第32卷 第15期 计 算 机 工 程 2006年8月
Vol.32 № 15 Computer Engineering August 2006
·人工智能及识别技术· 文章编号：1000—3428(2006)15—0191—03 文献标识码：A 中图分类号：TP182

基于语义 Agent 的 Web 服务智能化

陈红英[1,2]，杨宜民[1]，李卫华[1]
(1. 广东工业大学自动化学院，广州 510090；2. 华南师范大学计算机学院，广州 510631)

摘 要：采用了 OWL-S 对 Web 服务进行语义化，构造了若干个 Agent 服务，这些 Agent 服务分别完成需求转换、语义比较、结果合成、调用服务等功能，消除了用户程序对具体服务的依赖性，可以实现服务的较准确的查找和调用，对 Web 服务的智能化和自动化有一定的借鉴作用。

关键词：语义网；Web 服务；Jean；Agent；向量空间法

Using Semantic Web and Agent to Implement Intelligent Web Service

CHEN Hongying[1,2], YANG Yimin[1], LI Weihua[1]
(1. Institute of Automation, Guangdong University of Technology, Guangzhou 510090;
2. Institute of Computer Science, South China University, Guangzhou 510631)

【Abstract】This paper uses multiple methods to implement Web service intelligent. It uses OWL-S to make Web service semantic, then uses several agent services, those agent services implement require transition, semantic comparison, results compose, service transfer. By the way, it can find Web services correctly, and it is useful for intelligent and autoimmunization of Web service.

【Key words】 Semantic Web; Web service; Jean; Agent; Vector space arithmetic

Web 服务由于具有跨平台性和易用性，得到迅猛的发展。目前调用 Web 程序时需要程序员手工在网上查找合适的服务，然后根据此服务的参数，编写客户端应用程序。这些工作都需要人工参与，随着 Web 服务的发展，网络上的 Web 服务数量剧增，如何找到用户需要的服务，如何得到更新的 Web 服务，如何自动调用这些服务，如何对 Web 服务自动监控，能够了解 Web 服务执行的状态，如何整合各个服务实现从而满足用户的复杂需求，实现 Web 服务智能化，成为目前迫切需要研究的问题。

制约Web服务智能化的主要问题在于Web服务的描述文件WSDL，WSDL由以下几部分组成：type(所传输的数据类型)，message(所传输的消息)，porttype(所支持的操作接口)，binding(传输协议)，service(Web服务的位置)。可以看出，WSDL文件缺少语义，当用户面对多个具有相似功能的Web service时，往往不能决定哪一个满足要求。语义网是建立在本体理论上，使计算机能够更多理解网上的信息，从而进行知识发现、数据集成、信息导航等[3]。

本系统的目标是：

(1)使服务成为机器可以解释、用户明了的、能够使用智能主体的个体。

(2)个性化的机器代理能实现自动的 Web 服务发现、执行、组成和互操作。

1 服务的语义化工作

2001 年提出的DAML-S草案是第一个针对Web服务的本体描述语言。它主要包括 3 个部分：服务轮廓(profile)，服务过程模型(process model)，服务基点(grounding)[2]。

其中服务轮廓讲述服务是做什么的，它负责给服务搜索 Agent 提供信息以帮助服务搜索 Agent 决定给服务是否满足其需要。

服务过程模型讲述服务是怎样工作的，这个描述信息可能会被服务搜索Agent用于以下几种用途：(1)更深入分析该服务是否满足其需要；(2)为了完成特定的任务从多个服务中集成若干服务；(3)在服务的设定过程中调整不同参与者的活动；(4)监控服务的执行[2]。

服务基点定义了一个 Agent 如何可以访问到一项服务的细节。典型的服务基础会详细说明通信协议(RPC,HTTP-FORM,CORBA IDL,SOAP,Java RMI) 以及在联系服务的过程中会用到的端口号等细节。除此之外，服务基点必须详细说明服务模型中定义的每个抽象类型。

总的来说，服务概况提供 Agent 发现服务所需要的信息，服务模型和服务基础与服务联系在一起提供给 Agent 足够的信息以便于其使用服务。随 DAML-S 的发展出现了 OWL-S，OWL-S 包含一套本体，提供描述 Web 服务的词汇表，描述服务的语义，能够根据服务的要求和效果进行推理。

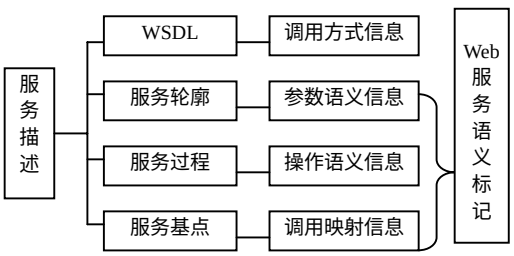

图 1 语义化后的服务所具有的文件描述

要实现服务的智能化，首先必须为服务添加语义信息，在 WSDL 的基础上，添加如上所述的服务文件作为服务的附

基金项目：广东省自然科学基金资助项目(032496)
作者简介：陈红英(1969—)，女，讲师、博士，主研方向：Agent, GIS, Web service；杨宜民，教授、博导；李卫华，教授、博士
收稿日期：2005-12-27 **E-mail：**mgfchy@sohu.com

加文件，供用户端查询，具有语义的服务端文件如图 1 所示。

2 采用 Agent 服务实现智能化服务的流程

系统实现如图 2 所示，在用户程序和服务之间增加代理，由代理完成：服务的查找，更新，自动执行等操作，其中每个 Agent 可以定义为一个 Web 服务，分别完成不同的任务，供用户程序调用。

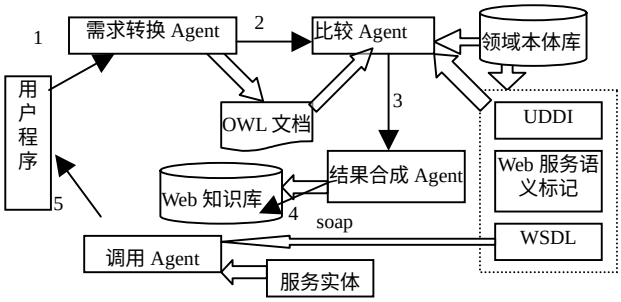

图 2　Agent 动态调用服务的原理图

具体流程如下：

(1)用户程序提交请求程序，该程序是按照需求转换 Agent 要求的规范编写，该程序调用需求转换 Agent 服务，需求转换 Agent 服务利用 Jean 生成该用户程序接口的 OWL 语义描述，同时通知比较 Agent 服务。

(2)比较 Agent 服务将由用户程序生成的 OWL 文档与领域本体库知识结合，得到更全面的用户需求的 OWL 文档，然后与 Web 服务语义标记进行相似性比较，从而找到更加符合用户需求的服务，这里 OWL 文档与领域本体库知识的比较，与 Web 服务语义标记进行的相似性比较都是用 Jean，相似性比较采用了 OWL 推理和用向量空间法。

(3)结果合成 Agent 将比较 Agent 找到的 Web 服务的信息(wsdl，语义标记)进行比较,选取相似性大的放入 Web 知识库，同时通知调用 Agent。

(4)调用 Agent 访问 Web 知识库，依次找到自己所需要的 Web service 的 wsdl，进行访问，并将返回结果提交给用户程序。

3 技术要点

3.1 利用 Jena 实现 OWL 文件的语义的提取、查询、翻译

Jena 是 HP 实验室的一个专门针对基于 RDF,OWL 的语义网的第 2 代 JavaAPI,它提供了以 RDF API 为核心的语义 Web 工具集。Jean2.1 版本以上支持 OWL-S 解析，创建和查询，推理。利用 jean 可以将用户提交程序中的语义部分转换为标准的 OWL 文档。

Jena的调用方法[4]，下面是一个采用Jena向一个描述student信息的OWL文档添加内容，取OWL文档的关键属性，查询的例子:

(1)将语句添加到一个 OWL 文档(外部文档 OWL 文档)

```
Model mymodel
String namespace=http://www.test.com;
 mymodel.createResource=(http://www.myWeb.com/student);
.addProperty(mymodel.createProperty(namespaces,"height"),175)
.addProperty(mymodel.createProperty(namespaces,"age"),23)
```

(2)从 OWL 文档得到 OWL 的属性(OWL 文档外部文档)

```
Model mymodel;
StmtIterator myiterator;
Statement mystatement;
myiterator =mymodel.listStatements();
while(myiterator.hasNext()){
mystatement = myiterator.next();
}
```

(3)在 OWL 文档中查找信息

```
Model mymodel;
Resource myresource;
ResIterator myiterator；
myiterator=mymodel.listSubjectsWithProperty(mymodel.createProperty(http://myWeb.com/desination),"graduated");
While(myiterator.hasNext()){
myresource = myiterator.next();
System.out.println("graduated student:"+r.toString());
```

3.2 利用 OWL 的可转换规则进行 OWL 文档相似性推理

当得到两个本体时，有时这两个本体从表面上看是不相同的，但是可以通过推理，发现两个本体的相似性。可转换规则[3]如表 1。

表 1　可转换规则

若本体中包含	则加入
xxx aaa yyy	xxx rdf:type rdfs:Resource aaa rdf:type rdf:Property yyy rdf:type rdfs:resource
aaa rdfs:subPropertyOf bbb bbb rdfs: subPropertyOf bbb	aaa: subPropertyOf ccc
aaa rdfs:subPropertyOf bbb	aaa rdf:type rdf:Property bbb rdf:type rdf:Property
xxx aaa yyy aaa rdfs:subPropertyOf bbb	xxx bbb yyy
xxx aaa yyy aaa rdf:domain zzz	xxx rdf:type zzz
xxx aaa uuu aaa rdf:domain zzz	xxx rdf:type zzz
xxx aaa uuu aaa rdf:range zzz	uuu rdf:type zzz
zzz rdf:type uuu	uuu rdf:type rdfs:class
xxx rdf:type rdfs:class	xxx rdfs:subclassof rdfs:resource
xxx rdfs:subClassOf yyy	xxx rdf:type rdfs:Class
xxx rdfs:subClassOf uuu	uuu rdf:type rdfs:Class
xxx rdfs:subClassOf yyy yyy rdfs:subClassOf zzz	xxx rdfs:subClassOf zzz
xxx rdfs:subClassOf yyy aaa rdf:type xxx	aaa rdf:type yyy
aaa rdf:type rdfs: Constrain Resource aaa rdf:type rdf:Property	aaa rdf:type rdfs:constraint Property

3.3 基于 xml 的 FIPA-ACL 实现各个 Agent 之间消息的传送

由于Agent的传递可能是跨平台的，同时为了维护消息的统一性和较好的语义，将Agent通信语言acl包装成xml形式，在各个Agent之间传送。基于XML的Agent通信的层次结构如图 3 所示[4]。

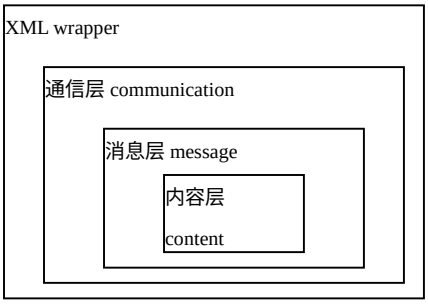

图 3 基于 XML 的 Agent 通信的层次结构

下面给出一个 FIPA ACL 的 XML Schema：

```
…
<elementtype name=”sender” content=”textonly” dt:type= ”string”/>
<elementtype name=”receiver” content=”textonly” dt:type = ”string”/>
<elementtype name =”in-reply-to” content=”textonly” dt:type =”string”/>
<elementtype name =” to” content=”textonly” dt:type=”string”/>
<elementtype name =”from” content=”textonly” dt:type =”string”/>
<elementtype name =”reply-with” content=”textonly” dt:type =”string”/>
<elementtype name =”content” content=”textonly” dt:type =”string”/>
<elementtype name =”language” content=”textonly” dt:type =”string”/>
<elementtype name =”ontology” content=”textonly” dt:type =”string”/>
….
```

该 XML Schema 表示了 ACL 的基本消息。

3.4 利用向量空间法进行 OWL 文档相似性的比较

由于可以用 Jena 将 OWL 文档中的属性解析出来，采用向量空间法对两个文档的属性进行相似性比较。

设 OWL 文档 W 有 WCOUNT 个属性，OWL 文档 S 有 SCOUNT个属性，设W_j为OWL文档W的第j个的属性，S_j为OWL文档S的第j个属性,首先取两个文档的最大属性集$=\{W_1,W_2,W_3,\ldots W_{WCOUNT},S_1,S_2,\ldots S_{SCOUNT}\}$,集合中去掉相同的项目后为 RCOUNT 个元素，属性集 SUM=$\{\ldots W_i,W_j\ldots S_i,\ldots,S_j\}$。设置文档W的向量w’为RCOUNT个元素，表示为{1，10，0}，其中属性集SUM中的元素下标为x的元素如果出现在W文档中则该向量下标x的元素置为 1，否则置为 0，同样设置S文档的向量S’，然后采用向量的余弦来衡量两个文档的相似度，表示为

$$\cos(w, s)=\Sigma s'_j w'_j/|s'_j||w'_j|$$

最后的取值 cos(w, s)∈(0，1)，值越大表示文档越相似。

4 结束语

本文采用多种方法实现基于语义的 Web 服务智能化。首先采用 OWL-S 对 Web 服务进行语义化，然后构造了若干个 Agent 服务，这些 Agent 服务分别完成需求转换、语义比较、结果合成、调用服务等功能，具体功能如下：

(1)需求转换 Agent:利用 Jean 将规范化的用户程序的接口的语义转换为 OWL 文档。

(2)语义比较 Agent: 将由用户程序生成的 OWL 文档与领域本体库知识结合，得到更全面的用户需求的 OWL 文档，然后与 Web 服务语义标记进行相似性比较。具体过程是：将从用户需求的 OWL 文档和 Web 服务语义标记中分别提取的属性用向量空间法进行相似性比较。

(3)结果合成 Agent：将相似性大于某个值的 Web 服务的 wsdl 和语义标记存入知识库。

(4)调用服务 Agent:从知识库中逐一调用 Web 服务。

各个 Agent 之间的通信采用基于 XML 的 acl 语义。系统通过引入 Agent 消除了用户程序对具体服务的依赖性，可以实现服务的较准确的查找和调用，对 Web 服务的智能化和自动化有一定的借鉴作用。

参考文献

1 赵慧杰, 沈建京. 基于OWL的Web服务构件研究[J]. 计算机应用, 2005, 25(3): 634-636.

2 张大陆, 刘　畅. Web 服务语义描述的架构[J]. 计算机工程, 2004, 30(2): 124-128.

3 宋　炜, 张　铭. 语义网简明教程[M]. 北京: 高等教育出版社, 2004.

4 Wooldridge M. Verifiable Semantics for Agent Communication Languages[C]. Proc. of the 3rd Int. Conf. on Multi-agent Systems. IEEE Computer Society Press, 1998.

5 韩贵来. 基于 Agent 的语义网格通信研究[D]. 广州: 广东工业大学, 2005.

~~~~~~

（上接第 190 页）

(1)给出故障现象：Engine Start=Abnormal；运行决策。

(2)根据决策结果 1，执行观测：ETO=no；继续决策。

(3)根据决策结果 2，执行观测：Light shine=work；继续决策。

(4)根据决策结果 3，对 Starter 执行维修，维修后 starter=good；汽车能够正常启动。

4 结论与将来工作

本文应用贝叶斯网络来对设备故障维修提供决策支持，着重论述了如果在不确定的条件下对维修过程中的观测和维修动作进行低代价决策的决策理论。

传统的维修系统，维修过程只是很简单的两个阶段：选择最有可能的故障然而进行对应的修理。针对上述原因，我们的研究主要集中到将诊断和修理过程融合到一起，诊断和修理动作交替进行，从而使系统尽可能地提高效率(减少时间、费用等代价)，达到快速维修的目的。通过本实例以及大量的前期工作[5]证实了采用上述决策该故障检修的决策理论的确能够提供快速指导维修工作。

由于在上述决策过程中，我们对观测维修节点赋给的代价值均是常数，然而在实际过程中，随着维修工作的开展代价在不同阶段代价将变化，下一步将引入动态代价函数。

参考文献

1 Breese J S, Heckerman D. Decision-theoretic Troubleshooting: A Framework for Repair and Experiment[R]. Technical Report MSR-TR-96-06, Microsoft, 1996.

2 Heckerman D, Breese J S, Rommelse K. Troubleshooting Under Uncertainty[C]. Proceedings of the Fifth International Workshop on Principles of Diagnosis, 1994.

3 Basics of Knowledge Engineering[Z]. http://freelock.com/technical/KE.pdf, 1999.

4 Haddawy P. An Overview of Some Recent Developments in Bayesian Problem Solving Techniques[J]. AI Magazine Special Issue on Uncertainty in AI (Summer), 1999.

5 Heckerman D, Breese J S, Rommelse K. Decision-theoretic Troubleshooting[J]. Communications of the ACM, 1995, 38(3): 49-57.

6 Kadie C M, Hovel D, Horvitz E. MSBNx: A Component-centric Toolkit for Modeling and Inference with Bayesian Networks[R]. Technical Report, MSR-TR-2001-67, 2001.
~~~~~~