一个面向服务契约的Web服务适配器

李军怀 张 璟 张卓彬
(西安理工大学计算机科学与工程学院 西安 710048)

摘 要 Web服务适合于分布、松散耦合的应用环境。针对企业应用系统中的Web服务适配器开发问题,提出并实现了一个面向服务契约的Web服务适配器。详细介绍了适配器所包括的分析引擎、框架代码生成引擎、基于Web的WSDL文档编辑器等关键技术和实现方法。同时,将本方法与相关工具进行了性能分析和对比,表明了本方法的良好交互性和稳定性。

关键词 Web服务,服务契约,服务适配器,引擎
中图法分类号 TP311 **文献标识码** A

Service Contract-oriented Design Approach to Developing Adaptable Services

LI Jun-huai ZHANG Jing ZHANG Zhuo-bin
(School of Computer Science & Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract Web Service is fit for distributed, loose-coupling application environment. A Web service adapter of service contract-oriented was presented in order to solve the problem of enterprise application system development. We still introduced some key issues about service contract analysis engine, skeleton code generate engine and Web based WSDL editor in the Web service adapter. Also, the Web service adapter designed in this paper was compared with related tools, the results showed that our methods are more interactive and stability.

Keywords Web services, Service contract, Service adapter, Engine

多年以来,大多数分布式应用程序的开发人员都习惯于分析对象然后开始编写一大堆的代码,这也是我们所熟知的"代码先行(Code First)"。在面向服务的时代,随着Web服务渐渐成为了分布式应用程序特别是异构系统整合的标准协议,这种服务自治,共享协定但不共享实现的软件构造方式得到了充分的应用[1]。Web服务应用接口系统的开发原理——"契约先行(Contract First)"也源于此。这个契约就是WSDL文档,即先产生基于模式(Schema)的WSDL文档,再产生代码。

采用契约先行的开发方法,可以提高应用系统开发效率,由于服务设计与开发分开,服务的设计者通过WSDL具体化需求,然后交于开发人员实现。服务提供者和服务请求者能够并行工作。客户端的开发人员能够独立于服务器端的开发人员,两者都可以以它们使用的开发平台进行开发。这样能够帮助用户快速开发出与之相对应的Web服务接口适配器。另外,可以最大限度解决客户端和服务器端异构问题,首先设计服务的互操作性,即首先在XSD中创建数据类型,然后设计基于此数据类型的服务(WSDL),最后使用自动化的工具产生服务器端或者客户端的Web服务接口代码。这种"逆向"的开发方法能够最大限度地解决系统互操作性问题,而且还提高了代码的鲁棒性,提高了代码的开发效率。

本文针对Web服务开发和应用中的异构系统适配问题,设计实现了一个面向服务契约Web服务接口适配器,主要包括分析引擎、框架代码生成引擎以及基于Web的WSDL文档编辑器。

1 问题分析与系统架构

由于企业中大量遗留系统的存在,在进行企业应用系统集成时的关键问题是如何对遗留系统进行封装,使其在不改变原有实现的基础上能够与其它遗留系统乃至企业外部的系统进行交互[2-4]。

遗留系统中存在着异构的数据、异构的应用逻辑及异构的表现等层次,在进行系统集成时无需也无法对每一个异构层次都进行封装。由于异构系统的应用逻辑是整个异构系统的核心,它体现着该应用所要表达的业务逻辑,因此可以采用功能集成的方式对异构系统的应用逻辑部分进行封装,并以Web服务的标准形式将相应的功能向ESB(企业服务总线)注册,最终将功能以服务的形式暴露给外界用户。

通过对企业遗留系统可能的开发平台进行分析研究,发现有很大一部分遗留系统是在.NET平台上开发的,而另一些遗留系统是在J2EE等其它平台上开发的。对那些与应用集成系统平台完全异构的遗留系统,为了与系统进行交互,通

到稿日期:2009-01-16 返修日期:2009-03-20 本文受国家863重点项目(编号:2007AA010305和2007AA010402),西安市科技计划项目(编号:GX07026和LD0704)资助。

李军怀(1969—),男,博士,副教授,主要研究方向为分布式计算、Web服务及应用,E-mail:lijunhuai@xaut.edu.cn;张 璟 教授,博士生导师,主要研究方向为服务计算、网络化制造;张卓彬 男,硕士研究生。

过 Schema 文件描述其系统的相关接口，通过使用标准的元数据文件，可以实现对异构系统的封装。

虽然不同平台使用不同的集成方法，但在集成过程中，统一使用面向服务的集成方法，将不同类型遗留系统的具体功能适配为标准的 Web 服务，并对外进行发布，在不影响原有系统应用的基础上可以保证系统间的松散耦合[4]。

2 面向服务契约的适配器

依据上节所描述基本思想设计的 Web 服务适配器包括 3 大核心部分，分别是 Analysis Engine(分析引擎)、Generate Engine(框架代码生成引擎)以及基于 Web 的 WSDL 文档编辑器[3,4]。这 3 个部分相互配合，又各自独立，在契约的自动化生成过程中可循环迭代，直到产生满足系统需求的服务契约和框架代码。系统的核心模型如图 1 所示。

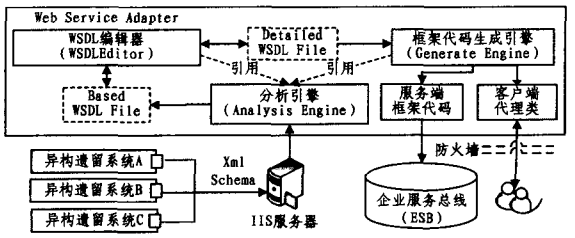

图 1 Web 服务适配器结构

分析引擎对所有输入信息进行分析并自动生成描述系统服务的基本服务契约(Based WSDL File)。

WSDL 编辑器对上述契约文档进行进一步的修改和编辑，最终生成内容详尽的服务契约(Detailed WSDL File)，之后契约被送入 Generate Engine。

Generate Engine 负责分析最终的服务契约，依据该服务契约自动产生服务端框架代码和客户端代理类。

由于服务端框架代码和客户端代理类由同一份服务契约产生，共享稳定的接口。这样，服务开发人员可方便地在服务端框架代码的基础上加入具体的逻辑实现，以完成服务的开发；服务消费者也可直接使用该代理类完成相关功能的开发，而无需担心服务端代码改变后对代理类的重新修改和部署。因此，服务双方可并行地进行开发，大大提高了整个应用的开发效率。

3 关键技术实现

为了保证系统的可扩展性和更适于变化，在进行服务契约文档和框架代码的构建时使用到了 Builder(生成器)模式，从而保证了构建过程中的具体步骤可以按要求进行替换；在对基于 Web 的 WSDL 编辑器进行设计时，使用 Abstract Factory(抽象工厂)模式，使 WSDL 文档的关键节点可以动态生成并动态添加，从而保证了契约文档定制的可扩展性；在进行框架代码的生成过程中，使用 Strategy(策略)模式，使不同的代码生成算法得到封装，从而降低了系统的耦合度，实现了代码生成算法的动态可扩展性。

3.1 契约分析引擎

契约分析引擎是异构适配器的底层组件，负责对用户提供的服务契约进行解析，以及按照用户提供的 Schema 文件进行对应契约的构建。它实现了两个主要功能——服务契约的解析与服务契约的构建。

在进行契约解析时，引擎接收用户提交的 WSDL 契约文档，首先判断其基本格式是否正确，若正确则进入具体的解析流程。分析引擎采用基于文档对象模型(DOM)的方式，对 WSDL 文档进行解析，构建一个 XML DOM 树，并将其转换成一个 WSDL DOM 树，以实现对契约文档的解析[5]。服务契约的解析流程如图 2 所示。

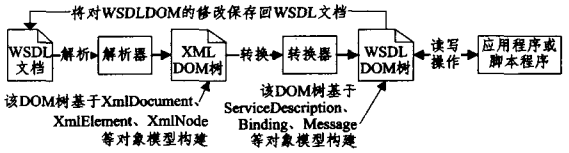

图 2 服务契约的解析流程

服务契约的构建是服务契约解析流程的逆向过程，也是分析引擎的核心。契约文档构建算法基于 Builder(生成器)模式，以 WSDL 契约文档作为构建的目标对象。使用生成器模式，将复杂的 WSDL 契约文档的构建过程封装起来，向客户类隐藏了具体构建过程的实现，允许 WSDL DOM 树通过多个步骤来创建，并且可以添加或改变某一具体的构建步骤[6]。契约文档的构建算法描述如下所示：

算法 GenerateWSDL

```
Input:Function based Xml Schema File,CustomInfo
Output:Based WSDL File
//算法开始
Declare Fbxsd as XmlSchema;
Fbxsd= XmlSchema. Read(Function based Xml Schema File);
//创建 Messages 节点对象;
For each xmlSchemaElement in Fbxsd. Items
Begin
    Messages. Add(xmlSchemaElement);
End For
Messages. Add(CustomInfo);
//创建 PortTypes 节点对象
For each message in Messages. Items
Begin
    Switch Type of the message
      Case Request/Response:
          PortTypes. Operation. Add(Operation with In and
          Out Message);
      Case Oneway:
          PortTypes. Operation. Add(Operation with only In
          Message);
    End Switch
End For
//创建 Bindings 节点对象;
For each portType in PortTypes. Items
Begin
    SoapBinding spbinding=new SoapBinding
                          (portType. Attributes);
    Bindings. Add(spbinding);
End For
//创建 Services 节点对象;
Services. Name= CustomInfo;
Services. Ports. Add(CustomInfo);
//创建 WSDLDOM 对象(ServiceDescription)
ServiceDescription. Add (Messages, PortTypes, Bindings, Serv-
```

ices);

ServiceDescription. Write(//目标路径);

Return ServiceDescription;

3.2 基于 Ajax 的 WSDL 编辑器

基于 Web 的 WSDL 文档编辑器，以 WSDL 文档的关键节点为单元对 WSDL 文档进行编辑，这样可以简化用户对复杂契约文档编辑的难度。在编辑完成后，可以向用户展现完整的服务契约文档。此外，WSDL 编辑器与基于桌面应用的编辑器一样具有新建、打开、保存等文档基本操作的功能。

采用 Ajax 技术后，相当于在客户端和服务器端之间加入了一层 Ajax 引擎(XmlHttpRequest 对象)。将用户提交的 WSDL 契约文档解析为 ServiceDescription 对象，并最终转化为 JSON(javascript object notation)对象。当客户端发送一个 Ajax 请求时，实际是向服务器端传送一个 JSON 对象，对象解析器首先进行 JSON 解析，将 JSON 对象内所包含的描述 WSDL 文档的 ServiceDescription 对象提取出来，转化为一个 WSDLDOM，然后编辑器调用 Analysis Engine 分析该文档，提取出关键节点的属性信息，将其动态绑定在页面 TreeView 控件的相关节点上，完成文档的加载工作。用户可以按节点类型对 WSDL 文档进行编辑和修改，之后仍以 JSON 对象的形式将保存的文档发回客户端，客户端异步解析出 ServiceDescription 对象，并调用该对象的 Write 方法更新本地的 WSDL 文档，从而完成对契约文档的编辑工作，得到经过用户完善内容详尽的服务契约文档(Detailed WSDL File)。整个编辑过程以异步方式进行，保证编辑页面无刷新，给用户很好的体验[7]。

3.3 框架代码生成引擎(Generate Engine)

在得到描述系统的服务契约后，基于“契约先行”的开发原理，服务端和客户端都要以此契约为标准，进行各自功能的开发。框架代码生成引擎以 WSDL 契约文档作为输入参数，自动生成服务端 Web 服务框架代码和客户端代理类，从而简化了服务双方的开发流程，使“契约先行”的开发方式更容易得到实现。

与分析引擎类似，在进行框架代码的生成时，也分为两个主要的流程——服务契约的解析流程和框架代码的构建流程。由于分析引擎已经实现了对服务契约进行解析的完整功能，因此在服务契约的解析流程中可以直接使用分析引擎对外提供的解析接口来实现对服务契约的解析工作；而在具体的框架代码构建中，将分析引擎对契约解析的实时结果及时映射为对应的代码元素，在代码构建流程中引用分析引擎的解析接口，使契约的解析和代码的生成并行执行，提高代码生成的效率。

由于需要产生服务提供者和消费者两种不同类型的代码，因此在设计框架代码生成引擎时，使用策略模式，将服务端框架代码的生成算法和客户端代理类的生成算法分别独立封装起来，使它们之间可以相互替换，同时向外界提供统一的算法接口。

4 性能分析与评价

Christian Weyer 的 WSCF 是目前为数不多支持从契约先行的角度对 WSDL 服务契约进行自动化生成的工具。本文从工具的交互性、契约生成的稳定性等方面对 Web 服务适配器的核心构建契约分析引擎进行比较分析与评价。

(1)工具交互性

WSCF 是一款基于桌面的工具，它与 Visual Studio 开发环境紧密结合，用户必须在 Visual Studio 的支持下才能完成契约的生成工作，因此该工具的适用性有很大的限制。本文开发 Web 服务适配器的核心部分契约分析引擎的实现完全基于 Web 应用并且与开发环境无关，因此可以使各种角色的用户(不仅仅是熟悉开发环境的开发人员)对服务契约进行定制，具有更好的平台独立性，能使异构系统乃至异构平台的用户对其系统进行描述并自动生成服务契约，因此更能满足 SOA 开发的需要。

(2)契约生成的稳定性

下面的测试结果是通过使用不同大小的 Schema 架构文件传入 WSCF 和本文开发的契约分析引擎中，获得对应的服务契约，同时记录每次契约的生成时间而得到的。

图 3 显示了在小数据量架构文件(文件大小小于 500kb)条件下的测试结果，从这组数据中可以看出，在传入的架构文件较小的情况下本文的契约分析引擎与 WSCF 在契约生成的效率与稳定性方面基本持平；图 4 显示了在大数据量架构文件(文件大小大于 500kb)条件下的测试结果，从图中可以看出，当架构文件的大小逐渐增大时，WSCF 的契约生成时间波动很大，而本文开发的契约分析引擎的契约生成时间并没有产生急剧的波动，工具表现出了很好的稳定性。

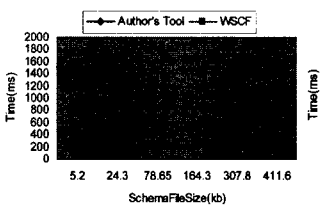

图 3 服务契约的生成时间(小数据量)

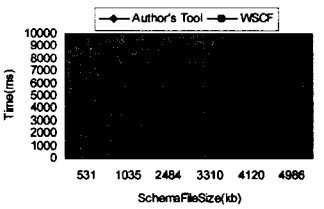

图 4 服务契约的生成时间(大数据量)

结束语 按照“契约先行”的开发方法，设计了一个面向服务契约的 Web 服务适配器，实现了对同、异构遗留系统的良好封装。本文提出并实现的 Web 服务适配器已经成功应用到多家大型企业信息系统的开发中，取得了良好的效果。但是，本文方法尚无法对随需应变的企业业务流程，进行编排与处理。因此下一步的工作应放在如何实现一个基于该集成系统的工作流引擎，对封装的服务功能进行随需编排，实现对企业应用的更高级集成。

参 考 文 献

[1] 麻志毅，陈泓婕. 一种面向服务的体系结构参考模型[J]. 计算机学报，2006，29(7)：1011-1017

[2] Benatallah B, Casati F, Grigori D, et al. Developer Adapters for Web Services Integration[C]// the proceedings of the 17th International Conference on Advanced Information Systems Engineering(CAiSE 2005), LNCS 3520. 2005：415-429

[3] Chang Soo Ho, Kim Soo Dong. A Service-Oriented Analysis and Design Approach to Developing Adaptable Services[C]// 2007 IEEE International Conference on Services Computing (SCC 2007). July 2007. Salt Lake City, Utah, USA. IEEE Computer Society, 2007：204-211

(下转第 134 页)

因此，存在最佳的再生周期，使系统的可用性达到最大，即 Tr_1 为 9 时，系统可用性最大为 0.9362。

表 3　系统可用性与 Tr_1 关系

Tr_1	3	6	9	13
Avail	0.9307	0.9346	0.9362	0.9352
Tr_1	16	19	23	26
Avail	0.9325	0.9295	0.9257	0.9230
Tr_1	29	33	36	39
Avail	0.9205	0.9179	0.9164	0.9154

图 3 为 $Tr_2=20$，Tr_1 分别为 16，24，32 时，集群系统的吞吐量随时间的变化情况，从图中可以看出：(1)经过一段时间后，系统的吞吐量趋于稳定；(2)随着 Tr_1 的增加，稳态吞吐量逐步减小，但差值也逐步减小。

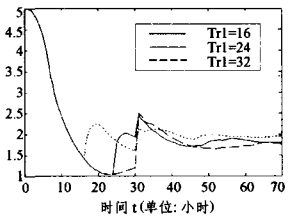

图 3　吞吐量随时间的关系图

结束语　服务器集群是提高系统 QoS 和可用性的一种方法，在现实中应用得越来越广。但由于软件老化的影响，使得系统的可用性与 QoS 不能得到保障。由于软件老化过程与时间有关，针对两节点集群系统，采用流体随机 Petri 网对系统的工作模式进行建模，分析系统各个状态概率，给出系统的可用性和吞吐量计算公式，并进行仿真试验。结果表明本文的方法能很好地描述系统且具有可扩展性，可以分析求得系统的最佳再生周期。

软件老化过程除了与时间有关外，还与已处理的任务数、系统当前的排队等待处理的任务数有关。因此，该研究点下一步的工作是在用 FSPN 对系统工作行为建模时，考虑已处理任务数和排队等待任务数对系统的影响；并考虑其它再生策略。

参 考 文 献

[1] Grottke M, Li L, Vaidyanathan K, et al. Analysis of Software Aging in a Web Server[J]. IEEE Transactions on Reliability, 2006, 55(3): 411-420

[2] Cotroneo D, Orlando S, Russo S. Characterizing Aging Phenomena of the Java Virtual Machine[C]// 26th IEEE International Symposium on Reliable Distributed Systems(SRDS'07). 2007: 127-136

[3] Vaidyanathan K, Trivedi K S. A Measurement-Based Model for Estimation of Software Aging in Operational Software Systems [C]// Intl. Symp. On Software Reliability Engineering. ISSRE 1999, Nov. 1999

[4] Silva L, Madeira H, Silva J G. Software aging and rejuvenation in a soap-based server[C]// IEEE-NCA: Network Computer and A-pplications. July 2006: 56-65

[5] Avritzer A, Weyuker J. Monitoring Smoothly Degrading Sys-tems for Increased Dependability [J]. Empirical Software Eng. Journal, 1997, 12(1): 59-77

[6] Huang Y, Kintala C, Kolettis N, et al. Software Rejuvenation: Analysis, Module and Applications [C]// Proc. 25th Symp. Fault Tolerant Computing. Pasadena, USA. IEEE Press, 1995: 381-390

[7] Gribaudo M, Sereno M, Horvath A, et al. Fluid Stochastic Petri nets augmented with flush-out arcs: modeling and analysis [J]. Discrete Event Dynamic Systems, 2001, 11(1): 97-117

[8] Horton G, Kulkarni V G, Nicol D M, et al. Fluid stochastic Petri nets: Theory, application, and solution techniques[J]. European Journal of Operation Research, 1998, 105(1): 184-201

[9] Vaidyanathan K, Harper R E, Hunter S W, et al. Analysis and Implementation of Software Rejuvenation in Cluster Systems [C]// Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. 2001: 62-71

[10] Xie W, Hong Y, Trivedi K S. Software rejuvenation policies for cluster systems under varying workload[C]// Proceedings of 10th IEEE Pacific Rim International Symposium on Dependable Computing. March 2004: 122-129

[11] Kiejin Park, Sungsoo Kim. Availability analysis and improve-ment of active/standby cluster systems using software rejuve-nation [J]. Journal of Systems and Software, 2002, 61(2): 121-128

[12] Aung K M M, Park K, Park J S. A rejuvenation methodology of cluster recovery[C]// IEEE International Symposium on Cluster Computing and the Grid. May 2005: 90-95

[13] Fan Xin-yuan, Xu Guo-zhi, Eng Ren-dong, et al. Performance analysis of software rejuvenation on dispatcher-worker based cluster system[C]// Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies. 2003: 562- 566

[14] Koutras V P, Platis A N, Applying Software Rejuvenation in a Two Node Cluster System for High Availability[C]// Proceedings of the International Conference on Dependability of Computer Systems. 2006: 175-182

(上接第 130 页)

[4] Kuo D, Fekete A, Greenfield P, et al. Expressing and Reasoning about Service Contracts in Service-oiented Computing[C]// IEEE International Conference on Web Services(ICWS'06). 2006: 915-918

[5] Skonnard A. Contract-First Service Development[EB/OL]. http: // msdn. microsoft. com/msdn mag/issues/05/05 /Service Station/, 2005-05-23

[6] SParastatidis A, Woodman S, Webber J, et al. Asynchronous Messaging between Web Services Using SSDL[J]. IEEE Internet Computing, January 2007: 26-39

[7] Bhuiyan J, Nepal S, Zic J. Checking Conformance between Business Processes and Web Service Contract in Service Oriented Applications[C]// Australian Software Engineering Conference (ASWEC'06). 2006: 80-89