面向 Web Services 的模型驱动开发方法*)

于笑丰　胡　军　李宣东　郑国梁
(南京大学计算机软件新技术国家重点实验室　南京 210093)
(南京大学计算机科学与技术系　南京 210093)

摘　要　随着分布式对象技术的发展和电子商务应用的扩大，Web Services 技术应运而生。由于在解决异构软件的交互和企业系统集成问题上表现了极大潜力，因此学术界和工业界对 Web Services 都备加关注。MDA 是 OMG 提出的用于解决不同中间件系统交互和集成问题的新的软件开发方法，是目前软件工程领域最引人注目的研究热点。本文阐述了 Web Services 和 MDA 的基本概念，对二者的交叉研究进行了分析和综述，提出了面向 Web Services 的模型驱动开发框架，并对未来工作做了展望。

关键词　Web Services，模型驱动开发，MDA

A Web Services -oriented Model Driven Development Method

YU Xiao-Feng　HU Jun　LI Xuan-Dong　ZHENG Guo-Liang
(State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　With the development of distributed object computing, Web Services is born. Web Services has gained focus from both academe and industry for its enabling of heterogenous software interaction and system integration. MDA is a new software development method initiated by OMG, which aims to solve the problems of software interoperability and integration. MDA is the most hottest issue in software engineering at present. This paper presents a survey on various aspects of Web Services and MDA from basic conceptions to intercross research topics. This paper also presents our Web Services oriented model driven development framework MDDF4WS and outlines some future research activities.

Keywords　Web services, Model driven development, MDA

1　引言

随着分布式对象技术的发展和电子商务应用的扩大，Web Services 技术应运而生。分布式对象技术是随着网络和面向对象技术的发展而兴起的。通过使用中间件机制，分布式对象技术能够屏蔽操作系统平台、网络协议和网络硬件平台的异构性，从而使系统可以适应异构的网络计算环境。在 Web Services 出现之前，已经有很多基于 RPC 的组件技术，如 DCOM、CORBA、EJB 等，但这些组件技术都要求交互的双方采用明确的、同类型基本构架的具体对象模型协议。由于这些特定的对象模型协议都是与厂商相关的，因此在无法严格受控的复杂、异构网络环境下，这些传统的分布式对象技术依然无法解决系统的交互和集成问题。另一方面，随着电子商务的迅猛发展，对异构系统之间的灵活交互和集成有着迫切的要求，为此传统的分布式对象技术必须进一步发展，而 Web Services 则是这一发展的产物。Web Services 是基于开放的、标准的网络协议（如 HTTP、SOAP）和数据格式（如 XML）。XML 提供了跨平台的数据编码和组织形式；SOAP 基于 XML 并且绑定于 HTTP 之上，支持跨语言和操作系统进行远程过程调用，实现了编程语言和系统平台的无关性。借助于 XML 和 SOAP，Web Services 可以为不同企业系统之间的交互提供跨语言、跨平台的解决方案，所有支持这些标准协议的系统都可以无缝地通信和共享数据。

Web Services 作为新的分布式计算技术已经备受学术界和工业界广泛关注。目前，针对 Web Services 的研究课题也很多，比如 Web Services 相关标准的制定、Web Services 体系结构、Web Services 发现、Web Services 建模、Web Services 应用开发框架、SOA 和 Web Services、语意 Web Services、Web Services 本体论、Web Services 组合、Web Services 的资源管理、Web Services 在电子商务中的应用、Web Services 和工作流、Web Services 应用中的事务处理、安全和隐私问题、Web Services 性能等[1~3]。

Web Services 是从软件体系结构的角度来解决异构系统的交互和集成问题，而 MDA[4] 则是 OMG（Object Management Group）从软件开发方法学的角度给出的解决方案。为了解决不同中间件系统地集成和交互问题，使系统具有良好的可交互性、可扩展性、能方便地集成现有系统和未来新技术、保护现有系统的投资，OMG 基于现有的标准（MOF，UML 以及 CWM）提出了新的软件开发方法 MDA。

MDA 的本质是分层建模，其中与具体实现平台无关的模型称为 PIM（Platform Independent Model），与平台相关的模型称为 PSM（Platform Specific Model）。这样便将系统的

*)本文的研究工作受到国家自然科学基金（批准号 60203009，60233020）、江苏省自然科学基金（批准号 BK2003408）和国家 973 项目（批准号 2002CB312001）的资助。于笑丰　博士生，主要研究方向：软件工程、模型驱动转换方法与验证；胡　军　博士生，主要研究方向：软件工程、嵌入式软件建模、形式化方法；李宣东　教授，博士生导师，主要研究方向：软件工程、形式化方法、模型检验；郑国梁　教授，博士生导师，主要研究方向：软件工程、形式化方法。

功能规约和系统功能在特定平台上的实现规约相分离，增强了系统的可移植性。MDA 的关键是实现 PIM 到 PSM 的自动映射。同一 PIM 可以映射为不同平台的 PSM，并且这些 PSM 可以通过 PSM 桥(PSM Bridge)相联系，因此可实现不同平台的互操作，同时也提高了软件复用。除此之外，MDA 还可以提高系统的开发速度和可验证性，以及降低人为引入的错误。

虽然考虑问题的角度不同，但是 Web Services 与 MDA 是殊途同归的，它们的目标都是解决异构系统的互操作和集成问题，因此我们认为把 Web Services 和 MDA 结合起来，将成为解决这一问题的有利手段。

本文第 2 节概括了 Web Services 的基本概念、基本架构，从不同角度描述了 Web Services 的定义、特征和当前的研究热点；第 3 节概述了 MDA 的基本思想以及 MDA 相对于传统软件开发方法的优点；第 4 节对 MDA 和 Web Services 交叉研究进行了分析和综述；第 5 节提出了面向 Web Services 的模型驱动开发框架；最后总结全文，并对下一步研究工作进行了展望。

2 Web Services 基本概念

2.1 Web Services 架构

Web Services 的架构是我们理解 Web Services 的基础，它从面向服务的角度给出了 Web Services 的描述。如图 1 所示，Web Services 的架构是建立在三种角色的交互基础上的，其中服务请求者(Service Requester)、服务提供者(Service Provider)和服务代理(Service Broker)是三种角色。三种角色的交互行为包括发布、查找和绑定。这三种角色和三种操作构成了 Web Services 的基本构架：Web Services 软件模块和 Web Services 描述。服务提供者拥有可以通过网络访问的 Web Services 实现，同时还为它拥有的每一个 Web Service 定义了服务描述，并将服务描述发布到服务代理的一个目录上。当服务请求者需要调用某个 Web Service 时，首先要通过查找操作，到服务代理的目录中去搜索相应的 Web Service 描述信息，然后根据得到的描述信息去调用服务提供者发布的服务。服务请求者和服务提供者的角色都是逻辑结构上的，两个角色之间是可以互相转换的，有时一个服务可能既是请求者又是提供者。

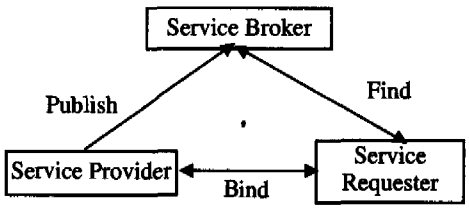

图 1 Web Services 架构

为了真正实现平台无关的通信和交互，Web Services 体系结构采用了一系列平台无关的标准和协议来实现相关的功能。采用 WSDL(Web Service Description Language)来描述服务，采用 UDDI(Universal Description、Discovery and Integration)来发布、查找服务，采用 SOAP(Simple Object Access Protocol)为 Web services 的调用提供标准的 RPC 方法[5]。

XML 是 Web Service 的核心技术，为 Web Services 提供了统一的数据格式，参与交互的角色之间以及角色内部的信息都是用 XML 格式进行传递的。XML 是 World Wide Web Consortium (W3C)提出的用于不同的计算机应用之间交互数据的标准，为数据的传递提供了标准的、平台无关的语法，同时为所传递的数据提供了定义和校验功能。XML 文档具有跨平台和松散耦合的特点，并且有很好的可扩展性[6]。正是结合了 XML，Web Services 协议标准才提供了交互和平台无关通信的广泛实现[7]。XML 为 Web Services 体系结构提供了核心的标准基础，WSDL，UDDI 和 SOAP 都是建立在 XML 之上的。

WSDL 是一种基于 XML 的语言，它提供了标准的方式来描述 Web Services 调用的输入参数、输出参数、返回值，以及函数的结构和传输所用的协议等[8]。WSDL 类似于 CORBA 或 COM 中的 IDL，是用于描述编程接口的。WSDL 独立于 Web Services 实现所用的语言和组件模型，只专注于接口的抽象描述。WSDL 文档可用于动态发布 Web Services、查找并且绑定已发布的 Web Services。通过 WSDL 文档，客户就可以得到足够的信息来调用相应的 Web Services。

UDDI 是新一代的基于 Internet 的电子商务技术标准，提供了在 Web 上描述和发现服务的框架。它包含一组基于 Web 的、分布式的、Web 服务信息注册中心的实现标准，同时包含一组使企业能将自己提供的 Web Services 注册到信息注册中心以便其他商业实体能够迅速发现的访问协议的实现标准[9]。UDDI 标准定义了 Web 服务的注册、发布与发现的方法，它的作用有点类似全球电子黄页。

SOAP 是用于交换 XML 编码信息的轻量级协议，提供了标准的 RPC 方法来调用 Web Services。SOAP 定义了一个简单的对象，用来以 XML 格式封装服务请求和应答的消息[10]。SOAP 可以用来交换任何类型的 XML 消息，因此，为 Web Services 使用者和提供者之间的服务请求和应答的消息传递提供了基本的标准。SOAP 是基于 XML 的，XML 是 SOAP 的数据编码方式。

2.2 Web Services 的定义

目前对 Web Services 的定义还没有达成共识，比较有代表性的是以下几种从不同角度给出的 Web Services 定义。

从服务请求者的角度，文[11]认为 Web Service 是描述一些操作的接口，接口中的操作是可以通过网络用标准的 XML 消息传递进行访问的。Web 服务是用标准的、规范的 XML 概念描述的，这一描述囊括了与服务交互需要的全部细节，包括消息格式、传输协议和位置。该接口隐藏了实现服务的细节，允许独立于实现服务基于的硬件或软件平台和编写服务所用的编程语言使用服务。从特征的角度文[7]给出了 Web Services 的定义，认为 Web Services 是自包含的、模块化的商业应用，有开放的、面向 Internet 的 、标准的接口。Web Services 是松散耦合的，采用标准技术通过 Internet 与其它 Web Services 进行直接通信。从模型的角度，文[12]给出了 Web Services 的定义，认为 Web Services 是一个描述了 Web 应用下一步演化的模型，这个模型具有如下的特征：事务由 Web 上的其它程序发起；服务可以在 Web 上动态地描述、发布、发现和调用；在应用对应用(application to application)的层次上进行通信；模型与实现语言和平台无关。从构架和实现的角度，文[13]中给出的 Web Services 定义是：Web Services 是一个软件系统，有一个以 WSDL 描述的接口，与其它系统是以 SOAP 消息进行交互的，它的目的是为网络上可互操作的机器与机器之间(machine-to-machine)的交互提供支持。

2.3 Web Services 的特征

虽然学术界对 Web Services 还没有一致的定义，但 Web Services 表现出来的一些基本特征还是得到了公认 。

跨平台的交互和集成：由于 Web Services 使用开放的标准协议进行描述、传输和交换，完全屏蔽了不同系统平台的差异，所以允许业务功能位于完全不同的系统中，并且以标准的方式进行通信。

良好的封装性：Web Services 功能的实现对服务请求者是不可见的，服务请求者能且仅能看到由 WSDL 描述的该 Web Services 的功能列表和参数模式。

松散耦合：Web Services 通信是通过消息传递实现的。由 WSDL 描述的 Web Services 接口给环境提供了一个抽象层，这使得环境与 Web Services 之间的连接富有弹性，适应性强。当一个 Web Services 的实现发生变更的时候，只要描述服务的接口不变，服务实现的任何变更对于服务请求者都是透明的。

2.4 Web Services 的相关研究

做为新的分布式计算技术，Web Services 融合了 Web 和基于组件开发的最佳之处[14]，为解决异构软件的交互和企业系统集成提供了比以往更好的解决方案。但当前的 Web Services 还处于发展的初级阶段，还存在很多问题亟待解决和探讨。工业界正在从规范和协议的标准化方面来完善 Web Services，例如 W3C 制定了 WSDL，UDDI，SOAP 等规范和协议，IBM 与 Microsoft 联合制定了 BPEL 规范等。与工业界相比，学术界更加关注 Web Services 相应技术的实现。目前，学术界针对 Web Services 的研究课题非常广泛，例如 Web Services 发现、Web Services 建模、Web Services 应用开发框架、SOA 和 Web Services、语意 Web Services、Web Services 本体论、Web Services 组合、Web Services 和工作流、Web Services 应用中的事务处理、安全和隐私问题、Web Services 性能等。限于篇幅，本文仅对 Web Services 和 MDA 结合研究进行了分析和综述，并提出了基于 EDOC 的 Web Services 模型驱动开发框架。

3 MDA

随着计算技术不断发展，新技术层出不穷，中间件便是软件领域近 10 年来发展起来的一种新兴技术。由于不同中间件的异构性和开放性各有不同，以及不同中间件厂商的利益纷争等因素，中间件技术难以形成统一的标准。因此随着中间件技术的不断发展，以及中间件平台数量的不断增加，采用不同中间件平台的系统之间的交互和集成，甚至采用相同中间件平台的不同版本的新旧系统之间的集成和系统的演化都面临巨大的挑战。除此之外，如何使已有系统方便地采用或集成未来的新技术，以便保护已有系统的投资，都成为亟待解决的重大问题。

为了解决基于不同中间件系统的交互和集成问题，OMG 提出了一种新的软件开发方法 MDA[4]。MDA 的目标是通过采用厂商中立的技术来促进软件系统和组件的集成。MDA 支持应用领域中不断演化的标准，以及不断增加的不同种类的中间件。MDA 致力于完整的生命周期：设计、实施、集成、管理应用以及使用开放标准的数据。基于 MDA 标准使用户能够将现有的和以后的系统集成在一起。

MDA 的主要思想是将系统的功能规约和系统功能在特定平台上的实现规约相分离，从而将技术与平台变化对系统的影响降低到最小程度。MDA 将系统模型分为 PIM(Platform Independent Model)和 PSM(Platform Specific Model)。MDA 的核心是首先将 PIM 抽象出来，然后针对不同实现技术与平台制订多个映射规则，再通过实现这些映射规则的转换工具将 PIM 转换成 PSM，最后将 PSM 不断求精直至形成实现代码。

和传统的软件开发方法相比，支持 MDA 的开发工具对大型软件系统的开发有很多改善：

提高了软件开发速度[4]。一方面，绝大多数开发人员只需把精力集中在 PIM 的开发上，而不用关心与具体平台相关的细节；另一方面，由于焦点由代码转移到 PIM，开发人员可以更多地关注如何解决业务问题，从而可以开发出与用户的要求更加吻合的系统。在文[15]中给出的 J2EE PetStore 开发实例，采用 MDA 工具开发和采用非 MDA 工具开发相比，开发速度可以提高 35%。

提高了软件的可移植性[16]。处于 PIM 层的任何模型都是完全可移植的，因为 PIM 是平台无关的。

提高了软件的互操作性[16]。同一 PIM 向不同平台映射产生的 PSM，可以通过 PSM 桥(PSM Bridge)相联系，进而可以将一个平台的概念映射成另一平台的概念，实现互操作性。

提高了软件的可复用性[17]。同一个 PIM 可以向不同的平台做映射，从而产生基于不同平台的 PSM。

避免了一些手工开发容易引入的错误[15]。

提高了系统的可验证性。PIM 中没有与具体平台相关的细节，因此易于对其进行模型检验，以验证其正确性。

4 MDA 和 Web Services

基于 MDA 的软件开发过程需要解决的关键技术问题是 PIM 到 PSM 转换的自动化。只有解决了这个问题，MDA 给软件开发带来的众多益处才能得以实现。由于 Web Services 是正在崛起的新一代分布式计算技术，为解决分布式异构平台环境中的系统交互、系统集成和系统互操问题提供了强有力的技术支持，因此将 MDA 和 Web Services 相结合，研究基于 MDA 的 PIM 到 Web Services 模型的转换方法和面向 Web Services 的模型驱动开发框架是非常有价值的。

4.1 基于 MDA 的 UML PIM 到 Web Services 模型的转换

由于 UML 和 MDA 都是 OMG 提出的规范，而且 UML 已经成为被广泛接受的、平台无关的建模语言，研究用 UML 描述的 PIM 模型到 PSM 模型的转换便成为一个研究热点。

从面向开发的角度，文[18]指出开发 Web Services 应用应该首先从 WSDL 和 WXS(W3C XML Schema)入手，而不是先用某种语言实现 Web Services，然后从实现生成 WSDL 和 SOAP 代码。但 WSDL 既不是广为所知的语言，也不足以简单到可以作为设计语言来用。所以真正的 Web Services 开发过程应该是 UML→WSDL→实现，而不是 WSDL→实现[19]。文[19]指出，Web Services 开发从用 UML 为系统构建 PIM 开始，然后通过构造型(stereotype)机制向 UML 模型中加入表示 Web Services 的语意信息，使之成为 Web Services 模型，最后用某种程序语言实现 Web Services 模型。文[20]从元模型的角度提出了 UML 元模型到 WSDL 元模型的转换算法。

WSDL 是用来描述 Web Services 静态行为的规范，BPEL4WS(Business Process Execution Language for Web Services)是由 IBM 和 Microsoft 联合提出的描述 Web Serv-

ices 工作流的规范。从描述 Web Services 动态行为和交互的角度，文[21]扩展了 UML，提出了针对自动化业务过程的 UML profile(UML Profile for Automated Business Processes)，此 Profile 中包含与 BPEL4WS 相对应的语意结构；此外还给出了用此 Profile 建立的 PIM 到 BPEL4WS PSM 的映射算法，该映射算法能自动生成 Web Services 的构件(如 BPEL，WSDL 和 XSD)。

4.2 基于 MDA 的 EDOC PIM 到 Web Services 模型的转换

EDOC(Enterprise Distributed Object Computing)是 ISO RM-ODP(Reference Model for Open Distributed Processing)运用于企业规模的系统所得到的规范，它提供了一套递归协作的建模方法。该方法可用于不同粒度层次和不同耦合度层次的业务及系统的建模。EDOC 能够同时支持松散连接和紧密连接系统的规约，同时支持容器管理的体系结构与基于信息的体系结构中的同步通信和异步通信[22]。EDOC Profile (UML Profile for EDOC)通过一个建模框架能够简化基于组件的 EDOC 系统的开发，这个建模框架是基于 UML1.4 并且符合 MDA 规范的。OMG 已经在 2001 年把 EDOC Profile 采纳为 Internet 计算、集成 Web Services、消息传递、ebXML 以及.NET 和其他技术的建模框架。EDOC Profile 主要包括 ECA(Enterprise Collaboration Architecture)和模式[23]。EDOC 的核心部分——ECA 是用于规约 EDOC 系统的 MDA 方法，包括 5 大元素：CCA(Component Collaboration Architecture)、Entities profile、Events profile、Business Process profile 和 Relationships profile。ECA 的 5 个元素联合起来，分别从 5 个不同侧面，也即 RM-ODP 定义的 5 个视点(企业视点、信息视点、计算视点、工程视点和技术视点)对一个分布式的计算系统进行刻画。ECA 关注的焦点是企业规约，计算规约、信息规约。这些规约构建了一个 EDOC 系统的平台无关模型 PIM。使用从技术相关模型得到的技术概念，软件设计者可以将这些规约进一步转化成平台相关模型 PSM 的工程规约和技术规约。

根据语意等价的原则，文[24]提出了用 YATL(Yet Another Language)描述的 EDOC 到 Web Services 的转换。该转换分成两个阶段：第一阶段完成数据类型的转换，即从 EDOC 模型文档(Model Document)到 Web Services XML Schema 的转换，即 EDOC 文档模型中的 DataType、ComopositeData 和 Attribute 分别转换为 XML Schema 中的 SimpleType、ComplexType 和 Attribute；第二阶段完成操作的转换，即 EDOC CCA 到 Web Services WSDL 的转换：EDOC 文档模型中的 FlowPort、OperationPort、ProtocolPort、ProcessComponent 分别转换为 WSDL 中的 Message、Operation、PortType 和 Service。文[20]从元模型的角度提出了 CCA 元模型到 WSDL 元模型的转换算法。着眼于动态行为，文[25]提出通过扩展 Web Services 元模型来实现 EDOC 动态行为到 BPEL 的转换。

4.3 模型驱动的 Web Services 组合

随着可用的 Web Services 数量的增加，无论从重用的角度，还是从降低成本、提高开发效率的角度考虑，组合已有的 Web Services，生成新的 Web Services，都成为 Web Services 研究的重要分支。

因为运行在异构系统中不同平台之上的 Web Services 可能是以不同的语言实现、由不同供应商提供的，而且 Web Services 要根据特定的应用背景和需求进行合理的组合，因此传统的编程方式不适合用来描述 Web Services 组合。为了寻求理想的方法和技术来描述 Web Services 的组合，在过去几年里，研究者们提出了很多 Web Services 组合描述语言：BPML，WSCI，BPMN，BPEL4WS，BPSS 以及 XPDA 等等。但在众多的语言之中，没有一个成为公认的标准。另一方面，这些语言大多基于 XML，虽然 XML 具有通用的表示形式和数据交换格式，但对于非 XML 专家而言，它仍然不易理解和编写。因此有必要把某种图形语言与 XML 结合使用来描述 Web Services 组合。文[26]把 UML 与 XML 相结合，提出用 UML 活动图来描述 Web Services 组合的方法，该方法生成的组合 Web Services UML 模型可以通过执行引擎和转换规则转换成不同的 Web Services 组合描述语言，从而实现了 Web Services 组合与描述语言无关。文中提出的方法分为 4 步：第一步，创建组合 Web Services 模型，并确定参与组合的候选 Web Services；第二步，从候选 Web Services 的 WSDL 文件逆向工程生成 UML 类图，从 UML 类图中识别出参与 Web Services 组合的具体操作，然后把所有参与组合的操作组合起来，从而生成组合 Web Services 的 UML 活动图；第三步，配置执行引擎并执行活动图转换生成的组合 Web Services 可执行规约；第四步，发布新生成的组合 Web Services。文[27]中提出了一种划分阶段的模型驱动 Web Services 组合方法，各阶段分别是：组合 Web Services 的抽象定义；组合 Web Services 调度；组合 Web Services 构建；可执行的组合 Web Services 映射。该方法根据 Web Services 组合的需求对业务规则进行分类，具体分类细节分别在文[28~30]中给以阐述。

5 面向 Web Services 的模型驱动开发框架 MDDF4WS

虽然目前存在支持 Web Services 开发的中间件平台，但 Web Services 开发缺乏坚实的方法学基础[31]。我们认为，MDA 能够为 Web Services 开发提供这一基础。虽然研究者们对面向 Web Services 的模型驱动开发做了一些研究工作，但现有的工作还存在很多需要改善之处：

1)还没有比较完备的基于 MDA 的 Web Services 开发框架。文[25]扩展了文[32]中提出的开放式建模设施，使之能支持 Web Services 的模型驱动开发，但并未考虑 Web Services 组合之前的可组合性验证，也没有考虑 QoS 在模型转换规则中的应用。

2)现有的 PIM 向 Web Services 转换大多采用直接转换，没有中间模型，源模型与目标模型耦合度大，不利于转换规则的重用。

3)现有方法主要采用扩展 UML1.4 为 PIM 建模，而 UML1.4 并不是针对分布式计算特定领域的，因此在实现 PIM 向 Web Services 转换的过程中要添加很多与 Web Services 相对应的语意信息，增加了转换的复杂性。

4)系统功能模型是由静态结构和动态行为组成的，但目前 PIM 向 Web Services 的转换方法没有显式把转换过程分为静态结构转换和动态行为转换，致使实现转换规则的转换引擎过于复杂，同时也不便于检验 PIM 动态模型的合法性。

为了全面系统地研究面向 Web Services 的模型驱动开发，克服现有研究方法的不足，从而实现 MDA 和 Web Services 的最佳结合，给异构系统交互提供理想的解决方案，我们

提出了基于 EDOC 的 Web Services 开发框架 MDDF4WS (Model Driven Development Frame for Web Services),该框架支持 Web Services 开发从建模到实现的全过程。

EDOC 是 OMG 提出的针对分布式企业运算的标准。和 UML1.4 相比,EDOC 与 Web Services 有更好的语意对应关系,因此我们的框架中采用 EDOC 为 PIM 建模。为了简化转换引擎的复杂性,我们将系统的 PIM 分为静态结构和动态行为两类,相应的转换引擎也分为静态和动态两种。为了降低源模型和目标模型的耦合性,提高转换规则的重用性,我们在转换的过程中引入了中间模型(Marked PIM),并分阶段地实现完整的转换过程。除此之外,我们的开发框架还可以对动态 PIM 的合法性进行验证,同时还可以在转换的过程中根据特定的 QoS 要求,优化相应的转换规则。

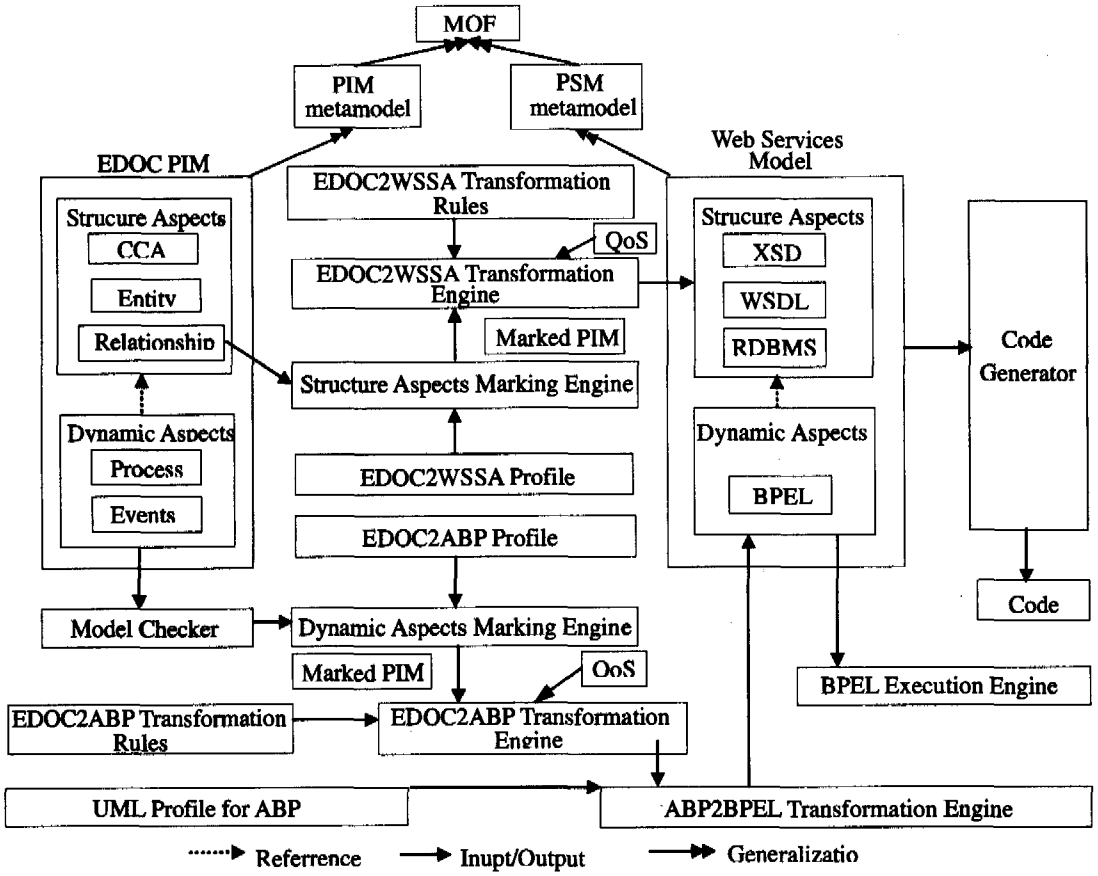

图 2 MDDF4WS 原理图

图 2 所示是我们提出的开放框架的原理图。系统的功能从静态结构(Structure Aspects)和动态行为(Dynamic Aspects)两方面来描述,其中 PIM 的静态结构采用 EDOC CCA profile、Entities profile 和 Relationships profile 建模,PIM 的动态行为采用 EDOC Business Process profile 和 Events profile 建模;Web Services 模型的静态结构采用 WSDL、XSD 和 RDBMS 建模,动态行为采用 BPEL 建模。因为 EDOC 和 Web Services 没有完全一致的对应关系,所以我们扩展了 EDOC,提出了 EDOC for Web Services Structure Aspects Profile(EDOC4WSSA)和 EDOC for Automated Business Process Profile(EDOC4ABP),其中 EDOC4WSSA 中包含与 Web Services 静态结构对应的语意信息,而 EDOC4ABP 则包含与 UML Profile for Automated Business Process(UML Profile for ABP)相对应的语意信息。

框架内的转换包括静态结构的转换和动态行为的转换。静态结构的转换分为两个阶段:第一阶段给静态 PIM 加标签,第二阶段把加了标签的 PIM 转换为静态 Web Services 模型。动态 PIM 的转换比静态转换复杂,我们扩展了文[21]的研究工作,把动态转换分成 4 个阶段:第一阶段检验动态 PIM 的合法性;第二阶段为动态 PIM 加标签;第三阶段把加了标签的动态 EDOC PIM 转换成用 ABP Profile 描述的 PIM;第四阶段实现第三阶段输出结果到动态 Web Services 模型的转换。有关动态转换第四阶段的细节请参见文[21]。除了支持 EDOC PIM 到 Web Services 的转换之外,我们的开发框架还支持 Web Services 具体实现的自动生成,即通过代码生成器(Code Generator)生成 Web Services 的实现代码。同时,我们的框架还可以根据具体的 QoS 要求实现转换规则的优化。目前我们的工作主要集中在 PIM 到 Web Services 的转换,代码生成的转换我们将会在以后的工作中进行研究。

结束语 随着电子商务的迅猛发展,对异构系统之间的灵活交互和集成有着迫切的要求,但传统的分布式对象技术需要交互的双方必须采用同类型的对象模型协议,而这些对象模型协议都是与厂商相关的,因此在无法严格受控的复杂、异构网络环境下,这些传统的分布式对象技术依然无法解决系统的交互和集成,无法满足电子商务进一步发展的需要。为了解决这些问题,传统的面向对象技术需要进一步发展,Web Services 则是这一发展的产物。Web Services 为解决分布式异构平台环境中的系统交互和系统集成问题提供了强有力的技术支持,因此在未来的软件开发和系统集成中必将扮演重要角色。

Web Services 从软件体系结构的角度解决了异构系统的交互和集成问题。与此同时,为了解决同样的问题,OMG 从

软件开发方法学的角度提出了 MDA。MDA 将系统的功能规约和系统功能在特定平台上的实现规约相分离，从而将技术与平台变化对系统的影响降低到最小程度。基于 MDA 开发的系统可以方便地实现现有系统交互、集成和系统移植，也可以方便地集成未来新技术，以及最大程度地实现模型复用，降低软件开发成本和保护现有投资。

我们认为 Web Services 和 MDA 的结合将会给异构系统的交互和集成提供理想的解决方案，因此研究基于 MDA 的 Web Services 开发是非常有价值的。本文分析了目前软件开发面临的巨大挑战，指出了研究动因，从 Web Services 和 MDA 的基本概念到二者结合的研究做了全面的阐述和分析，并提出了面向 Web Services 的模型驱动开发框架 MDDF4WS。

在今后的研究工作中，我们会逐步实现 MDDF4WS 中的转换算法，并使之成为真正可以运行的系统。

参 考 文 献

1 Carman M, Serafini L, Traverso P. Web Service Composition as Planning. In: Proc. of the Workshop on Planning for Web Services. Trento, Italy, June 2003

2 Martin J, Arsanjani A, et al. Web Services: Promises and Compromises. Queue, 2003, 1(1): 48～58

3 Staab S, van der Aalst W, et al. Web Services: Been There, Done That? IEEE Intelligent Systems, 2003, 18(1): 72～85

4 Soley R, the OMG Staff Strategy Group. Model driven Architecture. Object Management Group White Paper. November 27, 2000

5 W3C. Web Services Activity. http://www.w3.org/2002/ws/. Sept. 2004

6 Birbeck M, et al. Professional XML. 2nd ed. Wrox Press Inc, 2001

7 HP. web_services_tech_overview. www.hpmiddleware.com/downloads/pdf/web_services_tech_overview.pdf. Sept. 2004

8 http://www.w3.org/TR/wsdl/. Sept. 2004

9 http://www.uddi.org/. Sept. 2004

10 http://www.w3.org/TR/SOAP/. Sept. 2004

11 Kreger H. Web Services Conceptual Architecture (WSCA 1.0). IBM Software Group. May 2001

12 Tsur S, Abiteboul S, et al. Are Web Services the next revolution in E-Commerce? In: Proc. of the 27th Int'l Conf on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 614～617

13 http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis. Sept. 2004

14 Systinet. Web Services: A Practical Introduction. A Systinet White Paper. www.systinet.com, Sept. 2003

15 Middleware Company. Model Driven Development for J2EE Utilizing a Model Driven Architecture (MDA) Approach. Middleware Company Technical report. www.middleware-company.com/casestudy. June 2003

16 Klepper A, Warmer J, Bast W. MDA Explained : The Model Driven Architechure: Practice and Promise. Addision Wesley, ISBN 0-321-19442-X, April 21, 2003

17 Gerber A, Lawley M, et al. Transformation: The Missing Link of MDA. First International Conference on Graph Transformation (ICGT2002), Barcelona , Spain. October 2002

18 Will Provost. WSDL First . http://webservices.xml.com/pub/a/ws/2003/07/22/wsdlfirst.html. Sept. 2003

19 Will Provost. UML for Web Services. http://www.xml.com/pub/a/ws/2003/08/05/uml.html. Sept. 2003

20 Bezinvin J, Hammoudi S, et al. Applying MDA Approach for Web Services Platform. In: Proc of the 8th IEEE Int'l Enterprise Distributed Object Computing Conf (EDOC 2004). Monterey, California, USA . Sept2004. 20～24

21 Keith Mantell. From UML to BPEL. www.ibm.com/developerworks/ webservices/library/ws-uml2bpel/. Sept. 2004

22 OMG. EDOC Part II v1.0, Annex E - Technology mappings from EDOC to Distributed Component and Message Flow Platform Specific Models. OMG Document Number: ad/2001-08-20

23 OMG. UML Profile for Enterprise Distributed Object Computing Specification(EDOC). OMG Document Number: ptc/2001-12-04

24 Patrascoiu O. Mapping EDOC to Web Services using YATL. In: Proc. of the 8th IEEE Int'l Enterprise Distributed Object Computing Conf (EDOC 2004). Monerey, California, USA, Sept. 2004

25 Kath O, Blazarenas A, et al. Towards Executable Models: Transforming EDOC Behavior Models to CORBA and BPEL. Proc of the 8th IEEE Int'l Enterprise Distributed Object Computing Conf(EDOC 2004). Monterey, California, USA, Sept 2004

26 Skogan D, Gronmo R, Solheim I. Web Service Composition in UML. In: Proc. of the 8th IEEE Int'l Enterprise Distributed Object Computing Conf. (EDOC 2004). Monterey, California, USA, Sept. 2004

27 Orriens B, Yang Jian, et al. Model Driven Service Composition. www.unitn.it/convegni/download/ icsoc03/papers/Bart Orriens.pdf. Sept. 2004

28 Business Rules Group. Defining business rules, what are they really?. http://www.brcommunity.com, July 2000

29 von Halle B. Business rules applied: Building Better Systems Using the Business Rule Approach. Wiley & Sons, 2002

30 Veryard R. Rule Based Development. CBDi Journal, July/August, 2002

31 de Castro V, Marcos E, Vela B. Representing WSDL with Extended UML. www.unab.edu.co/editorialunab/ revistas/rcc/pdfs/r51-art1 c.pdf. Sept. 2004

32 Kath O, Soden M, et al. An Open Modeling Infrastructure integrating EDOC and CCM. In: Proc. of the 8th IEEE Int'l Enterprise Distributed Object Computing Conf (EDOC 2003). Brisbane, Australia, Sept. 2003. 198～207

(上接第 259 页)

2 A Methodology for Detection and Estimation of Software Aging. Sachin Garg, Aad van Moorsel, Kalyanaraman Vaidyanthan and Kishor S. Trivedi. In: Proceedings of The Ninth International Symposium on Software Reliability Engineering, 1998

3 Bao Yujuan, Sun Xiaobai, Trivedi K S. Adaptive Software Rejuvenation: Degradation Model and Rejuvenation Scheme. 2003 International Conference on Dependable Systems and Networks, 2003. San Francisco, California

4 Barlow R E, Campo R. total Time on test Processes and Applications to Failure Data analysis . reliability and Fault Tree Analysis (R. E. Barlow, J. fussell and N. Singpurwalla, eds.), SIAM, Philadelphia, 1975. 451～481

5 Okamura H, Fujimoto A, Dohi T, Osaki S, Trivedi K S. The Optimal Preventive Maintenance Policy for a Software System with Multi Server Station. In: Proc. 6th ISSAT Int'l Conf. Reliability and Quality in Design, 2000. 451～481

6 huang Y, Kintala C, KoletisN, Funton N D. Software Rejuvenation: Analysis, Module and Applications . In: Proc. 25th IEEE Int'l symp. On Fault Tolerant Computing, IEEE Computer Society Press , Los Alamitos, CA, 1995. 381～390

7 Candea G, Delgado M, Chen M, Fox A. Automatic failure-path inference: A generic introspection technique for software systems. In : Proc. 3rd IEEE Workshop on Internet Applications, San Jose, CA, 2003

8 Kephart J O, Chess D M. The vision of autonomic computing. Computer, 2003, 36(1): 41～52

9 Konstantinou A V, Yemini Y. Programming Systems for Autonomy. In: Proceedings of the Autonomic Computing Workshop fifth Annual International Workshop on Active Middleware Services (Ams'03)