第 32 卷 第 21 期 计 算 机 工 程 2006 年 11 月
Vol.32 No.21 Computer Engineering November 2006
• 网络与通信 • 文章编号：1000—3428(2006)21—0127—03 文献标识码：A 中图分类号：TP393.09

基于 Web Service 的 MAS 的集成系统的设计与实现

刘震宇，张素庆，王文杰
(中国科学院研究生院，北京 100039)

摘 要：设计了 Web Service 与多 Agent 系统的集成系统的体系结构。提出用 Web Service Agent 转换器来实现 ACL 消息与 SOAP 消息之间的转换，并对 Web Service Agent 转换器重要模块进行了实现，在消息层面上实现了 Web Service 与 Agent 之间的互连互通。从实验结果看，效果比较明显。

关键词：Web 服务；多 Agent 系统；MAGE；Web service Agent 转换器

Design and Implemention of Web Service Based MAS Integration System

LIU Zhenyu, ZHANG Suqing, WANG Wenjie
(Graduate School of Chinese Academy of Sciences, Beijing 100039)

【Abstract】This article designs a Web service and MAS integration system architecture. It proposes with Web service agent translator to realize the translation from ACL message to SOAP message, and implements the main module of the Web service agent translator, and makes the Web service agent communicate with each other. Seen from the result of the experiment, the effect is obvious.

【Key words】Web service; MAS; MAGE; Web service agent translator

在基于Web Service的应用[1]中，通常情况下用户为了找到合适的Web Service，由用户发送一个请求到UDDI服务器，UDDI服务器按照用户的请求进行查询，将查询的Web Service列表返回给用户。在这样的情况下用户要访问每一个Web Service的接口来判断结果Web Service是否是自己所需要的。如果Web Service数量较多，用户的工作量就非常大，显然这种查询方式效率很低，也缺乏灵活性而且用户不能定制自己的个性化需求。

Agent具备普通程序所没有的自治性、响应性、主动性、推理性以及社会性，它是Web Service使用知识、代表或代替用户行为的最佳实体。Agent可以在一定程度上模拟人类社会的行为，具有一定的智能和优良的判断力，能够自主运行，具有极大的灵活性和适应性，适合于Web这样开放、动态的网络环境。行为(Behaviors)是Agent的基本属性之一。服务(Service)的一个重要特点是它可以是一个黑盒子，可被客户端(可以是程序也可以是Agent)所访问，所以Web Service可以作为Agent行为的一部分[2]。

将 Web Service 集成到 MAS 系统能够实现动态和有效的查询结果并满足用户的个性化需求。客户 Agent 首先接收用户的需求，将用户的需求传递到 UDDI 服务器实现查询，这样查询的结果不是直接传给用户，而是经过 MAS 系统处理后再传给用户，用户最终得到的是数量较少的、更加符合自己要求的 Web Service。

本文利用 Web Service 与 Agent 两种技术各自的优点，在保持各自平台特点的基础上实现二者集成。设计并实现了一种相对简单、具有可扩展性且松散耦合的集成系统。

1 基于 Web Service 的 MAS 的集成系统

1.1 系统的体系结构

系统的体系结构如图 1 所示，系统中有两种类型的Agent，即ARB Agent[3]与Wrapper Agent[3]。前者所代理的服务由各种Agent平台提供，这些平台能够与FIPA Agent平台互相通信；后者代理由Web Service所包装的各种软件描述。客户Agent首先查询DF来发现所需服务由哪种类型的Agent提供，从而建立相应的绑定。在MAS中，Wrapper Agent代理是由Web Service封装的各类软件系统；ARB Agent代理是由Agent能够直接调用的服务。

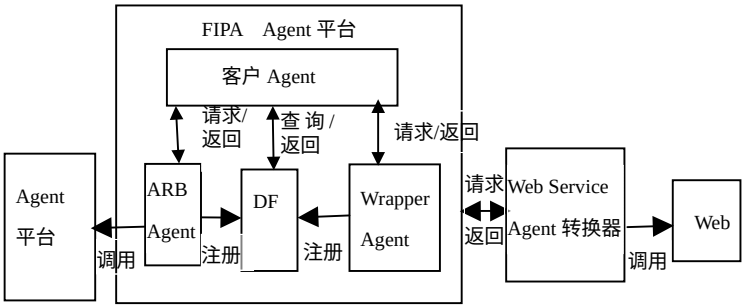


图 1 基于 Web Service 的 MAS 的集成系统体系结构

Web Service Agent 转换器是系统的核心部件，它接受客户 Agent 的 ACL 请求，将 ACL 请求转化成 WSDL 描述，并将 WSDL 服务请求发送到 UDDI；生成 SOAP 请求调用 Web Service，并将结果转化成 ACL 消息模式，最后将这一 ACL 消息返回 Wrapper Agent。

1.2 Web Service Agent 转换器的结构

Web Service Agent 转换器的结构如图 2 所示，主要由 MAGE 和 Jakarta Tomcat 服务器两大部分组成。

JUDDI：允许 Web Service 向其注册服务描述。

协同 Agent：协调并控制转换器中各 Agent 的行为顺序。

作者简介：刘震宇(1975－)，男，硕士生，主研方向：智能处理；张素庆，硕士生；王文杰，博士、副教授

收稿日期：2006-04-10 **E-mail：**Kevin_78@126.com

服务 Agent：完成 Web Service 的调用。

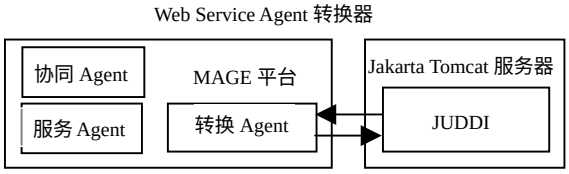


图 2 Web Service Agent 转换器的结构

转换 Agent 主要完成：

(1)检测 JUDDI 中的新注册的 Web Service 描述，将它们翻译成 MAGE 描述描述语言 ADL，提交给 Wrapper Agent。

(2)解析客户 Agent 的服务请求命令，如果在 JUDDI 数据库中存在对应的命令条目，将创建一条 SOAP 消息发送到服务 Agent 来调用 Web Service，其参数对应于 FIPA 请求命令参数。

(3)提取所调用的 SOAP 消息中的结果，将其转化成 ACL 格式。

1.3 Web Service Agent 转换器运行的支持平台

(1)Jakarta Tomcat (http://jakarta.apache.org/tomcat/)

(2)Apache jUDDI (http://ws.apache.org/juddi/)

(3)mySQL(http://www.mysql.com/)

(4)MySQL Connector/J (http://www.mysql.com/products/connector/j/)

(5)MAGE 平台

之所以采用这些平台，主要考虑除 MAGE 外它们具有免费的开放源代码，开放源代码的特点非常适合试验项目，而且提供对目前所有 Web Service 协议的支持。

2 Web Service Agent 转换器服务程序的设计与实现

2.1 服务接口程序

在分布式应用中，典型方法是采用接口类来描述服务。接口是一个纯抽象类，其作用是描述服务提供的方法，服务的基本单位是接口类（Interface）。转换器服务接口用来描述服务，它并不实现任何逻辑过程，只是通过接口类来描述转换器可以提供的方法以及参数、返回结果类型。其 UML 如图 3 所示。

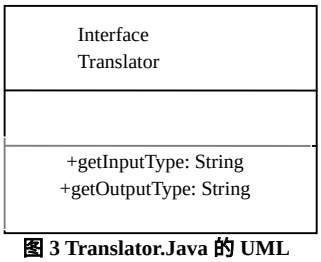


图 3 Translator.Java 的 UML

转换器服务接口仅仅表示它以提供的方法调用，接口本身不包括任何处理逻辑。本例中服务接口 Translator 有两个方法 getInputType()和 getOutputType()可被调用。服务接口的部分代码如下所示：

```
public interface Translator{
//converts a message.
    public Collection translate( CalledMessage inMsg ) throws Exception ;
    public String getInputType();
    public String getOutputType();
}
```

2.2 协同 Agent 的实现

在查询 UDDI 之后，需要控制服务 Agent 与合适的 Web 服务实现进行绑定，主要根据各任务及相关数据流的描述通过服务发现机制找到具体的服务提供者。服务集成平台需建立执行和协调机制，因为被集成的服务通常分布在网络不同节点中。

预先设定的工作流中相关的数据和控制结构已由设计者指定，服务实例化后，一些服务特定的信息还需手工填入集成业务流程中，根据此业务流程生成相应的协同 Agent 的行为类。协同 Agent 负责执行 Web 服务调用序列，并且担当同步调用和异步调用之间的转换，根据 MAGE 平台的特征设计了其行为，该 Agent 程序由以下 5 个部分组成:

```
/* ProcessAgentjava:负责协同 Agent 的交互，Web 服务的集成流程*/
public class ProcessAgent extends Agent{
        public void setup(){
        //创建 Agent 的行为类。
        /* ①根据服务集成流程，声明某个具体行为 */
        addBehaviour(new getFlowBehaviour( <fluenName>)){}
        /* ②将集成流程的结构映射为相应的 Agent 行为类 */
      addBehaviour(new struMappingBehaviour(<fluenName>)){}
       /* ③声明所有的原子动作(即调用 Service Agent) */
        addBehaviour(new callSerAgentBehaviour (<Service Agent name>, InterPattern)) { }
        // InterPattern: MP 或 MA, MP is Message Passing, MA is Mobile Agent
        /* ④集成执行结果 */
        addBehaviour(new InteResultsBehaviour()){}
        }
}
/* 声明上述 4 个 Behaviour, getFlowBehaviour 详细如下，其他 Behaviour 结构与其相似 */
private class getFlowBehaviour extends Behaviour{
        public void action(){}
        public boolean doneQ{}
}
```

协同 Agent 程序包含了常规任务的知识，根据服务集成流程得到具体的服务执行序列，至于具体的服务调用则通过发送服务调用请求至服务 Agent 完成。协同 Agent 可以感知来自服务 Agent 的响应，以获得必要的信息。

2.3 服务 Agent 的实现

服务调用 Agent 的功能包括:接受协同 Agent 提出的服务调用请求；并根据 WSDL 文件进行相应的参数传递，真正调用相应的 Web 服务。这里更关注的是在 MAGE 框架下，服务 Agent 如何与协同 Agent 通信和协作。服务 Agent 完成调用某个具体 Web 服务的功能。服务 Agent 程序大体框架如下所示:

```
/* ServiceAgentjava:负责调用 Web 服务的 MAGE Agent */
public class ServiceAgent extends Agent{
        public void setupQ{
        //创建一个 Agent 的行为类。
        addBehaviour(new CallServiceBehaviour(){}
        //清除 Agent 操作
          protected void takeDown(){}
```

```
        }
    }
    /*声明其 Behaviour，这里的 Behaviour 为调用某个 Web 服务的过程。*/
        private class CallServiceBehaviour extends Behaviour{
     public void action(){
            private boolean 函数名(){
            //调用相应 Web 服务的具体代码
            //构造返回的结果值
                return true;
            }
        }
    public boolean done(){}
    }
```

2.4 转换 Agent 的实现

转换 Agent 主要完成 ACL 消息与 SOAP 消息之间的相互转换，它在客户 Agent 和目标 Web Service 服务之间担任桥梁，是转换器服务组件中最主要的组件之一，不仅仅承担客户端请求传递给实际的目标 Web Service 实体。其主要目的是接受协同 Agent 提取客户端调用信息，转换成 WSDL 格式，组成 SOAP 格式的 Web Service 消息；另一方面，提取协同 Agent 提取的调用结果并转换成 ACL 消息，将 ACL 消息返回客户 Agent。转换 Agent 程序大体框架如下所示：

```
/* convertor from ACL to SOAP message content. */
public class ACLToSOAP implements Translator{ …… }
/* convertor from SOAP to FIPA ACL message content. */
public class SOAPToFIPACL implements Translator{ …… }
```

3 实验及结果分析

我们针对各种情况进行了多次试验，在所有以简单字符串为输入的试验中均获得较好结果，不仅结果返回正确而且平均响应时间很短。在大字符串作为输入参数的试验中，响应时间较长，主要原因是异步调用和同步调用之间的同步问题不容易解决，大字符串对象在基于消息为基础的异步调用过程中，对性能影响很大。在使用以中文字符串为基础的测试过程中，MAGE 支持中文处理，能够返回结果但响应时间较长。

文献[3]介绍了 Web Service Agent 集成项目，它是基于 JADE 平台的网关实现，文献[4]对其进行了进一步的研究，表 1 列出了该方法的实验结果。由此可见，在以简单字符串为输入的试验中，结果较好；发现如果字符串对象大于 10KB 之后，失败次数开始增多。在使用以中文字符串为基础的测试过程中，其返回结果均为乱码。主要原因在于 Jade 容器不支持对中文的处理。

表 1 实验结果比较

平台类型	基于 JADE 的 WSAG 平台	基于 MAGE 的 WSAT 平台
	试验结果	
简单字符串	成功（<1S）	成功（<1S）
100KB 的字符串	失败（Timeout）	成功（>1S）
中文字符串	乱码（<1S）	成功（>1S）

从表 1 可以看出，基于 MAGE 的 WSAT 平台在处理复杂字符和中文字符时有较好的性能。另外还应注意到，在基于 JADE 的 WSAG 平台中，客户端由 Web Service 担任，消息调用采用同步调用的方式实现，当调用一个消息时，必须等待上一个调用结果返回才能进行下一进程。而在基于 MAGE 的 WSAT 平台中，客户端由 Agent 担任，它采用异调用的方式，可以在发出一个消息调用后进行下一进程，这明显节省了用户等待消息返回所用的时间，同时，Agent 可以对返回的结果进行筛选，能够为用户提供更加个性化的服务。

4 结束语

Web Service 和多 Agent 技术都是独立发展起来的技术，不仅体系结构不尽相同，通信协议也完全不同。集成这两个平台，首先需要解决的是消息的通信问题，本文采用 Web Service Agent 转换器，能够较好地解决两个平台的消息通信。从试验结果看，效果较好。Web Service 和多 Agent 平台的结合将拓展各自应用领域，对实现 Web 服务自动化有较强的理论意义和现实意义。

参考文献

1 王晓霞. 基于 Web Services 的应用集成技术研究与实现[D]. 石家庄: 华北电力大学, 2002.

2 Buhler P A, Vidal J M. Semantic Web Services as Agent Behaviors[D]. Dept. of Computer Science, College of Charleston, University of South Carolina, 2003.

3 FIPA Agent Software Integration Specification. Foundation for Intelligent Physical Agents[EB/OL]. http://www.fipa.org/specs/fipa00079, 2001.

4 Web Services Agent Integration, Whitestein Technologies AG. Web Services Agent Integration Project[EB/OL]. http://wsai.Sourceforge.net/gateway. html, 2003.

5 杨 光. 基于 Web Service 的多 Agent 系统网关的技术研究[D]. 南京: 河海大学, 2004.

（上接第 126 页）

参考文献

1 Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid:Enabling Scalable Virtual Organizations[J]. International Journal of Supercomputer Applications, 2001, 15(3): 200-222.

2 周佳祥, 郑纬民, 杨广文. 一种基于进程迁移的自适应双阈值动态负载平衡系统[J]. 清华大学学报(自然科学版), 2000, 40(3): 121-125.

3 陈忠林, 孙 雁, 干 静. 分布式网络环境下的负载平衡原理及算法[J]. 四川大学学报(工程科学版), 2003, 35(6): 97-100.

4 Subramani V, Kettimuthu R, Srinivasan S, et al. Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests[C]. Proc. of the 11th IEEE International Symposium on High Perfomance Distributed Computing, Edinburgh, Scotland, 2002-07.

5 Anderson T E, Culler D E, Patterson D A. A Case for NOW (Networks of Workstations)[M]. IEEE Micro, 1995-02.