一个面向 Web服务集成的 Agent协同框架

罗家奇[1], 王业先[2], 李 斌[1]

(1. 扬州大学 信息工程学院, 江苏 扬州 225009; 2 盐城市盐都区信息中心, 江苏 盐城 224005)

[摘要] 多 Agent系统适用于 Web服务集成的分布式环境; 软件 Agent相比 Web服务具有更好的主动性和交互能力. 面向 Web服务集成的 Agent协同框架以软件 Agent和 Web服务作为系统基本组件, 可以构造开放的、分布式的、动态的、自主的、智能化的服务集成系统. 具有设计、编辑 Web服务集成流程; 验证服务集成流程的正确性; 执行 Web服务集成流程; 监控 Web服务集成流程的执行等功能.

[关键词] Web服务, Web服务集成, Agent协同, Agent会话

[中图分类号] TP 311 [文献标识码] A [文章编号] 1672-1292(2008)04-0123-05

An Agent Coordination Framework for Web Services Composition

Luo Jiaqi[1], Wang Yexian[2], Li Bin[1]

(1. Information Engineering College, Yangzhou University, Yangzhou 225009, China;
2 Yandou Area Information Center of Yancheng City, Yancheng 224005, China)

Abstract: Multi agent system are suitable for the integration of Web services. Compared with Web services, Agent is more active and interactive. An agent coordinated framework systems oriented on Web services composition is based on the agent and Web services components. Using the framework, a service composition system which is open, distributed, dynamic, independent and intelligent could be constructed. It has a workflow of designing and editing Web services composition, validating services composition workflow, executing Web service composition workflow, and monitoring Web composition workflow, etc.

Key words: Web service; Web services composition; Agent coordination; Agent conversation

传统的 Web服务集成存在集中式、被动性和缺乏自适应性等不足[1, 2]; 软件 Agent具有自主性、主动性、社会性、推理性以及自适应性等特点[3]. 本文针对传统 Web服务集成的不足提出了一个面向 Web服务集成的 Agent协同框架. 该平台以软件 Agent和 Web服务作为系统基本组件, 使用一种抽象三层体系架构来定义基于 Agent的 Web服务集成框架. 通过该框架可以构造开放的、分布式的、动态的、自主的、智能化的服务集成系统.

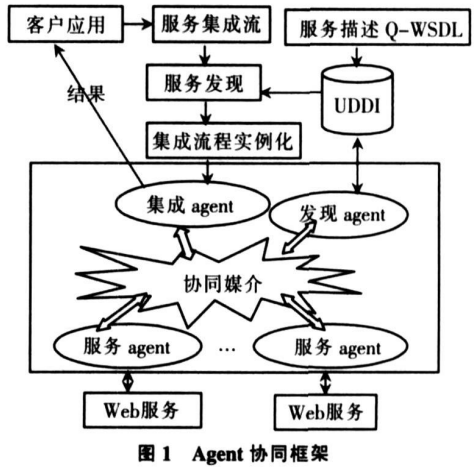

图 1 Agent 协同框架

Fig.1 An agent coordinated framework

1 面向 Web服务集成的 Agent协同框架

面向 Web服务集成的 Agent协同框架结构如图 1所示, 该框架具有设计、编辑 Web服务集成流程; 验证服务集成流程的正确性; 执行 Web服务集成流程; 监控 Web服务集成流程的执行等功能. 框架分业务流程层、Agent处理层和 Web服务层 3个层次. 业务流程层是用户与系统交互的接口, 主要向应用程序和用户提供一种编辑用户需求的环境. Agent处理层是

收稿日期: 2008-06-18
基金项目: 江苏省自然科学基金(BK2007074)和江苏省教育厅自然科学基金(06KJB520132)资助项目.
通讯联系人: 罗家奇, 实验师, 研究方向: 计算机软件、计算机仿真及应用. E-mail: jqluo@yzu.edu.cn

Web服务集成体系架构的核心,负责流程执行、服务定制、服务代理、协同工作,可实现服务集成流程的动态运行. Web服务层是服务集成系统的最终执行者. 包含多个 Web服务,这些 Web服务由 Agent发现和选择,并实例化到具体的业务流程中. 在流程执行过程中,每个 Web服务由 1个服务 Agent代理.

客户应用:针对目前 Web服务集成现状,用户用自然语言说明其需求,再实现服务的自动查找和集成是不太现实的. 用户层应包括一个服务集成语言编译环境. 在本系统中,用户应提供服务集成的需求、所要调用的功能及其功能间的结构等.

服务集成流: Web服务集成首先考虑的是使用一种有效的建模语言来为服务集成建模,并使用集成服务定义语言来描述它. 可以使用 BPEL4WS, BPEL4WS支持服务的静态和动态集成. 在服务发现之前先给出服务集成的大致流程,服务发现后,该流程实例化成真正绑定服务的流程. 客户应用程序根据用户请求,形成满足要求的服务集成流程,并提供可视化的集成流图形.

服务发现:在流程运行时刻,发现 agent进行服务动态查找,此时会在服务注册中心 UDDI中有关于服务的扩展描述.

集成流程实例化:绑定能够实现所需功能的 Web服务到原来流程中的服务.

多 agent系统:多 agent系统是多个 agent协同工作,实现服务集成流程的动态运行. 是整个框架的核心,多 agent系统由发现 agent,集成 agent和服务 agent组成. Agent之间通过 Agent交互平台进行交互. 实现的方式是 agent通信. 各个 agent功能在后面的 agent设计中描述.

Agent交互平台:实现 agent间的交互. 功能上等同于实例化流程的执行引擎. 由于多 agent系统交互方式的多样性,以及对环境的适应性,因此用多 agent系统执行服务的集成更灵活.

服务:指原子服务或复合服务. 复合服务是一种树状结构的服务,每个叶子节点都是一个原子服务,相互合作实现一定的功能. 原子服务提供对基于 Internet应用程序的访问,不依赖于其它的 Web服务去实现外部的请求. 复合服务集成了多个服务. 原子服务和复合服务一样被生成、发布、发现和调用.

2 Agent设计

面向 Web服务集成的 Agent协同框架共有 3种 Agent,如集成 Agent,发现 Agent和服务 Agent它们都是依照 JADE Agent模型设计:基本结构类似;能力模块部分不同.

2.1 发现 Agent的设计

2.1.1 发现 Agent模型

发现 Agent是接受用户应用程序输入的抽象流程,根据抽象流程从 UDDI中查找和匹配 Web服务,依据 QoS计算值选择相应的 Web服务,并将 Web服务绑定到流程中,将抽象流程实例化. 发现 Agent的主要功能有 3个子模块:服务匹配模块、服务 QoS计算模块和服务选择模块.

服务匹配:该过程根据服务的描述进行匹配,这些描述主要包括服务名及其它相关的文本描述. 根据这些描述计算服务间的相似度.

QoS度量计算:根据当时服务所处的环境,基于 QoS模型中的度量标准来动态计算服务的 QoS值. 其目的是确定 Web服务有多大的综合操作能力.

服务优选:根据上面 2个步骤得到的结果,通过选择算法选择一个现时最优的服务或一组满足客户需求且最优的服务集来实现集成.

2.1.2 抽象流程实例化

发现 Agent接收客户端应用程序提交的抽象流程描述,通过查找和选择相应的 Web服务,从而实现抽象流程的实例化.

① 接收抽象流程:发现 Agent从客户端应用程序接收服务集成的抽象描述.

② 查找 UDDI:根据要求解析出集成流程中所需用到的 Web服务,然后在 UDDI中查询满足此需求的服务.

③ QoS计算:一般情况下都会找到多个满足需求的 Web服务,这些 Web服务具有不同的服务质量. 发现 Agent根据一定的算法对这些候选的 Web服务进行分析,选择出最符合要求的 Web服务.

④流程实例化:将最终选择的 Web服务绑定到抽象流程中,即将 WSDL信息填入抽象流程中,从而

将抽象流程实例化为 BPEL4WS流程.

2.2 集成 Agent的设计

2.2.1 集成 Agent模型

集成 Agent是服务集成过程的协调者, 从用户应用程序获取实例化的流程, 将流程分解为若干子流程分发给服务 Agent, 并负责监控和控制集成流程的执行, 将服务集成的结果返回给用户应用程序. 集成 Agent的主要功能有 5个子模块: 流程分解、Agent会话、发送子流程、运行控制和结果收集.

流程分解: 将服务集成流程分解为若干子流程, 由服务 Agent执行这些子流程.

Agent会话: 集成 Agent通过与服务 Agent会话确定执行子流程的服务 Agent

发送子流程: 确定好执行子流程的服务 Agent后, 集成 Agent将子流程发送给服务 Agent

流程执行监控: 监控和调整服务集成流程的执行.

结果收集: 集成 Agent收集服务集成执行结果, 并将结果返回给用户应用程序.

2.2.2 服务集成过程

集成 Agent是服务集成过程的协调者, 在整个流程集成中担负着流程分解、流程发送、流程监控和结果反馈等任务, 集成 Agent在服务集成中的具体过程如下:

① 接收流程: 集成 Agent从应用程序客户端应用程序接收服务集成流程.

② 流程分解: 通过划分全局服务集成流程, 产生各个服务的本地服务流程, 从而将服务集成的控制逻辑和执行负载对等地分布到各个 Web服务结点上.

③ 流程发送: 采用 Agent间丰富的协同方式, 经过与其它服务 Agent的会话, 集成 Agent将本地服务流程脚本发送给相应的服务 Agent

④流程监控: 在集成流程执行过程中, 由于环境的动态变化以及服务的自主性等原因, 服务集成系统可能会发生这样那样的异常情况. 为了确保服务集成系统的顺利执行, 集成 Agent有必要对服务集成流程系统的执行过程进行监控, 从而达到服务集成系统自适应复杂多边网络环境的目的.

⑤结果反馈: 在整个服务集成流程执行完后, 集成 Agent需将集成结果反馈给用户应用程序.

2.3 服务 Agent的设计

2.3.1 服务 Agent模型

服务 Agent是 Web服务的代理, 是本地服务流程的执行者. 通过执行本地服务流程参与到服务集成中去, 包括与其它服务 Agent交互, 调用 Web服务的相关操作, 同时根据 Web服务的相关情境信息调整 Web服务, 达到 Web服务的自适应. 服务 Agent的主要功能有 5个子模块: Agent会话、子流程解析、发送子流程、服务调用和服务自适应.

Agent会话: 包括服务 Agent与集成 Agent和服务 Agent与其它 Agent两部分会话.

子流程解析: 解析服务子流程, 根据解析的动作来调用服务及与其它服务 Agent交互.

访问控制: 控制其它 Agent对服务中方法的访问.

服务调用: 调用 Web服务中的方法.

服务自适应: 在服务调用过程中, 服务 Agent作出相应的处理.

2.3.2 调用服务过程

服务 Agent是与各个具体的 Web服务绑定在一起的, 接收来自集成 Agent发送过来的工作流脚本, 依照工作流脚本与其它服务 Agent进行交互, 调用相应的 Web服务操作, 在调用操作的过程中感知相应的情境并作出相应的处理, 以达到服务的自适应. 具体的过程如下:

① 接收流程: 选择相应的 Agent协同方式, 服务 Agent从集成 Agent接收服务本地流程.

② 流程解析: 对于服务 Agent而言, BPEL4WS语言不是可执行的语言. 为了执行本地流程, 服务 Agent需要将 BPEL4WS本地流程转化为 Agent行为.

③ 服务调用: 根据服务本地流程解析出的行为, 服务 Agent调用 Web服务中相应操作.

④服务感知: 在服务本地流程的执行过程中, 由于环境的动态变化以及服务的自主性等原因, Web服务可能会发生各种异常情况. 为了确保 Web服务的顺利执行, 服务 Agent有必要对 Web服务的执行过程进行感知, 从而达到服务集成系统自适应复杂多变网络环境的目的.

⑤服务调整:根据服务的感知结果,参考相应的服务调整策略,服务 Agent调整相应的服务操作.

3 Agent会话

Agent会话是 2个或多个 Agent为达到一定的目的而进行的一系列消息传递过程.研究表明,使用会话作为分析 Agent之间的通信的基本单位比使用单个消息更容易建模,也更容易实现[4].因为 Agent会话规定了 Agent之间消息传递的内容和顺序,具有高度可靠性和一定的柔性.

3.1 集成 Agent与服务 Agent的会话

在 Web服务流程被分解后,集成 Agent和服务集成流程所对应的服务 Agent进行会话,请求服务 Agent执行服务子流程,服务 Agent有 3种回应: ①可以执行; ②不可以执行; ③可以执行但需延迟一段时间.相应地,集成 Agent也有 3种处理方法: ①如果可以执行,集成 Agent向服务 Agent发送子流程; ②如果不可以执行,集成 Agent请求发现 Agent查找替代服务 Agent; ③如果可以执行但需要延迟一段时间,集成 Agent需要对延迟的时间做出评估以确定是否能接受.对于服务子流程的执行请求,服务 Agent的回应是基于情境做出的: ①如果服务 Agent正在运行的服务实例数目小于它允许运行的服务实例数目,服务 Agent做出可以执行的回应; ②如果正在运行的服务实例数目达到可以运行的实例数目上限,并且请求的实例数目比较多,服务 Agent做出不可以执行的回应; ③如果正在等待执行的服务实例数目比较少,服务 Agent做出延时的回应.

```
Conversation-Protocol between Composition Agent and Service Agent
Coordination_Agent behavior(decomposedFlow f)
1: A ← GetServiceAgent(f)
2: response ← Request(A)
3: if response= accept then
4:    SendFlow(A, f)
5: else if response= delay(t) then
6:    if Coordination_Agent_can_accept_t_time delay then
7:       accept(A)
8:       SendFlow(A, f)
9: else
10:       reject(A)
11:       search(substituteA)
12:    end if
13: else if response= reject then
14:    search(substituteA)
15: end if
Service_Agent_behavior(ServiceRequestC)
1: search Service Context
2: if Running_instance_number < Allowed_instance_number then
3:       return accept
4: else if Request_instance_number> 1 then
5:       return reject
6: else
7:       return delay(next_Instance_available_time)
8: end if
```

3.2 服务 Agent间的会话

当从服务情境中获知服务实例执行完后,服务 Agent从服务情境中可以知道这个信息:该服务实例在集成流程中所对应的下一个服务 Agent 当前服务 Agent向下一个服务 Agent发送执行服务子流程的通知,下一个服务 Agent有 2种回应: ①确认可以执行; ②不可以执行.相应地,当前服务 Agent也有 2种处理方法: ①如果可以执行,服务 Agent向下一个服务 Agent发送相应的数据,通知集成 Agent更新集成情境关

于当前服务 Agent的描述; ②如果不可以执行, 服务 Agent首先将服务情境中关于下一个服务 Agent的描述设为 NULL, 然后通知集成 Agent更新它的集成情境, 最后请求发现 Agent重新查找服务. 类似于前面的会话, 下一个服务 Agent的回应也是基于情境做出的: ①如果服务 Agent正在运行的服务实例数目小于它允许运行的服务实例数目, 服务 Agent做出可以执行的回应; ②如果正在运行的服务实例数目达到可以运行的实例数目上限, 服务 Agent做出不可以执行的回应.

```
Conversation- Protocol between Service Agents
Previous_Service_Agent behavior()
1: service_context_inform ← instance_done
2: NA ← GetNextInstanceService (service_context)
3: Response← Inform(NA)
4: if response= affirm then
5:    send (NA, data)
6: else if response= disable then
7:    Send(C, NA, disable)
8:    SetNextInstanceService (Service_Context, Null)
9:    Search(substitute_Agent)
10: end if
Next_Service_Agent behavior(ServiceInform A)
1: search(Service Context)
2: if Running_instance_number < Allowed_instance_number then
3:    return affirm
4: else
5:    return disable
6: end if
```

4 结语

本文结合已有的 Web服务集成研究, 针对其中的集中式、被动性和缺乏自适应性等缺点, 提出一个面向 Web服务集成的 Agent协同框架, 支持服务集成系统的设计、发现、集成、分析、部署和监控. 详细介绍了协同框架的主要成分, 说明了该框架设计、部署 Web服务集成系统的步骤, 描述了基于多 Agent系统在整个框架中的核心作用. 阐述了多 Agent系统的设计以及其中的 Agent会话.

[参考文献] (References)

[1] V de Castro, Marcos E, Lopez Sanz M. Service composition modeling: a case study[C] // The 7th Mexican International Conference on Computer Science. Mexico: San Luis Potosi, 2006: 101-108.

[2] Karunamurthy R, Khendek F, Glitho R H. A novel business model for Web service composition[C] // The 2006 IEEE International Conference on Services Computing. Washington D C: IEEE Computer Society, 2006: 431-437.

[3] Wang Shuying, Shen Weiming, Hao Qi. An agent-based Web service workflow model for inter-enterprise collaboration[J]. Expert Systems With Applications, 2006, 31(4): 787-799.

[4] Lin F H, Norrie D H. Schema-based conversation modeling for agent-oriented manufacturing system[J]. Computers in Industry, 2001, 46: 259-274.

[责任编辑: 丁蓉]