后还需在以下方面进行更深入的研究:对目前存在的多种 Web 服务合成语言进行深入研究,总结出更通用的控制流模式;引入本体机制,解决 Web 服务合成过程中数据异构问题。

### 参考文献:

- [1] vd Aalst W M P. Don't go with the flow: Web services composition standards exposed [J]. IEEE Intelligent Systems, 2003, 18: 72-76.
- [2] Bezivin J, Hammoudi S, Lopes D, et al. Applying MDA approach to B2B applications: A road map [C]//Workshop on Model Driven Development (WMDD 2004) at ECOOP 2004. Oslo: Springer-Verlag, 2004.
- [3] Grønmo R, Skogan D, Solheim I, et al. Model-driven Web Services Development [C]//The 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (IEEE-04). Taipei, Taiwan: [s. n.], 2004.
- [4] Thöne S, Depke R, Engels G. Process-Oriented, Flexible Composition of Web Services with UML [C]//Int. Workshop on Conceptual Modeling Approaches for e-Business: A Web Service Perspective (eCOMO 2002). Tampere, Finland: [s. n.], 2002.
- [5] Grønmo R, Solheim I. Towards modeling web service composition in UML [C]//The 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure. Porto, Portugal: [s. n.], 2004.
- [6] Orriens B, Yang J, Papazoglou M P. Model Driven Service Composition [C]//ICSOC 2003. Heidelberg: Springer-Verlag, 2003: 75-90.
- [7] Christensen E, Curbera F, Meredith G, et al. Web Services Description Language (WSDL) 1.1 [EB/OL]. 2001-03-15. <http://www.w3.org/TR/wsdl>.
- [8] Andrews T, Curbera F, Dholakia H, et al. BPEL4WS 1.1 [EB/OL]. 2003-05-05. <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>.