文章编号:1005-0523(2009)03-0052-06

语义 Web Service 可视化组合平台的设计与实现

李明翠

(华东交通大学 信息工程学院,江西 南昌 330013)

摘要:随着越来越多的Web服务共享在网络上,为了更加充分的利用共享的Web服务,有必要将简单的Web服务组合起来,以提供更为强大的服务功能。文章着重研究语义Web服务的可视化合成技术,阐述了一个可视化的组合平台的实现原理。提出了基于本体层面,通过在图形面板中创建目标控制流和数据流的方法创建复合服务的方法;重点讨论了创建顺序结构和选择结构复合服务的方法;通过组装网络上和本地的Web服务,说明组合平台的组合方法是可行的。

关 键 词:Web服务;语义;可视化;组合
中图分类号:TP302.1 **文献标识码**:A

Web服务组合,是指把Internet上已有的功能相对简单的Web服务按一定的业务逻辑流程组合起来以构成复杂的复合服务,从而提供更强大、更完整的商业功能[1,2]。语义Web服务就是语义Web在Web服务领域应用的产物,它是一种更智能的Web服务,是Web服务未来的发展趋势[3]。本文主要研究基于OWL-S和本体的WEB服务合成方法,设计和实现一种可视化的、使用简单的、能编辑和执行复合语义Web服务的WEB服务合成平台SWSComposer,为业务开发人员提供可视化的设计工具。

1 SWSComposer的主要功能

本文设计和实现的SWSComposer平台将组装的过程用图形化的方式实现,让Web服务组装者在轻松的图形操作中就能完成Web服务的组装,而不需要具备太多的语义Web和Web服务方面的知识。SWSComposer平台主要实现基于本体层面的可视化组合功能,包括将已发布的Web Service的WSDL文档转换成OWL-S格式的语义Web Service描述文档;提供可视化的组装模式,让用户通过点击鼠标就能创建复杂目标服务的控制流和数据流;自动将用户创建的控制流和数据流转换成OWL-S描述文档;提供执行语义Web Service的功能。SWSComposer的主要功能模块及处理流程如图1所示。

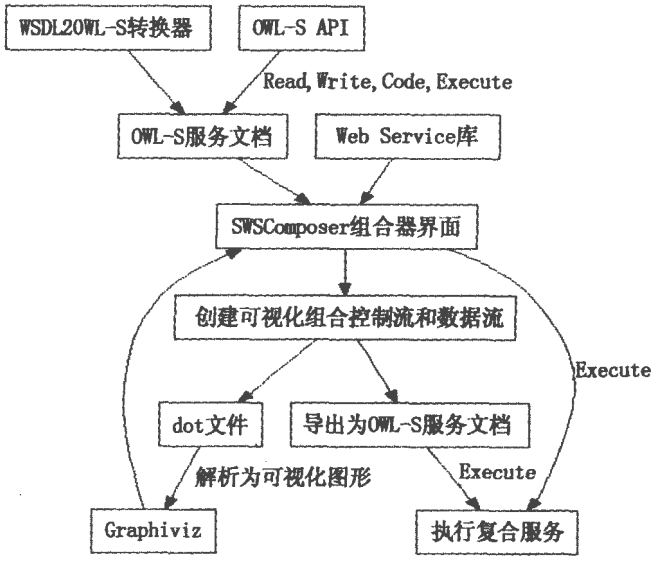

图1 SWSComposer体系结构

SWSComposer为用户提供了一个综合的UI操作界面,在组合器界面中,用户在其中进行服务的创建、组装、打开以前创建的服务、执行服务等功能。图中的WSDL2OWL-S转换器将网络上已发布的Web Service的WSDL文档导入SWSComposer并转换成OWL-S服务描述文档。Web Service库用于存储Web服务的OWL-S文档。目前本系统假设用户已经查找到组装过程中所需要的Web Service,并将其OWL-S文档存储于Web Service库中。

SWSComposer平台主要通过Eclipse平台开

收稿日期:2009-03-21

作者简介:李明翠(1980-),女,湖南人,讲师,主要研究方向为数据挖掘、Web service。

发实现,使用了 Spring 程序构架,用 Log4j 进行过程监视;可视化部分调用 AT&T Labs — Research 的 Graphviz 软件显示图形,调用 grappa jar 包实现图形交互处理;调用 MindSwap 的 owls jar 包和 HP Lab 的 jena jar 包读取和处理 OWL-S 和 RDF 文档;测试所用的领域本体文档用 Stanford 的 protege 开发;测试所用的本地 Web Service 用 Eclipse + Myeclipse 开发,用 Tomcat 作为发布服务器。

2 SWSComposer 平台的可视化处理

SWSComposer 平台的组装面板中的图形是用户组装的复杂服务的控制流和数据流的综合体现。与组装图相关的数据结构主要有 3 类:节点列表(用于保存组装图中控制流的所有节点)、边列表(用于保存组装图中所有控制流的边)、数据流节点列表(用于保存组装图中所有的数据流节点)。

平台中的组装图形通过调用 Graphviz 的 dot.exe 运行显示,用户与图形的交互借助 ATT 提供的 grappa 接口实现。平台中的主类实现了监听图形动作的接口和普通监听接口:GrappaListener 和 Actionlistener。并在主类的容器中加载了用于显示 dot 图形的图形面板:GrappaPanel。GrappaListener 将监听发生在 GrappaPanel 中的事件。

平台中采用 Graphviz 的 dot 格式显示组装图,每次用户对组装图(插入节点和边,编辑节点和边)更新时,都将重新生成对应的 dot 文件,再由 dot.exe 重新显示该图形。Dot 文件的数据来自当前组装的复合服务的节点列表(dotNodeList)、边列表(edgeList)和数据流列表(dataFlowList),创建完 dot 文件后,用 grappa 的解析器将 dot 文档解析成 Graph 图形,由 Java 运行时运行 dot.exe 程序显示 Graph 图形。生成 dot 文件时,根据节点列表的不同,采用了不同的形状显示,状态节点用默认的椭圆形显示,动作节点和 produce 节点用方形框显示,条件节点用菱形显示。

3 创建顺序结构的复合服务

创建顺序结构(Sequence)的复合服务主要是通过向新建的服务中依次加入已存在的 Web Service,并在各服务节点间创建相关的数据流来完成组装工作。在执行复合服务时顺序执行这些服务。在组合平台中体现为向组合图形中依次插入动作节点(ActionNode,每个动作节点对应一个用于组合的原子服务)并在动作节点间创建相应的数据流图。插入动作节点的过程通过插入动作节点对话框完成,对话框主要为用户提供与插入动作节点关联的 OWL-S 服务文档。对话框的组合框中显示的是 repository 中存储的 OWL-S 文档所描述的服务名称。用户组装完成的复合服务也可以保存在 repository 中,供组装其它更复杂的服务使用。SWSComposer 每次启动时都会首先加载 repository 中的 OWL-S 服务文档信息。当用户组装完复合服务后,将复合服务保存在 repository 中,这时候可以刷新 repoitory,重新加载 repository 中的服务文档,从而可以使用刚刚组装完的复合 Web 服务。在 SWSComposer 平台中,用户只需要用鼠标操作图形,在图形中设计复合服务的控制流和数据流,再点击"Export to OWLS"就能得到想要的复合服务。导出为 OWLS 的过程就是创建复合服务 OWL 文档的过程,整个过程在后台根据 DotNodeList、EdegeList、DataFlowMap 以及用户导入的领域本体依据 OWL-S 规范实现。将顺序结构的复合服务导出为 OWL-S 文档的处理流程如算法 1 所示。

算法 1:导出复合服务

(1) 创建 OWLOntology 对象 ont。

(2) 根据 ont 的节点列表、边列表、数据流节点列表创建复合服务。

① 在 ont 中创建组合服务的复杂过程

a.根据 Produce 节点的输入创建复杂过程的输出;

b.在 ont 中根据节点类表创建控制流结构(顺序、选择结构等);

c.根据数据流列表绑定节点间的输出和输入;

d.将节点列表中没有在 c 步骤中绑定的输入设置为复杂过程的输入。

② 创建复合服务的 profile

③ 为复合服务的 grounding 增加复杂过程中每个原子过程的 grounding

(3) 将 ont 写入 owl 文件保存。

4 创建具有条件分支(if-then-else)结构的复合服务

创建具有条件分支的复合服务旨在为用户提供根据分支前的 Web Service 的不同输出结果,而选择执行不同的 Web Service 的功能。例如,客户在网上购买图书,假设已经有发布好的 Web 服务:findBook 服务提供在网络商店中查询图书的功能;payBook 提供购买图书的功能;advise 服务为用户提供错误提示功能。我们可以将这三个服务组装成一个复合服务,直接为用户提供查询并购买图书的功能。但并不是每次查询图书时都有自己想要买的图书,因此组合后的复合服务应该有两种结果,一种是查询到了想买的图书,并直接进行了购买行为,另一种是没有查询到想买的图书,这时候可以调用简单的信息提示服务告知客户没有找到指定的图书。因此有必要为组合服务设置一个条件节点,将 payBook 服务组装在条件节点的条件为真时的执行分支中,将 advise 服务组装在条件节点的条件为假时的执行分支中。当 findBook 服务查找到了用户需要的图书时,将条件节点的条件设置为真,继续执行 payBook 服务,否则为假,从而实现根据不同的查询结果执行不同的服务的功能。

创建具有条件分支结构的复合服务主要通过向新建服务中加入 If 条件语句,形成具有 Then 和 Else 两个分支的复合 Web Service。SWSComposer 中使用 RDF 三元组设置条件表达式,即采用 predicate(domain, range)的方式作为条件表达式。例如可以在购买图书的复合服务中增加一个条件节点 userfindbook(user, book),表示用户找到图书的情况。条件节点所用的 RDF 三元组对象和实例来自组装过程中导入的领域本体。

用 findbookService(查询图书)、paybookService(付款)、adviseService(没有查找到图书后的提示)三个原子服务组装成的具有条件分支的复合服务 ifFindBookPayService。IfFindBookPayService 在组合服务的复杂过程的树结构如图 2 所示。树中的 Perform1、Perform2、Perform3 分别调用 findbookService 服务的过程 findbookProcess、paybookService 服务的 paybookProcess 过程和 adviseService 服务的 adviseProcess 过程。整棵树的外层为顺序结构,顺序结构的第一个节点调用 finbookService 服务,顺序结构的第二个节点嵌套了 IF-Then-Else 结构,其中 then 部分调用 paybookService 服务,而 else 部分调用 adviseService 服务。且运行的顺序是:先调用 findbookProcess,再处理 If-Then-Else 结构,当条件成立时转去执行 then 部分,否则执行 else 部分。

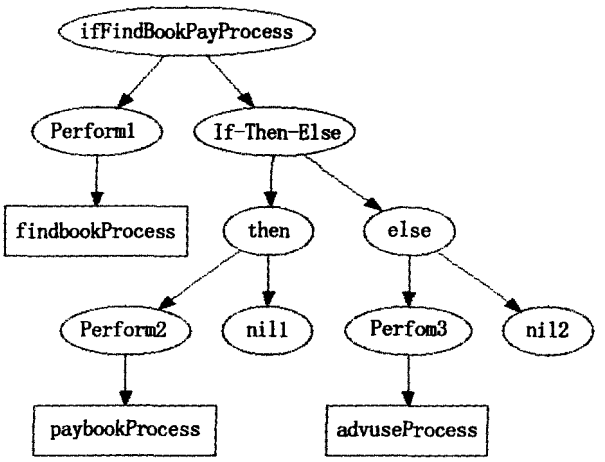

图 2 ifFindBookPayProcess 的树形结构

SWSComposer 实现具有 If-Then-Else 条件分支的复合服务的可视化组装要求 Web Service 具有良好的语义描述,即 Web Service 的 IOPR(Input, Ouput, Precondition, Result)需要很好的定义。执行一个 Web Service 后客户得到两种信息:一个是 Ouput,它是调用一个服务后得到的数据信息;另一个是 Effect,它是调用一个服务后对世界状态的改变,例如执行购买图书的服务后,客户可能得到两种信息,一个是购买图书的发票,另一个是客户信用卡状态的改变。OWL-S 规范用 Result 表示一对 Output 和 Effect,用以满足不同上下文的需求。OWL-S 规范使用 inCondition、hasResultVar、withOutput、hasEffect 四个属性来描述一个 Result。其中 inCondition 属性指定了该 result 出现的条件,withOutput 和 hasEffect 属性表示的是当 inCondition 所表示的条件成立的时候将会产生的状态。

SWSComposer 平台进行分支条件组装时,先对特定的 OWL-S 文档增加语义标注,如在 findBook 中添加

Result 注释,使 findBook 根据不同的查询结果产生不同的 Effect。假设服务 findBook 在没有查寻到用户需要的图书时的输出结果(Ouput)为"Sorry, there is no this book you specified.",则可以将 findbook 的 Result 设置为这样一个组合:当 findBook 的输出不同于"Sorry, there is no this book you specified."时,将效果设置为 userfindbook(user, book),而当 findBook 的输出与"Sorry, there is no this book you specified."时,将效果设置为 usercannotfindbook(user, book)。而查询并购买图书的复合服务的条件节点用 userfindbook(user, book)设置,在条件分支的成立分支上增加 payBook 服务,完成购买功能,而在条件分支不成立的分支上增加 advise 服务,提示用户购买其它书籍。

在将组装好的复合服务导出为 OWL-S 文档时,需要为复合服务定义控制结构(Control Constructs)。每个 If-Then-Else 控制结构分 ifcondition、then 结构、else 结构三部分。创建 If-Then-Else 控制结构的算法如算法 2 createIfThenElseControlConstruct 所示。

算法 2:createIfThenElseControlConstruct

将条件节点 conditionNode 和条件分支的闭合节点 closeNode 之间的节点序列转换成 IfThenElse 控制结构,conditionNode 是条件节点,closeNode 是以 conditionNode 为分支条件的 Then 分支和 Else 分支的闭合节点

(1) 创建新的 IfThenElse 结构 ifthenelse;

(2) 创建用 SWRL 描述的条件 condition,用于表达 conditionNode 节点指定的条件,将 ifthenelse 控制结构的 ifcondition 设置为 condition;

(3) thenNode = conditionNode.getThenNode();

elseNode = conditionNode.getElseNode();

(4) 如果 thenNode 不等于 closeNode,则

① 将 thenNode 和 closeNode 之间的接点序列转化成顺序结构 sequence1;

② ifthenelse 控制结构的 then 结构设置为 sequence1。

(5)如果 elseNode 不等于 closeNode,则

① 将 elseNode 和 closeNode 之间的接点序列转化成顺序结构 sequence2;

② ifthenelse 控制结构的 else 结构设置为 sequence2。

(6) 返回条件结构 ifthenelse。

5 SWSComposer 平台的执行模块

在执行模块,首先获取用户当前选择的语义 Web Service 的复杂过程(Process)的输入参数,按照这些输入参数的不同类型在执行面板中添加不同的输入组件,以供用户从中输入数据。语义 Web Service 通过 OWL-S API 调用执行,OWL-S API 提供了一个执行引擎(ExecutionEngine),通过这个执行引擎就可以运行原子过程(AtomicProcess)或组合过程(CompositeProcecess)。组合服务的运行过程与原子服务的调用过程完全相同,只须提供初始的输入条件即可,组合过程内部的执行流程则由执行引擎按照 OWL-S 描述的过程控制逻辑自动地执行。

运行服务的过程如下:

(1) 建立执行引擎(ProcessExecutionEngin);

(2) 为执行引擎增加过程监控器(ProcessMonitor);

(3) 加载当前服务的知识库(KnowledgeBase);

(4) 获取服务的过程(Process);

(5) 从执行面板中获取输入参数值(Values);

(6) 通过执行引擎运行 Process,Process 的参数为 Values;

(7) 获取运行的结果,将结果显示在执行面板中。

6 组装测试

本平台假设已经查找到组装所需要的已发布的 Web Service,在组装前先将这些 Web Service 的 WSDL 文档用 WSDL2OWLS 工具导入转换成 OWL-S 文档。测试组装只有顺序结构的组合服务采用了第三方发布的 Web Service,其 WSDL 文档的 URL 为:http://www.bs - byg.dk/hashclass.wsdl。该 WSDL 提供了两个方法:HashString 和 CheckHash,前者用指定的编码方式(MD5、SHA1 等等)对指定的字符串编码,后者根据指定编码方式检查一个字符串(HashString)是否是原字符串(OriginString)的编码结果。我们将把这两个方法组装成一个服务,用输入的编码方式和待编码字符串先进行编码,然后检查编码的结果是否正确,如果正确返回 true,否则返回 false。HashString 具有两个输入参数:str(表示被编码的原始字符串)、HashType(编码类型)。HashString 有一个输出:HashStringResult(对原始字符串用指定编码类型编码得到的编码结果)。CheckHash 具有三个输入:OriginalString(被编码的原字符串)、HashString(原编码结果)、HashType(编码类型),有一个输出:CheckHashResult(对编码的检测结果)。

首先将 HashString 和 CheckHash 分别导入为 OWL-S 文档,生成的两个 OWL-S 文档分别为 HashString.owl 和 CHeckHash.owl,描述的 Web Service 分别为 HashStringService 和 CheckHashService。现将这两个服务组装成顺序结构的复合服务 comHashCheckService。comHashCheckService 中包含了两个顺序结构的动作节点,这两个动作节点分别表示对 HashStringService 和 CheckHashService 两个 Web Service 的调用。comHashCheckService 还包含两个数据流节点,第一个数据流节点描述了 HashString 动作节点和 CheckHash 动作节点间的数据关系,即将 CheckHash 动作节点关联的 CheckHashService 的其中一个输入参数 HashString 的数据来源设置为 HashString 动作节点关联的 HashStringService 的输出变量 HashStringResult(表示用指定的编码方式如 MD5、SHA1,对指定的字符串的编码结果)。第二个数据流节点描述了动作节点 CheckHash 和 Produce 节点之间的数据流向,即将 CheckHashService 服务对 HashString 的验证结果传给 Produce,也同时指定了复合服务的最终输出结果为 CheckHashResult。

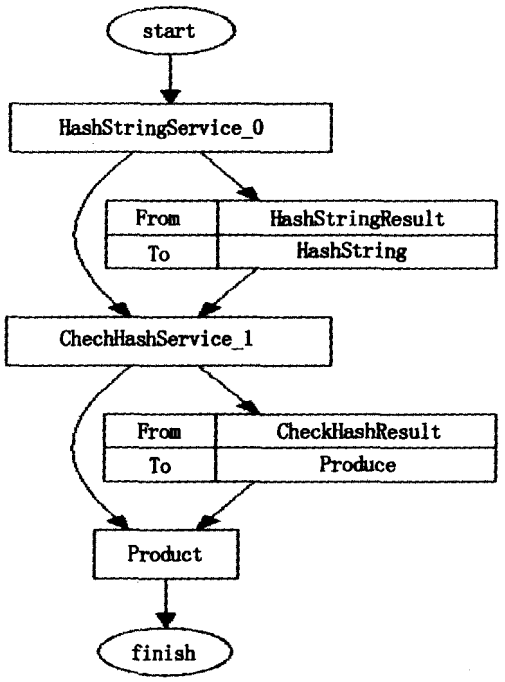

图3 顺序结构复合服务 comHashCheckService 的组装图

组装好复合服务后,将服务导出为 OWL-S 文档保存成磁盘文件 comHashCheck.owl。这时通过刷新服务库,通过打开文件对话框打开 comHashChek.owl,就可以用工具栏中的运行按钮运行组装好的复合服务 comHashCheckService。组装图如图 3 所示,组装图中具有 from、to 的节点是数据流节点,其他节点是控制流节点。

实验证明,SWSComposer 平台能很好的完成网络上共享的 Web 服务的组装和运行。另外测试组装选择结构的实例 IfFindBookPay 所用的原子服务是本地开发的 Web 服务,通过为 Web 服务增加语义注释,使用领域本体的数据配置 IF 条件,由执行引擎根据监控信息来动态的设置知识库的方法,使平台能较好的支持带 If 条件分支结构的复合服务的组装和运行。

7 讨论

本文提出和实现了可视化的语义 Web 服务组合方法,平台用 Graphviz 的 dot 图形形式将组合过程展示

出来,让用户通过操作图形就能轻松的完成复合服务的组合过程,但在服务的发现方面还没有做具体的研究。组合的前提是找到可以组合起来用于完成目标任务的子服务,因此服务的发现和匹配是一项非常重要而关键的工作,其中也会涉及到语义的标注问题,如何将现有的 Web Services 语义进行完整描述,如何实现更精准的语义匹配,以及如何保持组合 Web 服务的语义,这些都可以作为下一步的工作方向。在服务合成算法方面,需要对现有的合成算法进行改进,使其具有更好的效率和性能。平台现在实现的组合功能,需要用户的参与,需要用户制定复合服务的控制流和数据流,在进一步的工作中,需要改进算法,引入新的技术,逐步实现合成过程的半自动化和自动化,尽量合理的安排计算机和人之间的分工,使得系统可以更好地和用户进行协作,更好地完成服务合成的任务。此外,语义 Web 服务合成现在缺乏标准化的、能从设计到实现进行全程支持的 Web 服务合成模型,还缺乏统一的理论模型与相关的合成验证方法,也缺乏有效的手段来监控 Web 服务合成的运行质量[4,5]。这些问题都有待进一步的研究和解决。

参考文献:

[1] 陈　珊,许林英,袁　琳. web 服务综述[J]. 微处理机,2005,26(2):1-3.
[2] 喻　坚,韩燕波. 面向服务的计算——原理和应用[M]. 北京:清华大学出版社,2006.147-148.
[3] Leymann F, Roller D, Schmidt M T. Web services and business process management[J]. IBM System Journal, 2002, 41(2): 198-211.
[4] 顾　宁,刘家茂,柴晓路. web services 原理与开发实践[M]. 机械工程出版社,北京:2006.101-102.
[5] 李景霞. Web 服务组合综述[J]. 计算机应用研究,2005,22(12):4-7.

Design and Implementation of Composition Visualization Platform of Semantic Web Service

LI Ming-cui

(School of Information Engineering, East China Jiaotong University 330013, China)

Abstract: As more and more Web services are shared in the network, in order to make full use of Web service, it is necessary to combine Web services to provide more powerful services. The article focuses on the visual composition technique of semantic Web service, mainly presents the realization principle of a Web Service composition system. Based on ontology, the method of creating composite services by creating target control flow and data flow in a graphic panel is put forward. The methods of creating sequence and choosing compound construction web services are discussed in detail. Examples of assembling the network and local Web services show that the method of the platform is feasible.

Key words: Web Service; semantic; visualization; composition

(责任编辑:王建华)