基于语义的QoS感知Web服务发现机制

王晓峻[1] 毛莺池[2] 钱国锋[2]
(河海大学水利水电学院 南京 210098)[1] (河海大学计算机与信息学院 南京 210098)[2]

摘 要 随着提供相同功能的Web服务数量的日益增多,服务质量(Quality of Service,QoS)成为用户选择Web服务的重要考虑因素。目前,通过对服务QoS属性在语法层匹配来提供Web服务选取的机制不能很好地满足复杂QoS属性匹配的要求。研究了基于用户QoS需求偏好,将用户需求的QoS与候选服务的QoS进行语义比较,结合约束规划(Constraint Programming)方法,在语义层匹配Web服务的QoS属性,选取满足匹配要求的服务,最后对满足QoS属性值约束的候选服务进行优化选择处理,获取最终匹配的候选服务。

关键词 Web服务发现,QoS感知,约束规划,语义Web
中图法分类号 TP393 **文献标识码** A

QoS-aware Web Services Discovery Scheme Based on Semantic

WANG Xiao-jun[1] MAO Ying-chi[2] QIAN Guo-feng[2]
(College of Hydraulic and Hydropower Engineering, Hohai University, Nanjing 210098, China)[1]
(College of Computer and Information Engineering, Hohai University, Nanjing 210098, China)[2]

Abstract With the increase of Web services with the similar or same functionality, QoS (Quality of Service) has become the important criteria for users to select the appropriate Web services. At present, most approaches to the QoS-based Web services selection suffer from the purely syntactic matchmaking, which could not meet the requirements of complicated QoS matchmaking in semantic level. In this paper, we proposed a based on semantic QoS-aware Web service discovery scheme. Initially, a QoS ontology was presented to define QoS data into service descriptions. Then, the ontology reasoning was adopted to change previous syntactic matchmaking into a semantic way. Through confirming the compatibility of concepts, complicated QoS conditions were solved as constraints via constraints programming. The optimization algorithm based on QoS was proposed to obtain the appropriate service to users. Finally, case study was given to illustrate the effectiveness of discovery scheme.

Keywords Web services discovery, QoS aware, Constraints programming, Semantic Web

1 引言

Web服务是面向服务的体系结构中一项非常关键的技术,是下一代网络的核心技术之一[1]。在高度异构、自治且分布式的环境中,单纯的Web服务功能性描述已经不能满足实际应用需求[2]。此外,能提供相似功能却具有不同非功能属性(如安全性、传输率等)的Web服务在不断增加,这需要有更为成熟的发现过程以匹配用户的需求[3]。利用语义Web的相关技术对Web服务进行语义描述,可以在语义层为Web服务的匹配及发现提供很好的支持,使Web服务之间能够互相理解对方的功能、内容以及属性,并且为Web服务的自动发现、组合、执行、监控提供了技术基础[4]。

QoS感知的语义Web服务发现主要是从满足服务功能性需求的候选服务集中,按照用户的QoS需求,通过匹配处理,获取满足用户QoS需求的服务。现有的QoS感知的服务发现方法大多数都是通过对QoS进行语法层的匹配来实现的,没有充分考虑QoS属性的语义信息,导致服务发现的结果不够全面。此外,在进行QoS感知的语义服务发现时,需要考虑用户的个性化需求(如主要考虑服务的价格是否较低,而对服务的安全性要求不高)以及服务基本可执行条件(如服务的价格虽然很低,但是其延迟时间超过了一定的阈值,对用户而言这种服务基本上是不可执行的,那么此服务作为服务发现的结果将毫无意义),以获取最终匹配的服务。

目前,基于服务功能性需求的服务发现研究成果已有很多,本文主要研究基于服务的非功能属性(QoS)的服务发现方法,故所提及的候选服务(集)均假设为已经满足服务的功能性需求。本文研究基于用户QoS需求偏好,利用5种不同度量方式来区分QoS参数;将用户需求的QoS与候选服务的QoS进行语义比较;结合约束规划(Constraint Programming)方法,在语义层匹配Web服务的QoS属性,选取满足匹配要求的服务。

到稿日期:2009-09-05 返修日期:2009-12-09 本文受河海大学自然科学基金项目(编号:2008428911)资助。
王晓峻(1974-),男,博士生,主要研究方向为Web服务计算,E-mail:mao_yingchi@yahoo.com;毛莺池(1976-),女,博士,讲师,主要研究方向为分布式计算;钱国锋(1984-),男,硕士生,主要研究方向为语义网、Web计算。

2 相关工作

文献[5]在假设服务提供商和用户都认可同一 QoS 上层本体的情况下，通过将 QoS 属性匹配分离为单个基本的 QoS 属性匹配来简化匹配过程，但是该方法并没有考虑定性与定量的 QoS 属性在度量方式上的不同。文献[6]提出了一种扩展 WSMO 本体，将 QoS 条件添加到服务供求双方来进行匹配操作以支持 QoS 感知的服务发现。但是该方法仅限于“≤”，“≥”，“=”3 种运算符，不能很好地满足用户个性化需求。文献[7]基于本体和协调器(mediators)，提出了一种半自动化评价非功能属性需求的技术。该技术利用规则和推理机进行 Web 服务的自动推理和匹配操作，并设置了两种不同的指示变量来衡量供需双方的匹配程度，然后利用这两个指示变量的计算值来管理下一步的过滤、排序和选择操作。但该方法要求在运行之前进行必要的设置，且涉及的本体可扩展性比较差。文献[8]将 QoS 约束构建到服务描述中以获取合适的候选服务，但是他们描述 QoS 时并未选择通用的语义 Web 描述框架，而是采用自己的描述语言，而且约束问题也不是通过形式化的约束满足机制来解决的，因此此方法不具有通用性。文献[9]提供了一种分段合成的方法，它将整个发现过程分成若干不同阶段(如语义发现阶段、QoS 处理阶段)，通过使用约束规划方法将发现问题转化为约束满足问题。但是，其 QoS 阶段的处理没有考虑 QoS 语义。文献[10]提出了一个逐层逼近的三层匹配筛选法，以缩小匹配空间，减少发现的开销。该方法采用描述逻辑(Description Logic，DL)推理机来实现匹配。然而实际情况中，在面对复杂的 QoS 属性需求时，DL 推理机是无法满足需求的。文献[11]提出了一种 Web 服务语义匹配方法，并在此基础上分析了基于语义匹配和基于 QoS(服务质量) 的服务选择策略，以及面向原子服务和组合服务的服务选择方法。但是，文献[10,11]方法都没有考虑用户的个性化需求。文献[12]在本体基础上，提出了一个综合相似度模型，通过两步选择，计算 QoS 语义相似度和数值相似度来获得综合相似度，并以此指导 Web 服务的选择。但是，在 QoS 综合相似度模型中，该方法用二次曲线进行拟合的数值相似度函数存在一定片面性。

QoS 感知的服务发现方法是一种基于语义的服务发现方法，通过语义层的匹配来获取服务发现结果。由于实际情况中，QoS 属性有定性和定量之分，因此在进行 QoS 感知的服务发现之前必须建立相应的能够区分定性与定量 QoS 属性的本体。在进行 QoS 匹配时，很多方法采用描述逻辑推理。但描述逻辑无法对数学操作符进行推理，在面对复杂的 QoS 需求时，基于描述逻辑的匹配处理就显得无能为力[13]。大多数服务发现机制以服务 QoS 匹配的程度来决定最终服务发现的结果，而这种方法一方面忽视了用户的个性化需求，使得最终所得结果未必是贴近用户需求的结果；另一方面，由于绝大多数用户对 QoS 的所有属性并不是很了解，很有可能只是对他们熟悉的 QoS 属性提出了需求，而忽略了一些重要的但是不熟悉的，使得最终获取的服务未必真正接近用户需求(或者根本不可执行)。

因此，本文基于用户 QoS 需求偏好，利用 5 种不同度量方式来区分 QoS 参数；将用户需求的 QoS 与候选服务的 QoS 进行语义可比性判断；结合约束规划(Constraint Programming)方法，在语义层匹配 Web 服务的 QoS 属性，选取满足匹配要求的服务。

3 QoS 感知的 Web 服务发现机制描述与分析

在研究总结当前 QoS 感知的语义 Web 服务发现相关技术和方法的基础上，针对现有方法中存在的不足，本文提出了基于语义的 QoS 感知 Web 服务发现机制。该机制首先通过扩展 OWL-S 本体，设计建立 QoS 本体模型，统一服务的 QoS 数据描述。然后，对 QoS 进行匹配处理，判断候选服务是否满足领域专家指定的基本 QoS 需求，对候选服务集进行过滤选择，将满足领域专家指定的基本 QoS 需求的服务作为新的候选服务。最后，根据用户的个性化 QoS 需求，对 QoS 优化选择处理，以获取最终匹配的服务。

3.1 QoS 本体设计

(1)QoS 本体模型——OWL-UQ

建立 QoS 本体是 Web 服务发现机制的基础。综合考虑多种 QoS 模型的优缺点，在文献[12]本体建立方法的基础上，考虑到定性和定量 QoS 不同的度量方式，扩展 OWL-S 服务本体，构建一个包含核心 QoS 参数概念的简单 QoS 本体模型——QoS 的上层本体 OWL-UQ(OWL-S Upper QoS Ontology)。

OWL-UQ 本体模型核心部分如图 1 所示。该本体模型除了新增 CompositeQoSParamete 以及 Function 核心概念外，保留了文献[12]中的大部分核心概念。本文主要的改进包括：CategoryName 是用来描述 QoS 参数的类别的，而非简单的参数名称；用 5 种不同的度量类型取代文献[12]的 3 种度量类型，以期更好地区分定性与定量 QoS 属性；Tendency 的取值不但包含了文献[12]中的“越大越好”、“越小越好”，还增加了“期望区间”来表示用户需求是一个区间值的情况。

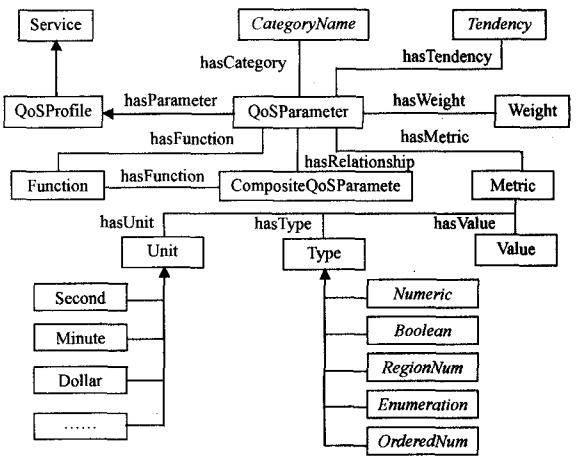

图 1 OWL-UQ 本体模型核心部分

Metric 类用来描述 QoS 参数的度量方式。包含 3 个子类：Type，Unit，Value。Type 表示参数度量类型，分为 5 种：数值型(具体的数值)、数值区间型(区间内的某一数值)、布尔型、枚举型(有限值的集合)以及有序集合型(有限的有序值集合，如一组描述安全性的值)。数值型表示为具体单值；数值区间型表示为由上下界构成的区间值；布尔型取值为 True 或者 False；枚举型即用它本身的值；有序集合型用其原有有序值的集合。Unit 表示该参数的度量单位，它也由一系列子类构成，且与转换函数类相关联，用以进行度量单位间的进制转

换。Value 表示参数的值，参数的取值与其类型相对应。

属性 hasCategory 描述该参数所属类型名称，如安全性、可获取性等。属性 hasWeight 用来表示该参数的重要性(即权重，本文取值限制在 0 和 1 之间，所有权重总和为 1)。属性 hasTendency 用来描述用户期望该参数的取值趋势，取值趋势包括 3 种情况：越高越好(如安全性)、越低越好(如价格)、期望区间。如果没有明显的取值趋势，则置为空。属性 hasMetric 用以描述该参数的度量方法。本模型支持数值型、数值区间型、布尔型、枚举型以及有序集合型 5 种度量方式。属性 hasRelationship 用来描述 QoS 参数与其他参数是有关联的。而属性 hasFunction 用来描述复合 QoS 参数之间具体是如何复合的。

(2)OWL-UQ 本体与 OWL-S 本体的关系

OWL-S 本体可以通过扩展 ServiceProfile 中的 serviceParameter 属性和 ServiceParameter 类来定义其他与 ServiceProfile有关的内容。sParameter 属性的值域 Thing 就是可扩展的部分。OWL-UQ 本体以子属性的形式继承了 serviceParameter 和 sParameter 两个属性，以此与 OWL-UQ 本体的核心类 QoSProfile 相关联。因此，OWL-QoS 对 OWL-S 的扩展是符合 OWL-S 规范的。处理 OWL-S 本体的应用在处理含有 OWL-UQ 本体对象的服务描述文件时，不会产生兼容性问题。

3.2 QoS 匹配处理

QoS 匹配处理主要分 3 步：首先，在 QoS 领域本体库的支持下，由推理引擎对候选服务的 QoS 数据与需求的 QoS 数据进行语义可比性的判断，找出与用户需求语义上可比的候选服务，防止出现仅仅在语法层对 QoS 匹配处理的情况。然后，利用约束规划方法对具有语义可比性的 QoS 条件进行转化，将 QoS 值匹配问题转化为约束满足问题进行解决。最后，利用现有的约束满足求解器来判断候选服务集的 QoS 值的约束满足情况。

(1)QoS 语义可比性判断

本文借鉴服务功能性匹配中语义匹配的方法，进行 QoS 语义可比性的判断，即利用 DL 推理进行语义可比性判断，确保候选服务中的 QoS 参数与用户需求的 QoS 参数在语义上是可比较的，避免在语法层进行 QoS 匹配处理。

假设请求服务的 QoS 数据为 $Q_R=\{q_{R1},q_{R2},\cdots,q_{Rn}\}$，其中 $q_{R1},q_{R2},\cdots,q_{Rn}$ 分别表示请求服务的 QoS 参数，候选服务的 QoS 数据为 $Q_C=\{q_{C1},q_{C2},\cdots,q_{Cm}\}$，其中 $q_{C1},q_{C2},\cdots,q_{Cm}$ 分别表示候选服务的 QoS 参数。语义可比性的形式化定义如下。

定义 1(QoS 参数可比性) QoS 参数 $q_{Ri}\in Q_R$，QoS 参数 $q_{Cj}\in Q_C$，当且仅当在 QoS 本体中 q_{Ri} 和 q_{Cj} 满足下列条件之一时，称 q_{Ri} 与 q_{Cj} 是语义可比的，并记为 $q_{Ri}\Leftrightarrow q_{Cj}$：

①$q_{Ri}\equiv q_{Cj}$；

②$q_{Ri}\subseteq q_{Cj}$ 或者 $q_{Cj}\subseteq q_{Ri}$；

③q_{Ri} 由 $Q_R=\{q_{R1},q_{R2},\cdots,q_{Rn}\}$ 经复合函数 f 组合而成，其中对于 $\forall q_{Rk}\in Q_R$，$\exists q_{Cj}\in Q_C$，使得 q_{Ri} 和 q_{Cj} 满足①和②中的任一条件。

在定义 1 中，$q_{Ri}\equiv q_{Cj}$ 是指在 QoS 本体描述中，q_{Ri} 和 q_{Cj} 两个概念是相同的或者语义等价的；$q_{Ri}\subseteq q_{Cj}$ 是指概念 q_{Ri} 是概念 q_{Cj} 的子概念(子类)；复合函数 f 则表示复合 QoS 参数的 hasFunction 属性所对应的复合函数，描述该复合函数是如何通过其它 QoS 参数构成复合 QoS 参数的。

定义 2(候选服务与请求服务的 QoS 参数可比性) 当且仅当对于 $\forall q_{Ri}\in Q_R$，总有 $\exists q_{Cj}\in Q_C$，满足 $q_{Ri}\leftrightarrow q_{Cj}$，则称 Q_R 与 Q_C 是语义可比的，记为 $Q_R\Leftrightarrow Q_C$。

从定义 2 看出，Q_R 与 Q_C 之间的语义可比性依赖于构成其 QoS 参数之间的语义可比性。定义 2 可以解决一对一概念间的可比性判断问题，如“价格”与“售价”之间的语义可比性判断。若遇到某些特殊情况(如复合 QoS 参数)，一对一的可比性判断方法就无能为力，如“可获取性”与“平均无故障时间”及“平均修复时间”(可获取性 = 平均无故障时间/(平均无故障时间 + 平均修复时间))之间的语义可比性判断。

定义 1 可以很好地解决 QoS 复合参数以及概念间的包容问题(如“汽车”与“小轿车”)。通过不断分解复合 QoS 参数，判断分解后的 QoS 参数是否满足语义可比性来确定复合 QoS 参数是否具有语义可比性。只要分解后的所有 QoS 参数都满足语义可比性定义，那么复合 QoS 参数也与候选服务的 QoS 语义可比。

(2)约束转化

文献[9]提出利用约束规划思想来解决 QoS 参数值匹配问题的方法，该方法将 QoS 的值约束问题转化为一系列的约束，然后利用约束满足求解器对这一系列约束进行判断，以获取 QoS 匹配的服务。但是该方法的约束转化是基于语法的转化，不能很好地满足用户个性化需求。本文提出在语义可比性判断后，利用 OWL-UQ 本体，将 QoS 条件转化为一系列约束。这样，QoS 参数值匹配问题就转化为约束满足问题，再利用约束满足求解器，找出候选服务集中参数值满足需求的服务。

定义 3(约束满足问题定义[9]) 一个约束满足问题是一个 $p=\langle V,R,C\rangle$ 三元组，其中 $V=\{v_1,v_2,\cdots,v_m\}$ 表示 m 个非空变量集合；$R=\{r_1,r_2,\cdots,r_m\}$ 表示 m 个变量的非空有限值域集合，即 r_i 是 v_i 的可能取值的集合，$1\leqslant i\leqslant m$；$C=\{c_1,c_2,\cdots,c_n\}$ 表示有限约束集合。p_{ck} 是一个 $\langle V_{ck},R_{ck},C_{ck}\rangle$ 元组，其中 $V_{ck}=\langle v_i,v_{i+1},\cdots,v_j\rangle\subseteq V$，$R_{ck}=\langle r_i,r_{i+1},\cdots,r_j\rangle\subseteq R$，$Cck$ 表示一个约束，每个约束定义于 V 的有限子集中变量取值的组合。

定义 4(约束满足问题解定义) 约束满足问题的解是为每个变量赋一个对应论域中的值，使之满足集合 C 中的所有约束。

本文充分利用 QoS 本体来辅助 QoS 条件转化过程。QoS 本体中的 QoSParameter 类可以映射为约束中的变量 v，QoS 参数度量类型(Type)对应区间 r。值的约束根据 5 种不同的度量类型来进行转换，使得所有约束均表示为数值上的约束(将定性的 QoS 参数值转化为数值)：

①布尔型：TRUE/FALSE 映射为 1/0。

②数值型和数值区间型：对应 Value 类中现有的数值和区间数值，无需变动。

③枚举型：根据需求来调整候选服务中的有限值集。将具有语义可比的候选服务的值与需求中的值进行比对，出现与否分别对应 1/0，参数 v 的比较就变成了检查相应的值集合是否相同。如一个书籍预订的服务要求“书籍的版式取值是{精装，普通，影印，做旧}”，候选服务提供的版式为{简装，

精装，做旧}，那么调整后候选服务的取值变为{1,0,0,1}。

④有序集合型：将所有可能的取值排序并量化到一个等间距的数值区间上，然后将具体的 QoS 参数取值映射到此区间，用一个区间值来表示其取值情况。例如安全性值为{很高，高，一般，低，很低}，那么映射转化后为{5,4,3,2,1}。这样描述安全性 v 比一般程度要高些就可表示为 v 取值[3,5]。

通常情况下，假设有两个 QoS 参数 q_{Ri} 和 q_{Cj}（其中 q_{Ri} 是请求服务的 QoS 参数，q_{Cj} 是候选服务 QoS 参数），两个约束变量 v_i 和 v_j 分别与这两个参数对应，如果 q_{Ri} 与 q_{Cj} 是语义可比的，那么这两个约束是等价的，即 $v_i=v_j$。

在处理复合 QoS 参数时，通过重写来分解复合 QoS 参数约束，如可获取性（Availability），表示如下：

$$c_1=\langle V_{c1},R_{c1},C_{c1}\rangle=\langle\{Availability\},\{[0,1]\},\{Availability\geqslant 0.8\}\rangle$$

$$c_2=\langle V_{c2},R_{c2},C_{c2}\rangle=\langle\{Availability,MTTF,MTTR\},\{[0,1],[0,+\infty],[0,+\infty]\},\{Availability=MTTF/(MTTF+MTTR)\}\rangle$$

式中，Availability 表示可获取性，MTTF 表示平均无故障时间，MTTR 表示平均修复时间。

约束转化处理过程如下：首先取 QoS 条件中的一个 QoS 参数（QoS 条件包括请求服务 QoS 与候选服务 QoS），判断其是否为复合参数。如果是复合参数，则用构成该参数的所有参数取代其自身；然后，判断该 QoS 参数的度量类型，按照上述提及的转化方法，对度量类型为布尔型、枚举型或者有序集合型的 QoS 参数进行值的转化。循环进行上述过程，直到所有 QoS 条件均转化完毕。

(3)约束满足判断

完成约束转化后，利用现有的约束满足求解器，判断候选服务与请求服务 QoS 值的一致性程度。若候选服务对应的约束满足问题的解都是请求对应的约束满足问题的解，则说明该候选服务的 QoS 值与用户需求的 QoS 值是一致的。约束满足一致性判断定义如下[9]。

定义 5（约束满足问题一致性定义） 在判断约束满足时，C_C 表示候选服务约束，C_R 表示请求服务的约束，当且仅当满足 $satify(C_C,\rightarrow C_R)=\varnothing$ 时，称候选服务的 QoS 值与用户需求的 QoS 值是一致的。

我们设定了 4 个不同的等级来表示 QoS 值匹配情况的判断结果，分别是完全满足、插入满足、交叉满足、完全不满足。“完全满足”是指候选服务的 QoS 参数值能完全满足需求的 QoS 值要求，甚至超出需求所期望的要求。“插入满足”是指候选服务的 QoS 参数值比需求的 QoS 参数值要稍微宽松些，在某些时候，可能导致不满足。例如需求 QoS 为“延迟时间在 1～3 秒间”，候选服务 QoS 为“延迟时间在 1～4 秒间”，此时，该服务与需求的 QoS 之间是插入满足关系。“交叉满足”是指候选服务的 QoS 参数值与需求的 QoS 参数值含有部分共同的取值。“完全不满足”是指候选服务 QoS 参数值完全不能满足需求 QoS 参数值的要求。判断为“完全满足”的服务集合将传递 QoS 优化阶段，以获取最终选择的服务。

3.3 QoS 优化选择

为获取最为贴近用户需求的 Web 服务集，对经过 QoS 匹配处理并判断为“完全满足”的服务集作进一步优化选择处理。首先对候选服务的 QoS 参数值进行预处理，将用区间值表示的 QoS 参数值转化为具有代表性的单值数值，去除布尔型和枚举型的 QoS 参数值。然后构造矩阵，并对矩阵进行规约化处理，消除取值区间不同带来的问题。最后将矩阵与用户 QoS 参数的权值相乘，按照所得结果对服务候选集进行排序，并最终输出给用户。

(1)预处理

鉴于不同的 QoS 参数值可能具有不同的类型和取值范围，采用规约化方法，将 QoS 参数值都转化为 0～1 之间的数值。但是，如果 QoS 参数值是一个区间值，直接规约化操作是不可能的，因此，转化前需要对 QoS 参数值做预处理。QoS 参数可能是复合型的参数，或者同类 QoS 参数的度量单位各不相同，也需要做相应的预处理。本文以用户的 QoS 参数需求为样本，按照复合型参数的复合函数计算后的 QoS 参数值，并按照单位转换公式，对与用户 QoS 参数度量单位不统一的候选服务的 QoS 参数度量单位进行转换。

在 QoS 匹配处理中，已经验证布尔型和枚举型 QoS 参数满足值约束。该参数不具备取值趋势，因此对优化选择无影响。在进行预处理时，以 QoS 本体为基础，以每个 QoS 参数的 hasTendency 属性为参考，选取最坏情况作为该参数值代表，防止预处理结果失真。

单值的数值型无需进行转化。对于数值区间型的 QoS 参数值，如 $a\leqslant q_{Rij}\leqslant b$，如果取值趋势为“越高越好”，则取下限 a 作为该参数代表；如果取值趋势为越低越好，则取上限 b 作为代表。如果取值趋势为期望值区间，则取中值 $(a+b)/2$ 作为代表。对于有序集合型的 QoS 参数，由于在约束转化过程中已转化为数值区间型表示取值，故处理方法与数值区间型的参数处理方式一样。这样，每个 QoS 参数值都是单值数值形式。利用该单值数值构成一个矩阵。

假设有 n 个候选服务通过了 QoS 匹配处理，用户需求的 QoS 参数为 m 个，即 $Q_R=\{q_{R1},q_{R2},\cdots,q_{Rm}\}$，则预处理后 QoS 参数构成的矩阵 M_A 为：$M_A=\begin{pmatrix} q_{R11} & \cdots & q_{R1m} \\ \cdots & \cdots & \cdots \\ q_{Rn1} & \cdots & q_{Rnm} \end{pmatrix}$。

(2)规约化处理

由于每一个候选服务的 QoS 参数都有各自的取值区间，若直接进行加权处理显然不合理。为了消除取值区间不同带来的问题，需对矩阵 M_A 进行规约处理，将 QoS 参数规约到 0～1 之间，使其具有可比性。

根据取值趋势，规约化公式分为 3 类（其中 q_{Rij} 表示 QoS 参数代表值，$q_{\min}$ 表示所有 q_{Rij} 的最小值，$q_{\max}$ 表示所有 q_{Rij} 的最大值）：

①如果 hasTendency 是“越高越好”，规约化处理以 QoS 参数值与最小值之间的差距在整个最小值与最大值间差距上所占比例来计算其规约化后的值，即：

$$q'_{Rij}=\left[1-\frac{q_{\max}-q_{Rij}}{q_{\max}-q_{\min}}\right] \tag{1}$$

②如果 hasTendency 是“越低越好”，规约化处理以 QoS 参数值与最大值之间的差距在整个最小值与最大值间差距上所占比例来计算其规约化后的值，即：

$$q'_{Rij}=\left[1-\frac{q_{Rij}-q_{\min}}{q_{\max}-q_{\min}}\right] \tag{2}$$

③如果 hasTendency 是“期望值区间”，设 QoS 需求的期

望区间为$[E_{start}, E_{end}]$。α表示QoS参数与期望区间中值的所有绝对距离的最大值，即$\alpha=\max\{|q_{Rij}-(E_{start}+E_{end})/2|\}$。$\beta$表示QoS参数与期望区间中间值的所有绝对距离的最小值，即$\beta=\min\{|q_{Rij}-(E_{start}+E_{end})/2|\}$。QoS参数值规约化后的值取区间与$\alpha$的距离在最大值与最小值间距离上所占的比例。

$$q'_{Rij}=(\alpha-|q_{Rij}-(E_{start}+E_{end})/2|)/(\alpha-\beta) \quad (3)$$

通过式(1)－式(3)转换，矩阵M_A中的QoS参数值都在0～1之间取值，这解决了取值区间不同带来的问题，并将规约处理后的矩阵记为M_A'。

(3)加权处理

不同的用户对于服务的需求不尽相同，用户偏好可以通过对请求服务的QoS参数重要性(即权值)设定来体现。参数权值即表示该参数在整个QoS需求中受关注的程度，使用参数受关注程度的百分比表示权值，所有QoS参数的权值总和为1。权值越大说明该参数在需求中的受关注程度越高，即用户越关心该参数的满足情况。

本文将QoS参数的权值向量表示为：$W=\{w_1, w_2, \cdots, w_m\}$，加权处理计算如式(4)所示。

$$M_A''=M_A'\times W=\sum_{i=1}^{n}\sum_{j=1}^{m}(q_{Rij}\times w_j) \quad (4)$$

式中，M_A''为加权处理后的矩阵，可以表示为$M_A''=\{M_1, M_2, \cdots, M_n\}$，若$M_i\in M_A''$，$(1\leqslant i\leqslant n)$，则$M_i$表示第$i$个候选服务的加权处理结果。服务加权处理结果的值较大，说明该服务较贴近用户的个性化需求；相反，服务加权处理结果的值较低，则说明该服务不能很好满足用户个性化需求。

最后，根据矩阵M_A''，对候选服务集进行快速排序。经过排序处理，排在候选服务集中第一的服务最贴近用户个性化需求。

4 实例验证

为了验证提出的语义感知的Web服务发现机制的有效性，用Java开发了一个原型系统，包括3个主要模块：语义可比性判断模块、约束满足判断模块、QoS优化选择模块。本文选取SemWebCentral提供的OWL-S服务测试实例[15]作为基础，验证技术的可行性和有效性。本文研究主要对符合服务功能性需求的候选服务集进行原子服务发现(即非面向组合的服务发现)，因此，假定候选服务集都满足服务的功能性需求。

4.1 实验数据

实验需要4个部分的数据，即QoS领域本体、领域专家指定的基本QoS需求、用户个性化QoS需求及服务提供商提供的候选服务QoS数据。本文的应用领域为酒店预定服务，采用文献[12]的案例中领域专家指定的QoS需求。本文采用的实验数据如表1所列。

表1 实验测试用例数据表

	Price/cost	Room Type	Security	Reputation	Response Time(s)	Exception Handing	Availability	
							MTTF(m)	MTTR(m)
RU	≤$5.0	Lu,St			[2.0,4.0]		KL≥0.91	
RD			≥Mid	≥2.5	[1.0,4.5]	True	≥0.90	
A_1	200C	Si,St,Lu	≥Hi	2.8	[3.0,4.0]	True	[120,130]	[5,10]
A_2	$3.3	Lu,St	≥Hi	3.3	[1.5,3.5]	True	[200,230]	[10,15]
A_3	250C	Lu,St	≥VH	4.0	[1.0,2.0]	True	≥0.95	
A_4	$5.0	Si,St,Lu	≥Mid	3.0	3.0	True	≥0.99	
A_5	$7.0	Lu,St	≥Mid	5.0	3.5	False	[220,250]	[10,20]
A_6	150C	St,Si	≥VH	3.0	[2.5,4.5]	True	[150,190]	[8,10]
A_7	350C	Lu,St	≥L	2.8	2.8	True	≥0.85	
W_1	0.7				0.2		0.1	
W_2	0.3				0.6		0.1	
W_3	0.4				0.4		0.2	
W_4	0.2				0.4		0.4	

在表1中，RU表示用户的QoS需求，RD表示领域专家指定的基本QoS需求。A_1－A_7表示候选服务的QoS参数取值情况。与领域相关的QoS属性Room Type，其度量类型为枚举型，取值范围为{Lu,St,Si}，表示房间的级别为豪华型、标准型以及单人间。QoS参数还有Price/cost(服务收费)，取值趋势为“越低越好”；Security(服务安全性)，取值趋势为“越高越好”；Reputation(服务提供商的信誉)，取值趋势为“越高越好”；Response Time(服务的响应时间)，取值趋势为“期望区间”；Exception Handing(服务是否可用)，无取值趋势；Availability(服务的可获取性)，取值趋势为“越高越好”。对应的QoS属性度量类型分别为数值型、有序集合型、数值型、数值区间型、布尔型、数值区间型。

为了验证其有效性，实验数据中价格用两种属性名称表示，分别是Price和cost，值的单位用美元和美分来表示。Availability由MTTF(平均无故障时间，取值趋势为“越高越好”)和MTTR(平均修复时间，取值趋势为“越低越好”)组合而成。这些概念在语义上都是可比的。

W_1，W_2，W_3和W_4是不同的用户对服务的个性化需求，此权重可用于服务的最终选择。

4.2 实验结果

验证本文提出的技术方案的有效性，以保证Web服务基本可执行，能处理多种异构的QoS数据，能够按照用户的个性化需求找到最为符合用户需求的候选服务。

以表1中的实验数据为输入条件，结合使用QoS领域本体库，利用开发的原型系统寻找满足用户要求的Web服务。按照本文提出的技术方案，实验结果分析如下。

首先，原型系统根据领域专家指定的基本QoS需求对候选服务集进行QoS匹配处理。可以发现候选服务A_5的匹配结果为“完全不满足”，因为该服务的Exception Handing不能满足领域专家指定的基本QoS需求；候选服务A_7的匹配结果为“插入满足”。其它候选服务的匹配结果都是“完全满足”。因此，候选服务A_5，A_7应该从候选服务集中删除，处理

后的候选服务集作为新的候选服务集，进行下一步处理。

针对用户的 QoS 需求，对候选服务集进行 QoS 匹配处理。候选服务 A_6 的匹配结果为“交叉满足”，其房间类型和服务响应时间的值与用户 QoS 需求值有一定的交集。而其他服务匹配结果都应该是“完全满足”的。经过以上两个步骤的处理，候选服务集中还剩下 A_1，A_2，A_3，A_4。原型系统对该候选服务集进行 QoS 优化选择处理，以获取最终服务。

对候选服务集进行预处理，将区间型的数值用一个代表值来表示，并统一 QoS 参数的度量单位；按照复合函数计算复合参数的值；然后以用户 QoS 需求为标准，构成矩阵 M_A。在测试数据中，由于房间类型不具备取值趋势，不需分配权值，因此预处理时只需保留价格、响应时间以及可获取性。价格的度量单位根据用户需求统一化为美元。响应时间因取值趋势为“期望区间”，故其代表值取值区间的中间值。可获取性的参数值则根据公式 $Availability = MTTF/(MTTF + MTTR)$ 求取。经过预处理后形成的矩阵 M_A 为：

$$M_A = \begin{pmatrix} 2.0 & 3.5 & 0.92 \\ 3.3 & 2.5 & 0.93 \\ 2.5 & 1.5 & 0.95 \\ 5.0 & 3.0 & 0.99 \end{pmatrix}$$

对矩阵 M_A 进行规约化处理，利用式(1)处理矩阵的第一列(价格)，其中 $q_{max}=5.0$，$q_{min}=2.0$；利用式(3)处理矩阵的第二列(反应时间)，其中 $\alpha=1.5$，$\beta=0$；利用式(2)处理矩阵的第三列(可获取性)，其中 $q_{max}=0.99$，$q_{min}=0.92$。规约化处理后所得矩阵为：

$$M_A' = \begin{pmatrix} 1 & 0.67 & 0 \\ 0.57 & 0.67 & 0.14 \\ 0.83 & 0 & 0.43 \\ 0 & 1 & 1 \end{pmatrix}$$

最后对矩阵进行加权处理。根据用户 QoS 需求可知，权值向量 $W_1=\{0.7,0.2,0.1\}$，$W_2=\{0.3,0.6,0.1\}$，$W_3=\{0.4,0.4,0.2\}$，$W_4=\{0.2,0.4,0.4\}$。根据式(4)，得加权处理结果，如表 2 所列。

表 2　QoS 优化选择的理论计算结果

服务	权值为 W_1	权值为 W_2	权值为 W_3	权值为 W_4
A_1	0.834	0.702	0.668	0.468
A_2	0.547	0.587	0.524	0.438
A_3	0.624	0.292	0.418	0.338
A_4	0.3	0.7	0.6	0.8

从表 2 可以看出，权值为 W_1 时，主要考虑价格因素，服务的排序应该为(A_1，A_3，A_2，A_4)；权值为 W_2 时，主要考虑响应时间，服务的排序应该为(A_1，A_4，A_2，A_3)；权值为 W_3 时，主要同时考虑价格和响应时间，服务的排序应该为(A_1，A_4，A_2，A_3)；权值为 W_4 时，同时考虑响应时间和可获取性，服务的排序应该为(A_4，A_1，A_2，A_3)。

结束语　本文在研究总结当前 QoS 感知的语义 Web 服务发现相关技术和方法的基础上，针对现有方法中存在的不足，利用 5 种不同度量方式来区分 QoS 参数；按领域专家指定的基本 QoS 需求对候选服务进行匹配处理，以确保服务发现的结果是基本可执行的；用约束规划方法进行 QoS 值匹配处理；通过 QoS 优化选择处理获取最终候选服务。

参 考 文 献

[1] Papazoglou M P. Web services and business transactions[J]. World Wide Web, 2003, 6(1): 49-91

[2] Cardoso J, Shet A P. Introduction to semantic web services and web process composition [C] // Proc. of First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC). 2004, 3387: 1-13

[3] O'Sullivan J, Edmond D, Hofstede A T. Formal description of non-functional service properties [EB/OL]. http://www.wsmo.org/papers/OSullivanTR2005, 2005

[4] McIlraith S, Martin D. Bringing Semantics to Web Services[J]. IEEE Intelligent systems, 2003, 18(1): 90-93

[5] Vu L, Porto F, Aberer K, et al. An Extensible and Personalized Approach to QoS-enabled Service Discovery[C]// The 11th International Database Engineering and Applications Symposium (IDEAS). 2007: 37-45

[6] Wang X, Vitvar T, Kerrigan M, et al. A qos-aware selection model for semantic web services[C] // The 4th Intl Conference on Service-Oriented Computing (ICSOC). 2006: 390-401

[7] Comerio M, Paoli F D, Maurino A, et al. NFP-aware Semantic Web Services Selection[C]// 11th IEEE International Enterprise Distributed Object Computing Conference. 2007: 484-486

[8] Benbernou S, Hacid M S. Resolution and Constraint Propagation for Semantic Web Services Discovery[J]. Distributed and Parallel Databases, 2005, 18(1): 65-81

[9] Garc'la J M, Ruiz D, et al. A Hybrid, QoS-Aware Discovery of Semantic Web Services Using Constraint Programming[C] // Proc. of the 4th Intl Conference on Service-Oriented Computing (ICSOC). 2007: 69-80

[10] 李春梅，蒋运承. 具有 QoS 约束的语义 Web 服务发现的研究[J]. 计算机科学，2007，34(6)：116-121

[11] 张佩云，黄波，孙亚民. 基于语义匹配和 QoS 的 Web 服务混合选择方法[J]. 武汉大学学报：信息科学版，2008，33(5)：140-145

[12] 刘斌. 基于 QoS 本体的语义 Web 服务选择研究[D]. 北京：北京邮电大学计算机科学与技术学院，2008

[13] Ma Q, Wang H, et al. A Semantic QoS-Aware Discovery Framework for Web Services[C] // The IEEE International Conference on Web Services. 2008: 129-136

[14] Schulte C, Smolka G. Finite Domain Constraint Programming in Oz: A Tutorial [EB/OL]. http://www.mozart-oz.org/documentation/fdt/, 2006

[15] OWL-S Service Retrieval Test Collection [DB/OL]. http://owlseditor.semwebcentral.org/projects/owls-tc, 2008

(上接第 119 页)

[14] Podolsky M, Yano K, Mccanne S. A RTCP-based retransmission protocol for unicast RTP streaming multimedia[S]. Internet draft, Internet Engineering Task Force. Oct. 1999

[15] Johanson M. Adaptive Forward Error Correction for Real-time Internet Video[C] // Proceedings of the 13th Packet Video Workshop. Nantes, France, 2003

[16] Blahut R E. Theory and practice of error-control codes [D]. AReading, MA: ddison-Wesley, 1983

[17] Liu H, Ma H, Elzarki M, et al. Error control schemes for networks: an overview[J]. Mobile Networks and Applications, 1997, 2(2): 167-182