分类号：TP301　　单位代码：10427
密　级：公开　　学　号：2011010157

济南大学

硕士学位论文

面向用户需求的Web服务发现与选择

研究生姓名　崔纪鹏
导师姓名　马炳先

学科（领域）　计算机科学与技术
申请学位类别　工学硕士
答辩时间　2014年5月23日

Web Service Discovery and Selection Based on Users' Requirement

By

CUI Ji Peng

Under the Supervision of

MA Bing Xian

A Thesis Submitted to the University of Jinan

In Partial Fulfillment of the Requirements

For the Degree of Master of Engineering Science

University of Jinan

Jinan, Shandong, P. R. China

May 23, 2014

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律责任由本人承担。

论文作者签名：崔纷鹏　　日　期：2014.5.23

关于学位论文使用授权的声明

本人完全了解济南大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借鉴；本人授权济南大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

☑公开　　□保密（____年，解密后应遵守此规定）

论文作者签名：崔纷鹏　导师签名：[illegible]　日期：2014.5.23

目 录

摘 要

Web 服务作为面向服务体系架构(SOA)和面向服务计算(SOC)的实现，以其自描述、跨平台、松耦合等特性，已成为企业业务流程集成的一个重要机制。随着 Web 服务技术的发展，越来越多的企业将自己的业务流程打包成 Web 服务，这些 Web 服务在功能、质量等方面不尽相同。Web 服务组合技术的出现解决了细粒度的单一 Web 服务难以满足复杂业务逻辑的矛盾，最大限度地提高了现有 Web 服务的可重用性和使用灵活性。

Web 服务组合技术的关键环节是如何根据用户需求，找到能够实现其业务逻辑的候选服务集，并从中选择用以组合的最优 Web 服务。现有的关于服务发现的主要工作，大多都假定已给定完成 Web 服务组合所需的各类功能相似、服务质量不同 Web 服务集，基于全局或局部 QoS 最优的方法进行服务发现与服务选择，但对候选服务集的构造以及用户 QoS 偏好方面研究有所欠缺。此外，用户有时候真正关心的是 Web 服务的一些具体内容细节信息，服务发现的过程中应该体现这些细节信息；语义 Web 技术，比如语义 Web 服务描述语言 OWL-S，可以描述这类细节信息，但语义 Web 服务本身也有很大局限。

针对现有工作的局限性，本文从三个方面考虑基于用户需求的 Web 服务发现与选择问题：

(1) 基于功能层面的服务发现。解决用户的业务逻辑需求可以由哪些抽象服务类实现的问题，最终目的是得到针对用户需求的候选服务集；候选服务集的建立为后续服务选择的进行提供基础。

(2) 基于服务质量 QoS 的服务选择。建立组合 QoS 权重表达模型，根据用户对 QoS 属性的偏好和专家打分得到不同属性的权重；通过构造遗传算法，从全局的角度考虑 Web 服务组合的用户 QoS 满意度，并以此对候选服务集进行选择。

(3) 基于服务内容的服务选择。提出并建立 Web 服务内容表达模型，通过计算服务赋值向量和用户的需求向量之间相似度的方法，确定某一 Web 服务是否能够满足用户对服务内容方面的要求，并以此为依据对候选服务集进行过滤。

基于功能层面的服务发现形成候选服务集。在候选服务集的基础上，综合应用基于 QoS 和基于服务内容的服务选择，最终可得到功能层面、服务质量层面和服务内容层面最大程度符合用户需求的 Web 服务组合。这在总体上实现了一个 Web 服务发现与选择

的层次模型，本文在最后讨论了该模型的利弊，并提出了另外一种可能的模型——解析模型的思路和需要解决的问题。

关键词：Web 服务组合；服务发现；服务功能；服务质量；服务内容

Abstract

Web service, an implement of SOA (Service-Oriented Architecture) and SOC (Service-Oriented Computing), has become an important method for the integration of enterprise work flow, due to its self-description, platform-independence and loosely-coupled character. With the development of Web service technology, increasing number of enterprises choose to encapsulate their work flow as Web services, which differ in functions, quality of service, etc. The advent of Web service composition has addressed the problem that single fine-grained Web service fails to deal with complicated work flows, and meanwhile brings about both reusability and flexibility.

Given user' requirements, discovering candidate Web service set and, among which, choosing the most suitable services for composition, is the main process in Web service composition. Existing work about Web service discovery and selection mainly takes into consideration the global QoS (Quality of Service) optimum or local QoS optimum, under the assumption that candidate Web service set for composition is just available. But the establishment of candidate Web service set is seldom referred, and moreover, users' preference for QoS properties is not reflected. Sometimes, what users really concern is the content of Web services, which should be considered in Web service discovery. This kind of information can be described with semantic web technology, such as OWL-S (Web ontology language for services), but semantic Web service itself has got much limitation.

To combat the limitation of existing work, this paper focuses on three aspects to address Web service discovery and selection:

(1) Web service discovery based on service function. The aim is to determine exactly what kinds of Web services are necessary for candidate Web service set, which acts as a foundation for the subsequent course, say, Web service selection.

(2) Web service selection based on QoS. A composite QoS model is established on the basis of users' preference and related experts' experience. What follows is the construction of GA (Genetic Algorithm), which takes a global QoS optimum aspect to deal with Web service selection.

(3) Web service selection based on content of service. A model for content of services is

proposed and established and, with this model, content of services and users' requirement for it can both be expressed in value vectors. Web service selection can be carried out based on the similarity of the two value vectors.

Web service discovery based on service function helps to establish the candidate Web service set, with which Web service selection based on QoS and Web service selection based on content of services are synthetically used. Eventually, the most suitable Web service selection can be obtained, with its functions, QoS and content best meeting users' requirements. Viewed as a whole, we establish a hierarchical model to address the problem of Web service discovery and selection. In the end, a discussion on both the advatanges and disadvantages of this model is carried out; furthermore, a possible candidate for this model, say, the analytical model, is discussed, both about its solutions and problems to be solved.

Key Words: Web service composition; Web service discovery; service function; QoS; content of services

第一章 绪 论

1.1 选题背景与研究意义

伴随着网络技术的迅猛发展，传统的分布式计算技术，如 DCOM、CORBA 等，已经不能很好地适应现有的 Web 应用环境。面向服务的体系架构(SOA)是实现分布、异构信息系统开发集成的强有力工具，以 SOA 为基础的面向服务计算(SOC)是一种支持分布式应用的新型计算规范，Web 服务作为 SOC 的一种实现应运而生。Web 服务和面向服务计算是“软件即服务”理念的一种体现，极大丰富了现有信息技术，有助于有效整合服务资源并加以最大化利用。Web 服务以其平台无关性、松耦合性以及可编程访问性等优点而得到迅速推广和应用，现已有越来越多的企业将其业务流程包装成符合标准的、具有良定义接口的 Web service，并在互联网上对外提供服务。这意味着电子商务应用正越来越倾向于通过一系列的 Web 服务来实现，这些 Web 服务可以交互处理彼此间的请求。

随着 Web 服务技术及应用的不断发展，互联网上出现了大量服务功能、服务质量以及服务粒度等不尽相同的 Web 服务，这些 Web 服务可能来自不同的供应商，适用于不同的应用领域的潜在用户。细粒度的原子 Web 服务往往只能提供某一方面的有限功能，为了满足复杂的用户业务逻辑需求，往往需要将多个细粒度的原子 Web 服务组合起来，这就是 Web 服务组合技术。Web 服务的真正潜力在于服务组合，它既是资源整合的一种方式，同时也是资源重用的重要手段；Web 服务组合使 Web 服务具有相当大的灵活性，在一定程度上保证了原子 Web 服务的细粒度，降低了开发流程的复杂度。

互联网上极大丰富的 Web 服务在给用户实现其业务逻辑需求带来最大选择性的同时，也增加了用户使用这些服务的难度。如何根据用户对业务逻辑的需求找到能够满足其需求的候选 Web 服务集，并从中选出最优服务，以 Web 服务组合的形式为用户提供最终服务，不仅是服务计算的要求，同时也决定了一次服务过程中的用户体验。根据用户业务逻辑需求的服务发现、基于用户对服务质量偏好的服务选择以及基于对服务内容个性化需求的服务选择，是决定服务成败和用户对服务满意度的关键因素。本文的着力点就是基于用户需求的 Web 服务发现与选择。

1.2 相关工作

Web 服务发现从总体上来开可以分为基于功能层面的服务发现和基于非功能层面的服务发现两个方面。基于功能层面的服务发现是基础，因为只有在服务能够满足用户的功能需求的前提下，后续的工作才有意义。非同能层面的服务发现的对象是候选 Web 服务集，即能够满足用户功能需求的潜在 Web 服务集。在非功能服务发现层面，主要考虑的因素是服务质量 QoS(quality of services)。

基于功能层面的服务发现是根据用户对服务的功能需求，从服务注册库中找到一系列的服务，构建候选服务集的过程，主要解决功能可满足性问题。基于功能层面服务发现的一个方法是利用 UDDI(Universal Description Discovery and Integration)，这是一种基于关键字匹配的服务发现方式，通过 WSDL(Web service description language)、UDDI 中服务名称、服务输入、输出参数等关键字匹配进行服务发现。该方法最大的优点是，它基于通用标准。但是其缺点也是同样明显的，因为服务的功能不可能通过少量关键字描述的非常清晰，而且不同的服务发布者可能会对一些概念的理解和表述不同，这往往会导致服务发现时的查准率和查全率较低。

针对基于 UDDI 服务发现所存在的不足，研究者们尝试从多个角度对 UDDI 策略进行改进。文献[2]从 UDDI 实现的层面入手，提出了针对服务器端的改进方法；文献[3]则是从服务消费者的角度考虑，提出一种面向消费者的服务发现机制；文献[4]从服务发现的模型匹配入手，提出了一种基于组合网的服务发现机制。这些改进方案虽然在某些方面提高了查准率和查全率，但都将 Web 服务发现局限于某种特殊的限制条件下，通用性不强。

另外一个研究方向是对 Web 服务增加语义信息，即语义 Web 服务发现。Web 服务缺乏显式语义描述，这可能造成同一个概念在服务过程不同角色之间的表达不一致，从而影响服务发现的质量和效率。语义 Web 服务通过本体(ontology)对服务进行语义标记，本体是某一领域概念和关系的标准。

文献[5]中 Paolucci M 提出的基于 Web 服务语义匹配思想，成为语义 Web 服务发现的核心思想；文献[6]以过程本体论为依据实现 Web 服务发现；文献[7]通过使用 OWL-S 本体对服务添加语义信息，并在此基础上通过予以匹配实现 Web 服务发现；文献[8]设计描述目标 Web 服务的本体，从服务功能、服务性能以及语义三个层面描述 Web 服务，实现了基于 Web 服务特征元素的服务发现模型；文献[9]构建了基于本体 ontology 的服

务搜索引擎，通过服务参数匹配以及输入参数飞机匹配的方式，实现基于概念推理的Web服务自动定位。文献[10]-[28]都是基于语义Web服务发现的一些尝试。必须看到，语义Web服务技术在理论上能够提高服务发现的质量，但是语义Web本身存在着应用的局限性。目前尚没有完善的领域本体，不同组织所基于的领域本体也存在差异，这在客观上限制了语义Web服务的应用；另外，并非所有的Web服务都是语义Web服务。

在非功能层面，主要研究工作是基于QoS的服务发现，解决的问题是如何从一系列功能特性相似而非功能特性不同的服务中，选择出最符合全局或局部QoS约束的Web服务。多采用启发式算法设计选择方法，比如遗传算法(GA)、粒子群算法(PSO)以及一些融合算法。

遗传算法的求解过程表示为染色体的优胜劣汰，通过迭代进化，利用选择、交叉和变异等手段产生下一代个体，逐步向优化种群进化。文献[29]针对在Web服务聚合过程中，将服务动态选择条件下的全局QoS最优问题，转换成带有QoS约束的多目标组合优化问题，并利用多目标GA优化原理，同时优化多个目标函数，最后得到能够满足约束条件的优化服务聚合流程集；文献[30]提出一种基于遗传算法的多QoS约束快速选择算法，该算法运行速度快，适用于实时性要求高的场景，且扩展性良好；文献[31]实现了一种面向Pareto最优遗传算法的Web服务组合方法，采用伪二叉树构造满足约束要求的Pareto最优解集，对最优解排序结合个体相似度的计算得到遗传算法的适应度函数，最终产生一组满足约束条件的Pareto最优解服务集合；文献[32]实用带精英策略的快速非支配排序的遗传算法NSGA设计Web服务选择方法，可找出一个Pareto优化解集。

采用PSO算法解决Web服务选择问题，关键是建立粒子位置矢量与方案的映射关系，确定粒子位置表示，速度算式，以及适应度函数的评估问题。文献[33]基于多目标粒子群优化算法设计了求解QoS全局最优选择算法，通过同时优化组合路径的多个QoS参数得到一组满足约束条件的Pareto最优解，用户可以依需要选择满意解，而未被选择使用的非劣解可以作为故障时的备用Web服务；文献[34]基于PSO多目标优化策略，解决了全局QoS最优化问题；文献[35]重定义粒子位置、速度等，提出一种基于PSO的QoS动态服务组合算法，该算法能够快速得到一组满足约束条件的Pareto最优服务集。

此外，文献[36]提出一种可扩展的QoS计算方法，该算法基于启发式思想，将Web服务选择问题划分为多个子优化问题，能更快速找到近似最优解；文献[37]利用马尔科夫决策过程设计出一种QoS感知的可靠Web服务组合算法；文献[38]提出了基于集对

分析的 Web 服务选择方法；文献[39]从可选服务间 QoS 的以来关系出发，提出了支持关联的组合服务选择算法；文献[40]基于 skyline 方法，通过减少候选服务数量提高服务选择的效率。基于 QoS 的服务选择研究，多数工作都是单纯从算法的层面入手，很多时候未考虑真实的可用性和目的意义，方法在某种程度上会偏离用户的实际需求，带有一定的盲目性。

1.3 研究内容与创新点

本文主要研究内容是基于用户需求的 Web 服务发现与选择。通过调研用户可能真正关心的 Web 服务内容，以现有的 Web 服务描述为基础，充分挖掘相关的服务描述信息，建立相应的信息表达模型，选用合适的算法，分层次实现目标 Web 服务的发现与选择。研究内容和思路如下：

(1) 基于功能层面的服务发现。通过构造领域服务功能树功能分解，理解和分析用户的功能需求，从系统的注册服务库中，找到能够实现该用户功能需求的一组服务集合，称之为候选 Web 服务集。候选 Web 服务集将功能相似的服务聚合成服务类，为用户提供统一服务资源视图，为进一步的服务发现与选择提供基础。候选 Web 服务集主要描述两个方面的信息：第一，用户的功能需求应该由哪类 Web 服务或由哪几类 Web 服务组合来实现；第二，这些 Web 服务类分别对应于哪些可用的具体 Web 服务。

(2) 基于 QoS 约束的服务选择。建立组合 QoS 权重表达模型，该模型包括反映领域客观标准的客观权重和用户偏好的主观权重。定义用户 QoS 满意度， QoS 指标的重要程度，以及实际服务中该属性与用户期望值的差距，共同决定了用户的 QoS 满意度；从用户 QoS 满意度的角度出发，构造智能优化算法对候选服务集进行选择。

(3) 基于服务内容的服务选择。目前的 Web 服务技术中尚没有特别实用的 Web 服务内容描述模型。为了方便表达，定义依赖于 Web 服务类的 CoS(Content of Service)模型，分析了 CoS 的性质以及 CoS 初始化和更新方法。在此基础上，定义用户内容需求向量和 Web 服务赋值向量相似度的概念，通过比较相似度和预定义阈值的大小关系，对候选服务集进行再过滤。

基于功能层面的服务发现构造候选服务集是其余工作的基础，基于 QoS 约束的服务选择和基于服务内容的服务选择可以单独应用，也可以综合使用。本文创新点主要有：第一，分层次实现基于用户需求的 Web 服务发现与选择，各层次之间有很大的独立性和灵活性，同时又能够综合使用达到更优效果；第二，提出了描述 Web 服务具体内容

的 CoS 模型，分析了 CoS 的性质，给出了基于服务 CoS 向量和用户内容需求向量相似度的服务过滤方法；第三，过程的用户参与度高，服务选择的过程能够最大限度地体现用户的意愿，进而提高服务的用户满意度。

1.4 论文组织结构

全文共有六章，内容分别如下：

第一章介绍了本文的选题背景和研究意义，分析了现有的关于 Web 服务发现的研究动态。针对现有工作的局限性，以用户需求为出发点，提出了分层的 Web 服务发现与选择方法，并给出总体的研究思路。

第二章主要介绍 Web 服务和 Web 服务发现的一些相关概念。包括 Web 服务的概念、体系架构，Web 服务的角色，Web 服务组合的相关概念，Web 服务发现的概念，候选 Web 服务集等。

第三章是基于功能层面的服务发现。主要介绍领域服务功能树的概念以及基于领域服务功能树和功能分解原理的候选服务集构造。

第四章涉及到基于 QoS 的服务选择。主要包括 QoS 概述，组合 QoS 权重模型的建立，用户满意度的定义，以及智能优化算法构造；此外，本章还给出所提方法的实验过程及与其他工作的结果比较。

第五章提出了描述服务内容的 CoS 模型。引入 CoS 的概念，分析了 CoS 的性质，介绍 CoS 的初始化和更新算法；在此基础上，给出 Web 服务 CoS 的度量方法和基于 CoS 的服务选择方法；本章最后讨论了本文实现的基于层次模型的组合选择模型的利弊，并提出解析模型实现的可能性和难点所在。

第六章是总结与展望，全面总结了本文已完成工作，并指出下一步工作的着眼点与思路。

第二章 Web服务与Web服务发现

本章首先介绍面向服务的体系架构SOA (Service-Oriented Architecture)，包括SOA的特性、角色及行为；在此基础上介绍作为SOA目前最为有效实现技术的Web服务，分析了Web服务的技术构架。紧接着介绍了服务库，服务库是本文工作的基础，Web服务的发现、组合、规划执行都以服务库为基础。最后结合作者所在课题组的整体项目框架，明确了本文工作在整个框架中的地位和意义。

2.1 面向服务体系架构SOA

SOA可理解为一种软件设计的逻辑方法，能够通过发布或发现的接口向潜在用户提供服务，其主要目的就是使得已有的技术具有互操作性，并使得未来的应用和体系结构具备可扩展性。SOA能够利用基于一定标准的功能性服务，将孤立静态系统转化成模块化的灵活组件。同时，SOA还是一种设计理念，它独立于具体技术，可以有多重实现方式，而Web service是其中的一种实现。

2.1.1 SOA的特性

SOA的一个重要作用是提高已有服务的可重用性，既可以独立使用这些已有服务，也可以将其作为一个应用组成部分，实现更强的功能。SOA中服务具有以下特性[41]：

(1) SOA中的一切功能都被定义成服务。这些功能包括纯业务功能、底层功能构成的业务事务以及系统服务功能。SOA中使用的服务可以是全新的服务，也可以是原有服务的包装或组合。

(2) SOA中所有的服务相互独立存在。服务操作过程对外部程序而言是不可见的，外部组件只关心服务是否满足所需要的功能和QoS，而不必了解具体的实现过程。外部程序所需要的一些有用信息，比如服务所在的位置、功能以及调用方法等，都隐藏在服务接口中。

(3) SOA中所有服务可以通过接口被调用。这意味着在SOA中，无需区分本地服务还是远程服务，也不用区别服务采用的互联模式、协议和连接所需的组件等。

(4) 服务接口和服务实现分离。这是SOA松耦合原则的体现。

SOA 的这些特性是其重用现有服务，提高现有技术互操作性、增加使用灵活性的关键所在，也是 SOA 的具体技术实现，比如 Web 服务所必须遵循的基本原则。

2.1.2 SOA 中进行交互的角色和操作

SOA 为面向服务计算 SOC(Service-Oriented Computing)提供基本支持和运行环境，SOA 有三个基本角色，他们分别是：服务提供者、服务请求者和服务注册中心，如图 2.1[42]所示。服务的请求者可以是应用程序、软件模块，也可以是一个服务，它查询服务注册中心，绑定并服务并执行；服务提供者为可网络寻址的实体，发布服务描述到注册中心，并能够接收请求者的服务请求，便于请求者发现和使用某一服务；服务注册中心是有效服务的目录，扮演服务发布者和服务请求者中介的作用，前者在这里发布服务，后者在这里查找感兴趣的服务。

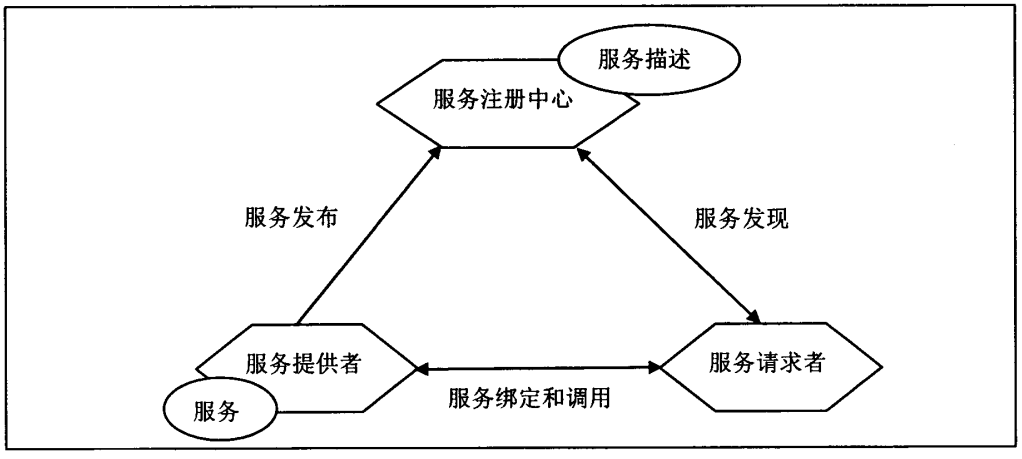

图 2.1 SOA 角色关系

SOA 的三种角色对应三种不同的操作和行为，它们分别是：

(1) 服务发布：由服务提供者完成。服务提供者将所能提供的服务发布到注册中心，服务请求者基于此发现和调用相关服务。

(2) 服务发现：在服务注册中心完成。服务请求者通过注册中心查询所需服务的相关描述信息。

(3) 服务绑定和服务调用：由服务发布者和服务请求者共同完成。服务请求者就某一服务和服务提供者建立连接，发送请求并接收服务结果。

分析 SOA 中三种角色及相关的操作行为可知，如果从服务管理的层面来看，服务的整个执行过程包括三个阶段：

(1) 服务描述：服务在注册中心通过某种标准化形式加以描述，所描述的信息应包含其功能及非功能属性。

(2) 服务发现和匹配：通过匹配服务描述信息和用户需求来实现；服务发现应该同时满足用户对服务功能及非功能的需求。

(3) 服务的实现：即服务的绑定和调用。

2.2 Web 服务及技术构架

SOA 不涉及具体的技术实现。事实上，已有的中间件技术，比如 J2EE、CORBA、JMS 等都可以实现 SOA。作为现今首选的 SOA 实现方式，Web 服务与这些中间件框架不同，它真正能够以语言独立、跨平台和松耦合的方式进行服务集成，最大限度实现了 SOA 的服务共享、服务复用和互操作。Web 服务降低了应用复杂性，通过良定义接口，屏蔽了诸多实现细节；甚至可以通过 Web 服务实现遗留系统的集成和互操作。

2.2.1 什么是 Web 服务

Web 服务作为 SOA 的实现，关于它的具体概念从诞生之日起就有很多种描述，在文献[43]-[46]中， IBM 公司、微软、SUN 公司和国际标准化组织 W3C(World Wide Web Consortium)分别对 Web 服务给出了自己的定义。综合业界观点，Web 服务是一个平台独立、松耦合、自包含的 Web 应用程序，基于可编程访问，可用开放的 XML(Extensible Markup Language)标准描述、发布、发现、协调和配置；是一种用于开发分布式、互操作的应用程序。

2.2.2 Web 服务技术构架

应用程序间可以通过 Web 服务技术，基于互联网协议进行协作，而无需人为干预。事实上，Web 服务代表了几类相关的技术，它的底层实现基于开放的 Internet 标准。

Web 服务的技术架构如图 2.2 所示[47]。最底层的是 Web 服务的使能技术标准，是 Web 服务实现其操作的保障。Web 服务基于 Internet 实现连接并构建基础架构，因此在

传输层选择使用可靠的 HTTP 协议；Web 服务使用的另一个开放标准是 XML，数据及相应语义的交换都是基于 XML 技术来实现的。

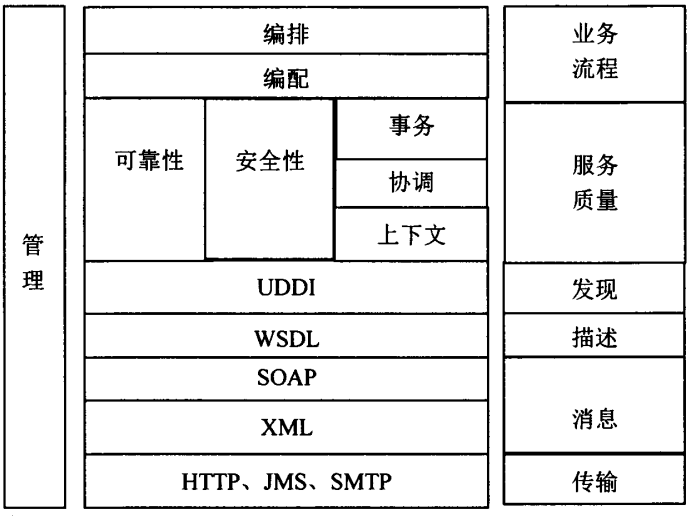

图 2.2 Web 服务技术架构

Web 服务的核心标准包括简单对象访问协议 SOAP(Simple Object Access Protocol)、Web 服务描述语言 WSDL(Web Services Description Language)和 UDDI 规范(Universal Description, Discovery and Integration)，这三大标准的建立为 Web 服务发展奠定基础[48]，分别介绍如下：

(1) 简单对象访问协议 SOAP：SOAP 是一个消息协议，它基于开放的 XML 标准，是 Web 服务的信息交换协议，包括 SOAP 信封、消息编码规则、SOAP 远程过程调用表示和 SOAP 绑定四个部分；SOAP 在传输层使用 HTTP 协议保障数据的可靠传输。

(2) Web 服务描述语言 WSDL：WSDL 基于 XML 标准，将 Web 服务描述为可进行消息交换的通信端点集合。WSDL 文档通过<types>元素、<message>元素、<operation>元素、<portType>元素、<binding>元素和<service>元素描述服务接口。

(3) UDDI 标准：UDDI 是一个目录，提供服务在线发布，是服务提供者和服务请求者的中间桥梁；UDDI 的使用使得 Web 服务潜在用户能够有效发现符合要求的服务并加以调用。

2.3 服务库

本文的研究基于项目组所建立的服务库[49]，它相当于一个可控的服务注册中心，是被假定包含了某一领域所有可能功能的服务集合，详细记录服务发现以及服务调用所需

要的大量服务描述信息，包括服务的 ID 号、服务名称、服务类别、服务 WSDL 文档、服务方法名、服务输入输出等。服务库是本文研究的基础，同时也是 Web 服务发现、服务组合、以及规划执行的基础。

2.4 项目框架及本文工作的地位

图 2.3 给出了作者所在课题组的项目框架：

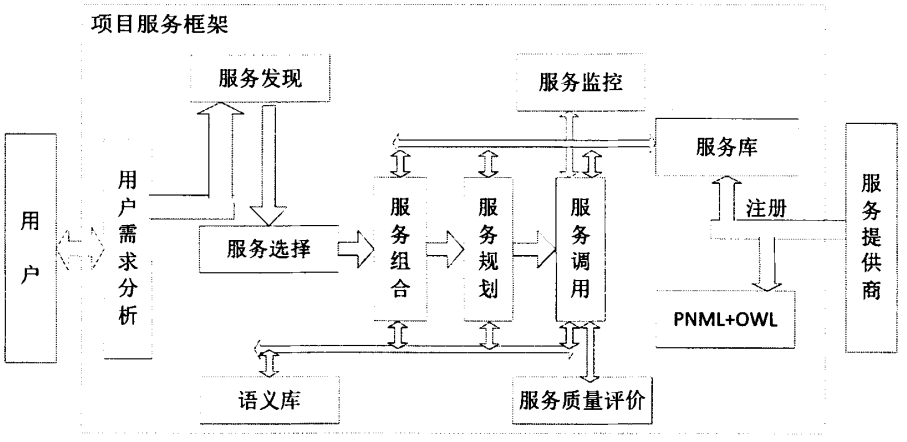

图 2.3 项目服务框架图

由图可知，项目的总体目标是搭建起服务请求者(用户)与服务提供者的业务联系，为用户表达需求、查找符合需要的 Web 服务或 Web 服务组合提供平台。整个过程从用户表达自己的需求开始，系统收到并分析用户需求，以此为基础在服务库中查找能够满足用户需求的 Web 服务，这就是本文工作中关于服务发现的研究；紧接着，根据用户的偏好和个性化需求，对发现的服务进行再选择，得到最优的 Web 服务或 Web 服务序列，这是本文工作中关于服务选择的研究；服务发现和选择得到的 Web 服务或服务序列经由服务组合、服务规划和服务调用，完成整个服务过程。

从对系统框架图的分析可以看出，基于用户需求的 Web 服务发现与选择是整个服务过程的基础，服务发现与选择的好坏直接关系到服务的成败，影响用户对该次服务的满意度和服务体验。

2.5 本章小结

本章从 SOA 的概念入手，先后介绍了 SOA 的特点和行为，在此基础上引入了作为 SOA 实现的 Web 服务，对 Web 服务的定义、技术架构进行讨论；本章还提到了服务库的概念，它可以简明的看作是一个可控的、完备的服务集，是本文研究工作的基础；最后结合本文所在课题组的研究工作，明确了本文工作所处的地位和重要意义。

第三章 基于功能层面的服务发现

在基于用户需求的 Web 服务发现与组合过程中，根据用户对服务功能需求，找到能够满足需要的 Web 服务或一组 Web 服务集合，并以此构建候选服务集是要解决的首要问题，也是决定整个服务过程成败的关键。只有在查找到 Web 服务或 Web 服务组合能在功能上满足用户需求的前提下，其他工作的进行才有意义。

本章主要介绍基于功能层面的服务发现，在服务库概念的基础上，首先介绍 Web 服务组合的概念和意义、Web 服务描述的细节模型和总体模型；基于 Web 服务的模型描述，介绍了抽象服务类和抽象服务集的概念，定义了候选服务集的形式；最后引入领域服务功能树，并介绍基于功能层面的服务发现和候选服务集的构造。

3.1 Web 服务组合

细粒度的单原子 Web 服务功能相对单一，难以满足复杂的用户业务逻辑需求；与此同时，整合 Internet 上出现的自治和异构的服务，充分应用现有的 Web 服务集成开发新的应用，既节约了开发成本，又提高了 Web 服务的可重用性，是 Web 服务发展的必然趋势。Web 服务组合[50-52]是为了完成复杂业务需求，通过协同调度，使多个 Web 服务按一定策略有机组合、交互协作，最终达到资源整合的目的；Web 服务潜力源于服务组合[21]。

本文研究所基于的项目组采用的 Web 服务组合方法，主要以 Web 服务间的语义关联为基础，通过语义标记的 Petri 网的共享合成[53]自动生成 Web 服务自合的 Petri 网模型，进而实现 Web 服务间的组合。实现 Web 服务组合的 Web 服务之间通过数据关联[54]确定其在业务逻辑中的关系，因此，本文在服务发现层面需要解决的问题，仅限于给出可供组合的 Web 服务序列，而不必关注这些 Web 服务之间的相互关联关系。

3.2 候选服务集

基于功能层面服务发现的目的就是构造候选服务集。在引入候选服务集的概念之前，首先要介绍几个相关的概念。

3.2.1 Web 服务描述模型

首先给出 Web 服务的形式化描述如下：

定义 3.1 Web 服务是一个五元组，即 $W=\{N,U,f,B,P\}$[55]，其中：

(1) N 是 Web 服务输入参数名称的集合，包括 Web 服务中所有的输入参数名称；

(2) U 是 Web 服务输出参数名称的集合，包括 Web 服务中的所有的输出参数名称；

(3) f 是 Web 服务中的过程变量名称的集合，包括 Web 服务中的其他的参数名称；

(4) B 是表示 Web 服务中的行为动作名称集合，是对 Web 服务过程进行抽象的集合，包括 Web 服务中所有的操作行为名称；

(5) P 是 Web 服务的端口地址的集合，包括要访问 Web 服务的链接地址。

该描述是 Web 服务的细节描述，包括了服务的参数、服务的行为动作和服务的端口集合。如果从更高的层面来看一个 Web 服务，它就像是一个黑盒子，可以对某一输入集合作出特定的响应，给出确定的输出集合。Web 服务在整个过程中所体现出来的是其功能属性和非功能属性；结合这个思想，文献[34]给出了定义 3.2 所示的 Web 服务描述模型。

定义 3.2 Web 服务可以描述为一个二元组，即 W = {F, Q}，其中：

(1) F 为 Web 服务功能属性的集合。服务的功能属性取决于定义 3.1 中所定义的输入参数、输出参数集合；

(2) Q 为 Web 服务非功能属性的集合，主要是服务质量 QoS。非功能属性，尤其是服务质量，在某种程度上决定了服务的成败和服务的用户体验，是在服务发现过程中必须考虑的一个重要因素。

该描述是从整体层面上考虑 Web 服务所得到的模型，它不涉及具体的服务细节，却方便问题的描述和定位。本章的研究工作是基于功能层面的服务发现，就是针对该描述模型中的 F 集合而进行的。

3.2.2 抽象服务类

Web 服务可以根据其功能进行分类；抽象服务类[36]用以描述功能相似的一组 Web 服务的集合，属于同一个抽象服务类的 Web 服务彼此间拥有相类似的功能属性。本文把包含系统所有注册 Web 服务的服务库 C 看作是论域，并在此基础上定义抽象服务类的概念。

定理 3.1 服务库 C 上的服务功能相似关系是一个等价关系，记为 R。

证明：假设 Web 服务 ws_1，ws_2，ws_3 是服务库 C 上的任意 3 个 Web 服务，则有：

(1) 显然对于任何一个 Web 服务 ws，它与自身功能完全相同，故满足自反性；

(2) 如果 Web 服务 ws_1 和 ws_2 功能相似，那么显然也有 Web 服务 ws_2 与 ws_1 功能相似，故满足对称性；

(3) 如果 ws_1 与 ws_2 功能相似，且 ws_2 与 ws_3 功能相似，那么必然有 ws_1 和 ws_3 功能相似，故满足传递性。

综上可知，服务库 C 上的 Web 服务功能相似关系是一个等价关系；而且，服务库 C 是该等价关系的自反闭包、对称闭包和传递闭包。

定义 3.3 假设服务库 C 在等价关系 R 下的商集为：$C/R = \{ [ws_1]_R, [ws_2]_R, \ldots, [ws_k]_R \}$，构造字符串集 $S = \{ S_1, S_2, \ldots, S_k \}$，在此基础上，构造如图 3.1 所示的从字符串集 S 到商集 C/R 的一一映射 f：

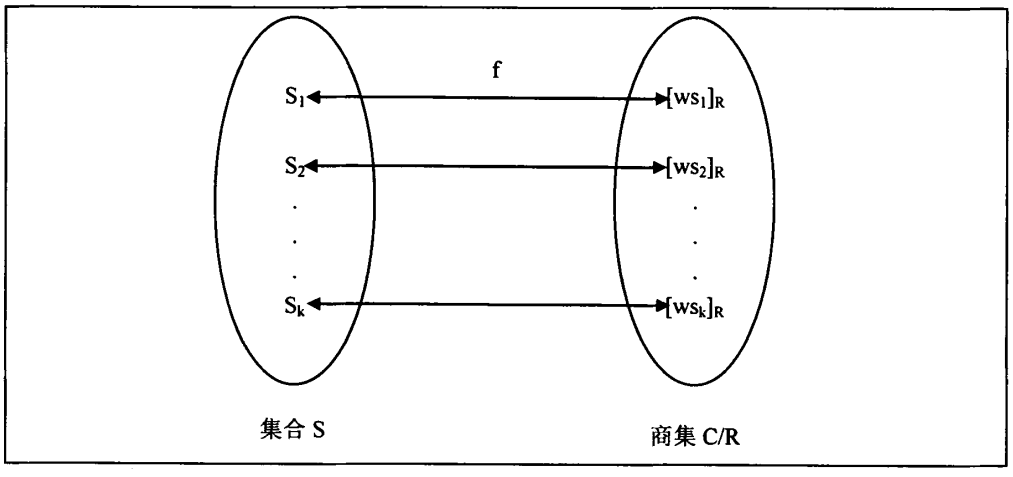

图 3.1 集合 S 到 C/R 的一一映射 f

则称 $S_1, S_2, \ldots, S_k$ 为服务库 C 的 k 个抽象服务类。抽象服务集 S 中字符串 $S_i (i=1,2,\ldots,k)$ 的取值是可以自定义的，一般来说其取值应该是可以对某类 Web 服务功能进行简要描述的字符串；但不论取值如何，一旦规定了就不可再更改。

这样我们就从集合论的角度定义了抽象服务类的概念。

定义 3.4 由一个或多个抽象服务类构成的集合称为抽象服务集。

抽象服务集的定义是建立在抽象服务类的基础之上的，每一个抽象服务集都是具体服务集在功能层面的抽象表示；其中定义 3.3 中的字符串集 S 是服务库集 C 的抽象表示，S 中元素的个数 k 表示服务库 C 中所有 Web 服务所能提供的功能数。

有了抽象服务集和具体服务集的概念，接下来定义抽象 Web 服务组合和具体 Web 服务组合：

定义 3.5 由抽象服务类构成的 Web 服务组合称为抽象 Web 服务组合；由具体 Web 服务构成的 Web 服务组合称为具体 Web 服务组合。

3.2.3 候选服务集

至此可以总结说，基于功能层面的服务发现，就是要找到满足用户业务逻辑需求的抽象服务类，并给出抽象服务类所对应的具体 Web 服务集合。为了更加清楚的描述这个问题，引入候选服务集的概念。

定义 3.6 给定服务库 C、服务库对应的抽象服务集 $S = \{ S_1, S_2,..., S_k \}$和服务库 C 在等价关系 R 下的商集 $C/R = \{ [ws_1]_R, [ws_2]_R,..., [ws_k]_R\}$，以及从抽象服务集 S 到商集 C/R 的一一映射 f，则对应于某一用户需求的候选服务集定义为一个四元组，即 $Candidate\_Set = \{ S^*, P^*, C^*, f^* \}$，其中：

(1) S^*是抽象服务集，它是服务库 C 所对应抽象服务集 S 的子集；S^*中抽象服务类的选取取决于用户的功能需求；

(2) P^*是服务库 C 在等价关系 R 下商集 C/R 的子集；

(3) C^*是服务库 C 的子集，是抽象服务集 S^*所对应的具体服务的集合；

(4) f^*是从抽象服务集 S^*到集合 P^*的一一映射，它必须包含于 f。

图 3.3 给出了候选服务集中各元的对应关系。

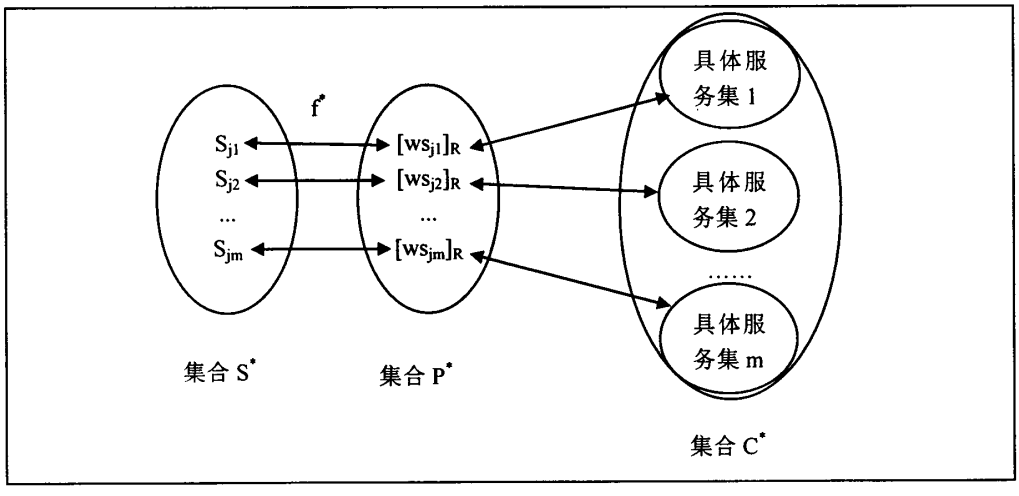

图 3.3 候选服务集各元对应关系

关于图 3.3 的一些说明：抽象服务集 S^*有 m 个元素，表示 m 个抽象服务类；这些抽象服务类在和集合 P^*中的元素存在一一对应关系，而 P^*中的每个元素都是一个关系 R 诱导的等价类，它是一组功能相似的 Web 服务。这样一来，就可以由抽象服务类找到对应的具体 Web 服务。

从候选服务集的定义及其描述不难看出，某一用户需求对应的候选服务集可以描述以下两个方面的内容：

(1) 该用户需求对应的抽象服务集，这反映了用户的业务逻辑需求可以由哪些抽象的服务类来协作完成，由候选服务集概念模型中的 S^*元素来描述；

(2) 每一个抽象服务类对应哪些可用的具体 Web 服务，由候选服务集概念模型中的 P^*元素来描述。

可见，基于用户功能需求的 Web 服务发现要达到的最终目标就是，得到对应于该需求的候选服务集。

3.3 领域服务功能树

3.3.1 功能分解原理

在某一特定领域中，一个较大功能可以通过分解为相对较小的子功能来实现；而每一个小的子功能又可能分解为更小的次子功能来实现。这个递归的过程所描述的就是领域功能的逐层分解，它都有以下特点：

(1) 一个领域功能总是可以分解为多个较小的子功能；

(2) 功能的分解可以持续进行下去，直到达到分解截止条件；

(3) 功能的分解过程是一个有限的过程，它总是可以达到分解截止条件。

其中，分解截止条件定义如下：

定义 3.7 假设 Func 是某一服务领域的一个功能，称 Func 达到分解截止条件当且仅当对于 Func 的任意一个子功能 sub_Func，都有 Func = sub_Func，其中“=”代表两个功能相同。

一个确定的服务领域所能提供的服务功能总是可列的，而一个大的功能总可以分解成几个相对较小的功能来实现，任一功能的分解都是一个有限过程，都能够达到分解截止条件。这是构造领域服务功能树的理论基础。

下面定义功能分解的层次描述。

定义 3.8 领域中某一功能 F_i，它一次功能分解得到的子功能集合称为 F_i 的 1-度功能集；功能 F_i 的 1-度功能集中所有子功能的 j-度功能集的并集是 F_i 的(j+1)-度功能集。

定义 3.8 定义了功能分解的层次描述，通过某一功能的 j-度功能集表示该功能在分解 j 次之后所对应的所有子功能集合。

定义 3.9 对于服务领域内任意一个功能 Func，对其按照功能分解原理进行分解，当其所有子功能都不能再分时，这些不可再分的子功能(假设有 t 个)构成的集合{sub_Func_1, sub_Func_2,…,sub_Func_t}称为功能 Func 的等价功能集，且称 Func 等价功能集中的每一个元素都属于功能 Func。

定义 3.9 给出了用抽象服务类的集合表达领域功能的方法。

在服务功能分解时我们还注意到一个现象，一个功能的实现有时候必须建立在其子功能全部能够得到实现的基础上，而有时候只要求某一个子或部分几个功能得到实现，我们必须区分功能与其子功能之间的这两种不同依赖关系。

定义 3.10 假设功能 F_i 的 1-度功能集为{F_{i1}, F_{i2},…,F_{in}}，则如果 F_i 能够实现当且仅当其 1-度功能集中的所有 n 个子功能都必须能够得到实现，那么称 F_i 对其子功能是强依赖的；否则称 F_i 对其子功能是弱依赖的。

一个功能对其子功能的依赖关系在很大程度上决定了可对其子功能进行取舍的程度，这个取舍的过程取决于用户的选择。

接下来将介绍领域服务功能树的构造，先从其特殊的部分开始。

3.3.2 领域服务功能树的树根——服务领域名

作为一颗树总要有其树根。在领域服务功能树中，树根仅仅是一个形式化的标识，并没有什么实际的意义。作为一个标识，通常将领域服务功能树的树根用某一特定服务领域的领域名称来表示。图 3.4 是科学计算服务领域服务功能树的树根表示：

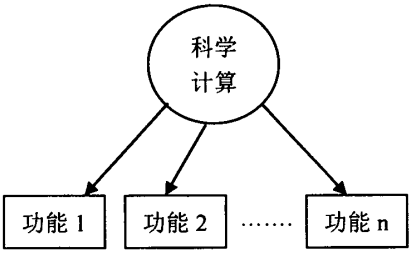

图 3.4 科学计算领域功能树的树根

树根用“科学计算”这个字符串来标识。树根下面的功能 1,功能 2,...,功能 n 是该领域最大的 n 个功能，它们在领域服务功能树中作为树根的孩子结点而存在。

3.3.3 领域服务功能树的树叶——抽象服务类

功能分解截止条件和抽象服务类的关系可由下面的定理描述：

定理 3.2 任何一个领域的功能分解截止于抽象服务类。

该定理的证明过程如下：

由定义 3.3 可知，抽象服务类是对某一类功能相似的 Web 服务的描述，所以抽象服务类也是一种服务功能，理应属于领域功能树的一部分。另外，抽象服务服务类是对单个 Web 服务功能的描述，它只能依靠具体 Web 服务调用来实现其功能；换句话说，抽象服务类所代表的功能是领域功能树中最小的功能，它不可以再细分为其他的子功能。根据定义 3.7 中对分解截止条件的描述可知，一个功能达到分解截止条件的充分必要条件是该功能等价于抽象服务类所描述的功能。

由定理 3.2 可知，抽象服务类不但是领域功能树的一个必要部分，而且在树中充当着树叶的角色。

定理 3.3 假设某个领域中所有 Web 服务对应的抽象服务集为 $S = \{S_1,S_2,\ldots, S_k\}$，该领域所有服务功能的集合是 $F = \{F_1,F_2,\ldots,F_n\}$，若将其中每一个功能写成其等价功能集的形式而得到集合 F^*，则 F^*是抽象服务集 S 的一个覆盖。

定理 3.3 证明如下：由定理 3.2 可知，功能分解截止于抽象服务类，结合定义 3.8 对等价功能集的定义可知，对于抽象服务集中的任意元素，它必然属于某一个功能的等价服务集，从而 F^*是 S 的一个覆盖。

定理 3.3 的意义在于，领域中的所有功能对应的等价功能集覆盖所有的抽象服务类；由抽象服务类和抽象服务集的定义可知，抽象服务类对应具体 Web 服务集合。这样一来，某一领域的所有注册 Web 服务都可以参与到某些功能的实现中，同时领域的所有功能最总能够找到合适的具体 Web 服务来实现。

3.3.4 构造领域服务功能树

有了前面介绍的相关理论，现在我们开始构造领域服务功能树。如果已知领域名称为 root，抽象服务集为 $S = \{S_1,S_2,\ldots, S_k\}$，我们记构造服务功能树为 Create(root)，则有：

Procedure Create(root)

Step1. 构造仅由树根 root 构成的领域服务功能树 T = { root }；

Step2. 判断领域服务功能树的叶子结点是否全部由抽象服务类构成：如果是，转向 Step4；否则，执行 Step3；

Step3. 对于非抽象服务类构成的叶子结点，定义其 1 度功能集、确定该结点与其子功能的强弱依赖关系，并把集合中的元素作为该结点的孩子结点添加到服务功能树中；转向 Step2；

Step4. 构造过程结束，返回领域服务功能树 root。

构造的领域服务功能树如图 3.5 所示。根据前面介绍服务功能分解原理、分解截止条件以及定理 3.2 可知，该构造过程总是可以在有限的步骤内结束。

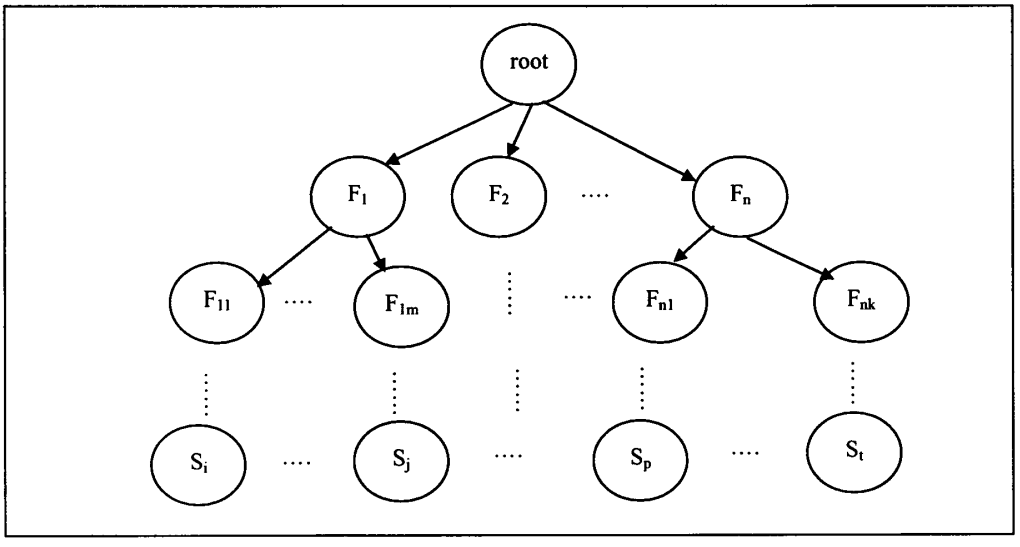

图 3.5 领域服务功能树

在所构造的领域服务功能树种，处于叶子结点的功能都是抽象服务类，可以由它们得到对应的具体 Web 服务，来实现功能树种的任意一个功能。

3.3.5 领域服务功能树的更新

已构造的领域服务功能树并非一成不变的，结合领域服务的发展，不断有新注册 Web 服务加入和现有 Web 服务的淘汰，这会导致在领域服务功能树中新增一些功能或

删除一些已有的功能，甚至会导致抽象服务类的增加或减少。本节主要讨论领域服务功能树中删除功能操作和增加功能操作的相关方法。

首先讨论从领域服务功能树中删除功能。和普通的树操作删除结点不同，领域服务功能树中的结点间具有相互依赖关系。删除一个功能结点，必然会导致以该功能为某 k-度功能集元素的其他结点功能实现受到影响；同时，被删除节点的后继结点必属于该结点的某 p-度功能集，它们理应同时被删除。我们把在领域服务功能树 root 中删除一个结点 X 的操作记为 DeleteNode(X)，则有：

Procedure DeleteNode(X)

对于领域服务功能树上一个 X 结点的出现，执行以下步骤：

Step1. 判断 X 的父结点 X^*与其子功能的依赖关系：如果是弱依赖的，转向 Step2；如果是强依赖的，执行 Step3；

Step2. 删除 X 结点及其后继结点；转向 Step4；

Step3. 删除 X 结点及其后继结点，执行 DeleteNode(X^*)，转向 Step4；

Step4. 考察领域服务功能树 root 中是否还有 X 结点的存在：如果有，转向 Step1；如果没有，转向 Step5；

Step5. X 结点删除完毕，Procedure DeleteNode(X)结束，返回新的领域服务功能树 root。

从 Procedure DeleteNode(X)的执行可以看出，删除一个结点 X 的关键在于考察 X 的兄弟结点能否继续支持其父结点功能的实现，如果可以支持，则直接删除 X 结点及后继结点；如果 X 的兄弟结点无法继续支持父结点功能的实现，则在删除 X 结点及其后继结点的同时，还要删除 X 结点的父结点 X^*，对 X^*的删除执行相同的 DeleteNode(X^*)过程，因此这是一个递归的过程。

接下来考虑在领域服务功能树中添加一个结点的操作。在讨论具体的添加过程之前，我们先定义在领域服务功能树中结点间的盖住关系。

定义 3.11 在领域服务功能树中，称功能节点 F_i 盖住功能结点 F_j 当且仅当 F_j 在 F_i 的 1-度功能集中。

结点间的盖住关系如图 3.6 所示；可见，结点 F_i 盖住 F_j 就是这两个结点间不存在其他的功能层次。

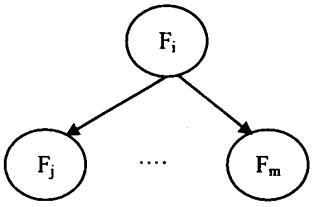

图 3.6 结点间盖住关系

相对于删除结点的操作，添加一个新的结点则没有那么多的顾虑，唯一需要考虑的就是新添加结点在领域功能树中的位置；而这个添加位置由服务功能树中已有的结点和新增结点间的盖住关系来决定。如果把在领域功能树 root 中添加一个结点 Y 的操作记为 Add(Y)，则有以下的过程实现：

Procedure Add(Y)

Step1. 调用领域功能树构造过程，构造以 Y 为根结点的功能树，即执行 Create(Y)；

Step2. 按层次考察领域服务功能树 root 中的功能结点；对于结点 Z，判断结点 Z 是否盖住结点 Y：如果 Z 盖住 Y，执行 Step3；否则执行 Step4；

Step3. 将过程 Create(Y)返回的功能树 Y 作为结点 Z 的子树添加到结点 Z 下面，结点 Y 作为结点 Z 的孩子结点；转向 Step6；

Step4. 考察对服务功能树 root 中是否还有为遍历的结点：如果有，执行 Step2；否则转向 Step5；

Step5. 将 Create(Y)返回的功能树 Y 作为根 root 的子树添加到 root 下面，结点 Y 作为根结点 root 的孩子结点；转向 Step6；

Step6. 添加结点成功，Add(Y)过程结束，返回新的服务功能树 root。

通过对 Procedure Add(Y)的分析可知，添加一个结点主要有两部分工作：第一是以新添加结点为根结点构造一颗服务功能子树；第二是在原有的领域服务功能树中找到能够盖住结点 Y 的功能结点 Z，并把 Y 作为 Z 的孩子结点添加到领域服务功能树中。如果所有的功能结点都不能盖住新增结点 Y，那么直接将 Y 结点作为根 root 的孩子结点添加到领域服务功能树中。

3.4 基于功能层面的服务发现及候选服务集的构造

至此我们已经全面介绍了抽象服务类、抽象服务集和候选服务集的概念，引入了领域服务功能树的相关理论，本节将基于这些概念和理论讨论基于功能层面的 Web 服务发现以及候选服务集的构造。

3.4.1 基于功能层面的服务发现

基于功能层面的服务发现，就是根据用户业务逻辑的功能需求，从服务库中找到能够实现该需求 Web 服务或一组 Web 服务集合。本文对基于功能层面的服务发现定义如下：

定义 3.11 假定用户业务逻辑的功能需求为 F_x，那么基于功能层面的 Web 服务发现就是在领域服务功能树中，查找与 F_x 功能等价的功能结点过程。

我们记基于功能层面 Web 服务发现过程为 Func_Based_Discovery(F_x)，则有以下过程：

Procedure Func_Based_Discovery(root, F_x)

建立一个功能队列 Func_Queue，

Step1. 遍历领域服务功能树根结点 root 的 1-度功能集 set_1，对于 set_1 中的任一功能元素 F_y，判断 F_x 与 F_y 是否是相同功能；如果是，转向 Step5；如果不是，转向 Step2；

Step2. 将 set_1 中的所有元素依次加入到功能队列 Func_Queue 末尾，转向 Step3；

Step3. 如果功能队列 Func_Queue 中元素个数不为 0，则从队列出口取出一个功能元素，假设为 F_p，执行过程 Func_Based_Discovery(F_p, F_x)；如果队列元素个数为 0，转向 Step4；

Step4. 领域服务功能树中没有与 F_x 功能符合的功能结点，服务发现失败，过程结束；

Step5. 领域服务功能树中的功能结点 F_y 与用户功能需求 F_x 相符合，返回功能结点 F_y，过程结束。

从过程 Func_Based_Discovery(root, F_x)的构造可以看出基于功能层面的 Web 服务发现过程，是一个基于领域服务功能树的广度优先遍历过程。

3.4.2 候选服务集的构造

在讨论候选服务集的构造之前，先定义针对领域服务功能树的剪枝操作，剪枝是对树中结点的后继结点进行取舍的过程，它的主要依据是功能与其子功能间的依赖关系。

定义 3.11 剪枝操作是针对领域服务功能树中具有弱依赖性的功能结点，根据用户的选择删除其部分子功能，达到缩减其 1-度功能集、同时不影响其实现用户功能需求目的而进行的操作。

可见，剪枝操作是删除掉服务功能树中和用户功能需求不直接相关的结点的操作；这些被删除的功能结点理论上也可以实现用户的功能需求，但因为不满足用户的偏好而被剔除。剪枝操作可以在最大程度上保证候选服务集的精练和有效。

由定义 3.6 可知候选服务集的形式为 Candidate_Set = { S^*, P^*, C^*, f^* }，其中 S^*通过映射 f^*决定了 P^*；而 P^*作为服务库 C 在等价关系 R 下的商集的子集，P^*中的元素与一组确定的具体 Web 服务是对应的，故而 P^*决定了具体 Web 服务集 C^*。由以上分析可知，S^*决定了候选服务集；因此，候选服务集的构造其实就是 S^*的构造，下面讨论抽象服务集 S^*的构造。

候选服务集存在是以基于功能层面的 Web 服务发现成功执行为先决条件的，假定对于用户功能需求 F_x，服务发现 Func_Based_Discovery(root, F_x)过程返回的功能结点为 F_y，抽象服务集为 S = { S_1, S_2,..., S_k }；记 S^*的构造过程为 Create(S^*)，则有：

Procedure Create(F_y, S^*)

建立一个空集合 S'和一个队列 Q，将结点 F_y 加入到队列 Q 中，

Step1. 判断队列中的元素个数是否为 0；如果不为 0，执行 Step2；否则，转向 Step3；

Step2. 从队列中取出一个元素 F_x，按照用户的选择对 F_x 进行剪枝操作；考察剪枝后 F_x 的 1-度功能集中的元素：如果是抽象服务类描述的功能，则将其放入集合 S'，否则加入到队列 Q 的队尾；转向 Step1；

Step3. 将集合 S'赋值给 S^*，即 $S^* = S'$，完成 S^*的构造，过程结束。

关于 S^*的构造过程作几点说明：第一，和基于层面的 Web 服务发现一样，候选服务集的构造过程也是一个广度优先遍历过程，只是遍历要从结点 F_y 开始，这是由领域功能分解原理所决定的；第二，S^*是抽象服务集 S 的子集，它所包含的元素只能是抽象服务类；过程 Create(F_y, S^*)通过判断一个元素是否属于抽象服务集 S 来确定是否将其加入集合 S'，这保证了集合 S'就是我们需要构造的 S^*。

3.5 基于功能层面 Web 服务发现的例子

本例基于一个旅游服务领域，按照领域所能提供的功能和系统注册的 Web 服务，执行过程 Procedure Create(root)得到的领域服务功能树如图 3.7 所示。功能树的树根 root 是领域名，即 tourism；该服务领域提供的服务功能，即 root 的 1-度功能集，包括 travel、housing 和 catering。类似地，travel 又可划分为三个子功能，它们分别是 taxi，tram 和 bicycle；housing 包括 hotel，hostel 和 apartment；catering 包括 package，buffet 和 fast food。树中的叶子结点对应抽象服务类。

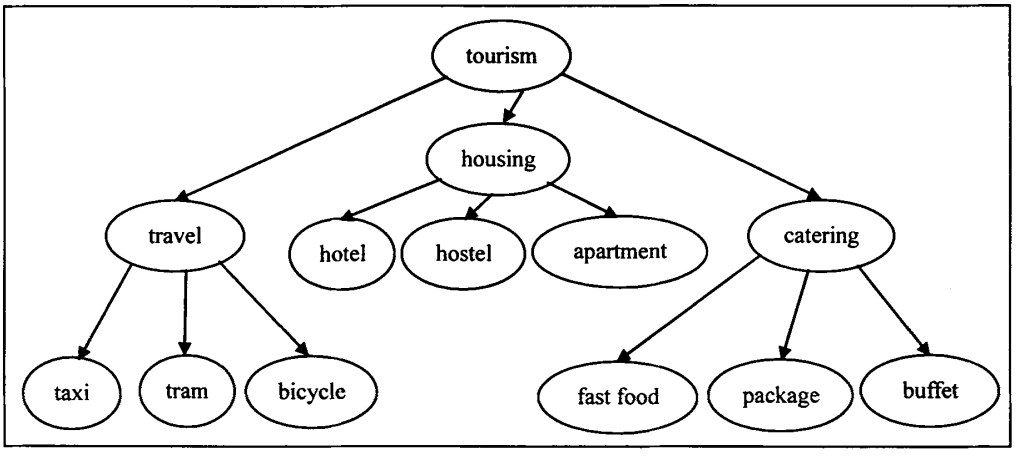

图 3.7 旅游领域服务功能树

为了方便描述，我们将叶子结点与抽象服务类一一对应，不妨设 taxi、tram、hotel、package、buffet、bicycle、hostel、apartment 和 fast food 对应的抽象服务类按顺序分别记为：S_1、S_2、S_3、S_4、S_5、S_6、S_7、S_8、和 S_9，则抽象服务集为 $S = \{S_1,S_2,\ldots,S_9\}$。

示例 1：

在旅游服务领域，比如用户想找能够提供出行工具的服务 bus，那么基于功能层面的服务发现过程 Func_Based_Discovery(root, F_x)，其中参数 root=tourism，F_x=bus，执行如下：创建存放功能结点的队列 Queue，

(1) 得到根 tourism 的 1-度功能集为 set_1={travel, housing, catering}；

(2) 将 bus 与 set_1 中的功能进行匹配，匹配失败，将 set_1 中的结点存入队列 Queue；

(3) 因为队列不为空，所以取出队首元素 travel，得到其 1-度功能集 set_2={taxi, tram, bicycle}；将 bus 与 set_2 中的功能进行匹配，仍然匹配失败，将 set_2 中的元素存入队列的队尾；

(4) 重复执行(3)的操作，直至队列为空，没有发现与用户功能需求 bus 匹配的功能结点，服务发现失败。图 3.8 是显示了输入功能需求 bus 时的运行结果。

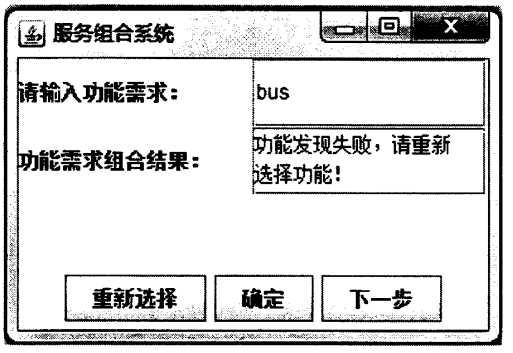

图 3.8 功能需求为 bus 时的运行结果

示例 2：

如果用户的功能需求是旅游，而且他同时表达自己不愿意使用自行车，不愿意居住招待所和公寓，同时不愿意吃快餐，那么基于功能层面的服务发现过程以及候选服务集的构造如下：

创建空集合 S^*，创建队列 Q，

(1) 基于功能层面的服务发现过程 Func_Based_Discovery(root, F_x)，设置参数 root=tourism，Fx=tourism，则服务发现成功，所返回的功能子树与领域服务功能树相同，即返回 root=tourism；

(2) 执行候选服务集构造过程 Create(F_y, S^*)，设置参数 F_y=tourism,将 F_y 加入到队列 Q 中；

(3) 队列 Q 不为空，取出队首元素 tourism，得到 tourism 的 1-度功能集 set_1={travel, housing, catering}，对比用户要求，不需要进行剪裁操作；同时由于 set_1 中的元素都不属于抽象服务类描述的功能，故将其全部加入到队列 Q 中；

(4) 判断队列不为空，此时取出队首元素 travel，得到其 1-度功能集 set_2={taxi, tram, bicycle}，对比用户要求可知，需要进行剪裁操作，删除子功能结点 bicycle；同时由于

set_2 中剩余的元素 taxi 和 tram 分别是抽象服务类 S_1 和 S_2 所描述的功能，故将其全部加入 S^*中；

(5) 类似地，通过裁剪操作删除掉 hostel 结点、apartment 结点和 fast food 结点，最终得到 S^*={taxi, tram, hotel, package, buffet}，用抽象服务类来表示就是：{S_1, S_2, S_3, S_4, S_5}，确定了 S^*就相当于确定了候选服务集。

在该用户需求下系统的运行结果为图 3.9 和图 3.10 所示：

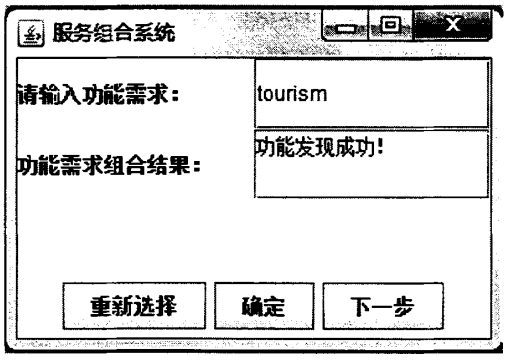

图 3.9 功能需求为 tourism 时的运行结果

这表明基于功能层面的 Web 服务发现是成功的。

组合成功!
第5个服务类的候选服务有:
服务号: 32;服务号: 33;服务号: 34;服务号: 35;服务号: 36;服务号: 37;服务号: 38;服务号: 39;服务号: 40;服务号: 41;服务号: 42;服务号: 43;服务号: 44;
第4个服务类的候选服务有:
服务号: 52;服务号: 53;服务号: 54;服务号: 55;
第3个服务类的候选服务有:
服务号: 23;服务号: 24;服务号: 25;服务号: 26;
第2个服务类的候选服务有:
服务号: 45;服务号: 46;服务号: 47;服务号: 48;服务号: 49;服务号: 50;服务号: 51;
第1个服务类的候选服务有:
服务号: 1;服务号: 2;服务号: 3;服务号: 4;服务号: 5;服务号: 6;服务号: 7;服务号: 8;服务号: 9;服务号: 10;服务号: 11;服务号: 12;服务号: 13;服务号: 14;服务号: 15;

图 3.10 S^*中抽象服务类与具体 Web 服务的对应

图 3.10 是系统构造候选服务集的过程，由图可知，系统构造的候选服务集中有 5 个抽象服务类，这和我们的分析是一致的；同时，系统以服务号的形式给出了每一类抽象服务对应的具体 Web 服务。

3.6 本章小结

本章的核心目标是介绍基于功能层面的服务发现。作为先导理论，首先提到 Web 服务组合技术，紧接着介绍了 Web 服务的细节描述模型和整体描述模型，并在此模型的基础上引入抽象服务类的概念；由抽象服务类组成的抽象服务集决定了候选服务集。在领域服务功能树中，由功能分解原理入手，依次介绍了领域服务功能树的构造过程、功能树结点的删除过程和添加过程。基于领域服务功能树的模型，讨论了基于功能层面 Web 服务发现的定义，描述了服务发现的过程，给出了候选服务集的构造算法；最后通过一个旅游服务领域的例子说明了该算法的有效性。本章在全文中具有承上启下的重要意义，候选服务集的构造是后续 Web 服务选择工作的基础。

第四章 基于 QoS 的 Web 服务选择

基于功能层面的 Web 服务发现最终形成候选服务集，候选服务集主要描述两个方面的信息：一是用户业务逻辑的功能需求应该由哪些个抽象服务类来实现，二是每一个抽象服务类对应于哪些可用的具体 Web 服务。在候选服务集模型中，S^*集合中每一个元素都代表一个抽象服务类，而每一个抽象服务类都对应着一组功能相似的具体 Web 服务，且这组 Web 服务所构成的集合是服务库 C 在等价关系 R 下商集的子集。于是，一个问题就自然而然地被提出来了：既然一个抽象服务类对应着多个具体的 Web 服务，那究竟选择哪一个来为用户提供最终服务呢？这是一个对功能相似 Web 服务进行再选择的过程，其中一个重要的依据就是服务质量 QoS，本章将讨论在候选服务集已创建的前提下，基于 QoS 的 Web 服务选择。

4.1 服务质量 QoS 概述

在实际 Web 应用中，对 Web 服务有多种不同的非功能要求，包括服务的可用性、服务性能、服务可伸缩性、服务安全性等等，这些技术要求能否满足往往决定了服务的成败；Web 服务的非功能属性是 SOA 应用程序能够可靠运行的重要保证，同时也是 SOA 程序提供多种级别一致服务的基础。

4.1.1 QoS 在 Web 服务中的地位和 QoS 定义

在第三章关于 Web 服务的模型描述中曾经提到，一个 Web 服务在宏观上可用一个二元组来描述，即 W = { F, Q }，其中 F 是 Web 服务的功能属性，描述了 Web 服务能够实现什么样的功能、提供什么样的服务；Q 是服务的非功能属性，用以描述 Web 服务除功能以外的其他属性。在 Web 服务的非功能属性中，最重要的是服务质量 QoS(Quality of Service)。

QoS 是 Web 服务的一种能力，它是能响应预期请求、并能够以符合服务提供者和用户预期的质量完成相关任务的能力[47]。现在，QoS 是反映 Web 服务效用和性能的重要准则，也是不同服务提供商针对同一类 Web 服务的重要卖点。随着 Internet 上功能属性相似、非功能属性不同的 Web 服务越来越多，QoS 越来越成为服务消费者关心的重要内容，因此，它是进行 Web 服务选择必须要考虑的重要因素。

4.1.2 常用的 QoS 指标介绍

QoS 的指标有很多种，这里仅介绍几种常用的指标，同时这也是本文研究过程中所考虑到的 QoS 指标[56]：

(1) 可用性：Web 服务的可用性用于表示服务能够正常运转的概率，取值介于 0 和 1 之间；一个 Web 服务的可用性越高，意味着其能够正常提供服务的可能性越大。可用性还与修复时间 TTR(time-to-repair)相关，TTR 表示故障 Web 服务的修复时间；

(2) 可访问性：任何 Web 服务的负荷都是有限的，它不可能为无限多的消费者同时服务；可访问性就是用以描述在某个时间点上，服务能够成功地被实例化的概率，同样它的取值介于 0 和 1 之间；如果一个 Web 服务的可访问性较低，那么服务消费者要么等待，要么选用其他可替换的 Web 服务，但无论如何这意味着时间上的浪费；

(3) 可靠性：可靠性是指 Web 服务在一个合理时间范围内完成客户请求的能力，是一个长期指标，通过周期性统计服务的成功率得到；

(4) 价格：服务提供商对外提供的 Web 服务，大多都不是无偿服务，需要服务消费者购买服务；服务的价格属性是指为了完成业务需求而调用某 Web 服务时，服务消费者要付出的金钱代价；

(5) 响应时间：是指服务客户端从发出服务请求开始，到收到服务响应为止之间的时间间隔；服务的响应时间是衡量 Web 服务性能的重要指标。

以上给出的 QoS 指标都是通用指标，适用于任何领域的 Web 服务，本文在研究基于 QoS 的服务选择时只考虑通用 QoS 指标。

4.1.3 QoS 指标的分类

指标的高低在不同的角度来看待会有不一样的意义，比如价格指标，服务的提供者希望价格指标越高越好，而服务的消费者则恰恰相反；考虑到本文的工作是要提高用户的服务满意度，因此从用户的角度出发考虑 QoS 指标。

从用户的角度来看，QoS 指标可以分为两大类[57]：

(1) 积极 QoS 指标：数值越大，越能够符合用户要求的 QoS 指标称为积极 QoS 指标；比如可用性、可靠性和可访问性都是积极指标，这些指标的数值越高，用户的服务体验越好；

(2) 消极 QoS 指标：和积极指标相反，数值越小，越能够符合用户要求的 QoS 指标称为消极指标；响应时间和价格从用户的角度来看都是消极指标。

基于 QoS 的服务选择一个重要的原则就是选择高积极 QoS 指标、低消极 QoS 指标的 Web 服务。

4.2 服务质量 QoS 的评价

基于功能层面的 Web 服务发现形成候选服务集；基于 QoS 的 Web 服务选择的目标就是利用 Web 服务的 QoS 属性，从功能相似、非功能属性不同的 Web 服务中，选出能够满足用户服务质量需求的具体 Web 服务或服务集。

本文只关注 5 个 QoS 属性，即价格、响应时间、可靠性、可用性和可访问性，并把它们分别记作 cost、res、rel、use 和 vis；为了描述方便起见，用一个 n 维向量描述一个具有 n 个质量属性的 Web 服务的 QoS。比如对于具有 5 个质量属性的 Web 服务，它的 QoS 可以记为：$qos=(cost,res,rel,use,vis)$。

4.2.1 用户 QoS 约束的表达

服务选择必须考虑到用户的需求，用户对服务质量的需求可以通过一系列的约束条件给出[58]，这些约束条件用不等式组 4.1 的形式表示：

$$\begin{cases} Cost \le C \\ res \le RES \\ rel \ge REL \\ use \ge U \\ vis \ge V \end{cases} \tag{4.1}$$

在不等式组 4.1 中，每一个不等式都表示用户对某项 QoS 属性的要求，不等式右边是用户给出的约束值。用户以不等式组(4.1)给出的 QoS 约束条件是基于 QoS 服务选择的依据。

但是必须看到，以不等式组(4.1)给出的用户 QoS 需求存在两点不足之处：

(1) 用户对 QoS 属性的约束是逐项给出的，并不能给出对 Web 服务整体的 QoS 约束条件，这就需要我们根据该不等式组按照一定策略获得对某个 Web 服务整体 QoS 的评价信息；

(2) 用户对各项 QoS 属性一般会有不同的重视程度，这种不同的重视程度应该在基于 QoS 的服务选择过程中体现，但显然该不等式组不能提供这样的一种表达机制。

解决好这两个问题是进行基于 QoS 服务选择的基础。

4.2.2 Web 服务 QoS 总体评价信息——用户 QoS 满意度

我们尝试定义一个对 Web 服务 QoS 的整体评价标准，这个标准应该建立在一个 Web 服务真实的 QoS 属性和用户给出的 QoS 约束的基础之上。考虑 Web 服务的某一 QoS 属性，如果其真实值与用户的期望值差别越大，用户对该项 QoS 属性的满意度越低，我们基于此定义 QoS 指标满意度。

定义 4.1 一个 Web 服务的 QoS 指标满意度是指其某一指标满足用户对该项指标约束的程度，是一个介于 0 和 1 之间的值；取 0 表示完全不满意，取 1 表示完全满意。

QoS 指标满意度的计算要区分指标的性质，即它是消极 QoS 指标还是积极 QoS 指标，其具体的计算方法如下：假设某 QoS 指标 p_i 的实际值是 x，用户给出的约束值为 X，QoS 指标满意度记为 $f(p_i)$，则有，

若 p_i 为消极 QoS 指标，

$$f(p_i)=\begin{cases}1 & x\le X\\ e^{\frac{X-x}{X}} & x>X\end{cases} \tag{4.2}$$

若 p_i 为积极指标，

$$f(p_i)=\begin{cases}1 & x\ge X\\ e^{\frac{x-X}{X}} & x<X\end{cases} \tag{4.3}$$

QoS 指标满意度的定义和计算，为总体评价一个 Web 服务的 QoS 奠定了基础，下面定义服务 QoS 的总体评价信息——用户 QoS 满意度的概念。

定义 4.2 一个 Web 服务的用户 QoS 满意度是其对应的所有 QoS 指标满意度的加权平均。

这里所说的权值，反映了不同 QoS 指标的重要程度，它可以用一个权值向量来表示，即 $weight=\{w_1,w_2,\ldots,w_n\}$。如果 $p_i(i=1,2,\ldots,n)$为 Web 服务 ws 的 n 个 QoS 指标，记用户 QoS 满意度为 f(ws)，根据定义 4.2，则有：

$$f(ws)=\sum_{i=1}^{n}w_i\times f(p_i) \tag{4.4}$$

至此我们已经讨论了服务 QoS 的评价标注——用户 QoS 满意度。从用户 QoS 满意度的计算公式(4.4)可知，体现各项 QoS 指标重要程度的权值向量 weight 起着举足轻重的作用，如何获取这个权值向量是问题的关键所在。

4.3 组合 QoS 权重和权重分解

现在讨论如何得到体现 QoS 各项指标重要程度的权重向量 weight。在考虑 QoS 某项指标的重要程度时，首先想到的是用户对该项指标的重视程度，即用户的偏好，它是用户主观意愿的体现[59]，权重向量 weight 应该包括对用户偏好的描述。同时，权重向量完全由用户来决定是不现实的：用户的偏好属于主观想法，通常是用语言文字来描述的，需要对其进行量化处理；具体到某一领域的 Web 服务，其 QoS 属性可能会涉及到与该领域相关的专业化指标，作为普通用不是总能准确理解这些指标的意义和重要性，这就使得单纯的用户偏好可能会带有一定的盲目性。因此，权重向量 weight 应该包含领域相关的、不随用户偏好变化的权重因素，该因素由服务所属领域的专家给出，具有权威性和相对稳定性。

我们把权重向量 weight 称为组合 QoS 权重，它是两个权重的组合：一是由领域专家给出的、依赖于某一服务领域的权重，我们称之为静态权重；二是由用户给出的、纯粹反映用户主观意愿的权重，我们称之为动态权重；接下来分别讨论两种权重的获得方法。

4.4 静态权重

由领域专家给出的静态权重可以有效修正用户偏好的不合理因素，它基于对某一领域的深刻立即，更具客观合理性；本文采用层次分析法，由领域专家打分的方式计算静态权重。

4.4.1 层次分析法

首先介绍层析分析法的相关概念。层次分析法[60]AHP(Analytic Hierarchy Process)是一种解决多目标问题的决策分析方法，它通过决策者的经验判断、权衡各指标间的相对重要程度并给出合理的权值，是一种将定性和定量结合的方法；其特点是在对复杂决策问题的本质、影响因素及其内在关系等深入分析的基础上，利用较少的定量信息使决策的思维过程数学化。

在使用 AHP 时，为了能够获得定量的结果，需要用判断矩阵表示因素间重要性的比较；假设判断矩阵为 $A=(a_{ij})_{n\times n}$，则其元素可用 Santy 的 1-9 标度法给出，如表 4.1 所示。

表 4.1 判断矩阵元素 a_{ij} 的标度方法

标度	含义
1	表示两个因素相比，具有同样的重要性
3	表示两个因素相比，一个因素比另一个稍微重要
5	表示两个因素相比，一个因素比另一个明显重要
7	表示两个因素相比，一个因素比另一个强烈重要
9	表示两个因素相比，一个因素比另一个极端重要
2, 4, 6, 8	上述两相邻判断的中值
倒数	因素 i 与因素 j 的判断为 a_{ij}，则 j 与 i 的判断 $a_{ji}=1/a_{ij}$

关于判断矩阵的一些说明：

(1) 判断矩阵 A 必须满足一致性要求。矩阵 A 满足一致性要求当且仅当对于 A 中任意元素 a_{ij}，a_{jk} 和 a_{ik}，有 $a_{ij}\times a_{jk}=a_{ik}$ 成立；要求矩阵必须满足一致性要求是为了避免元素间重要程度不协调情况的出现。

(2) 在实际的应用中，并不需要判断矩阵 A 完全满足一致性要求，但必须对其进行一致性检验。为了实现对判断矩阵 A 的一致性检验，必须引入几个概念：

一致性指标：$CI=\dfrac{\lambda-n}{n-1}$，其中，$\lambda$ 为矩阵 A 的最大特征向量，n 为矩阵 A 的阶数；CI 越接近于 0，表示矩阵 A 的一致性越好。

一致性比率：$CR=\dfrac{CI}{RI}$，其中，CI 为一致性指标，RI 为随机一致性指标，可以通过查表得到；一般地，当一致性比率 $CR=\dfrac{CI}{RI}<0.1$ 时，可认为矩阵 A 的不一致程度在容许范围之内，通过一致性检验。

4.4.2 静态权重的计算

假设只考虑 QoS 的 5 个指标，它们分别是 cost、res、rel、use 和 vis，运用 AHP 获得静态权重的步骤如下：

(1) 建立 QoS 指标层次模型，如图 4.1 所示：

其中 QoS 为目标层，是要解决的目的；准则层从 cost、res、rel、use 和 vis 等 5 个因素对目标层进行层次分解；该模型可获得目标层之下的准则层各元素间的相对重要程度。

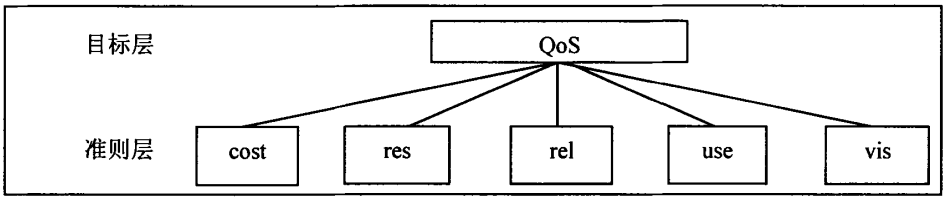

图 4.1 QoS 指标层次模型

(2) 构造表 4.2 所示的评价矩阵，即 4.4.1 节中提到的判断矩阵；领域专家通过自己的经验和判断给出矩阵中的元素值；因为本文考虑 5 个 QoS 指标，故而所构造的判断矩阵是一个 5×5 的方阵，且矩阵元素间满足以下的数值约束关系：

$X_{ii}=1\,(i=1,2,3,4,5)$，即对角线元素为 1，含义是任一指标相对于自身的值为 1；

$X_{ij}=1/X_{ji}\,(i=1,2,3,4,5;\ j=1,2,3,4,5)$，即关于主对角线对称的元素值互为倒数。

表 4.2 判断矩阵

指标	价格	响应时间	可提供性	可靠性	可用性
价格	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
响应时间	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}
可提供性	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}
可靠性	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}
可用性	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}

(3) 对步骤(2)所构造的判断矩阵进行一致性检验：因为此处的判断矩阵为 5×5 矩阵，可通过查表获得其随机一致性指标，即 $RI=1.12$；如果经过计算，一致性比率满足条件 $CR=\frac{CI}{RI}<0.1$，则认为通过检验；否则按照文献[61]中所提供的方法进行调整。

(4) 计算静态权重向量，它等于判断矩阵最大特征值所对应的特征向量。

4.5 动态权重和组合 QoS 权重计算

4.5.1 动态权重的计算

用户通常习惯于用自然语言表达自己的偏好，比如用一组词汇的集合{非常重要，重要，一般，不重要，可以忽略}表示对重要程度的要求。这些概念属于模糊集合的范畴，可采用三角模糊数[62]进行描述。

获取动态权重的步骤如下：

(1) 用户根据自己的偏好对不同的 QoS 指标进行重要性的选择，选择的范围为{非常重要，重要，一般，不重要，可以忽略}；

(2) 将用户选择的结果映射为三角模糊数；本文用如下的三角模糊数集表示(1)中关于 QoS 指标重要程度的描述：{[10,10,8],[5,7,9],[3,5,6],[1,3,4],[1,1,2]}；

(3) 得到的三角模糊数集反映了用户对各项 QoS 指标的偏好，对其进行量化得到对应的实数集；实数集中的元素的顺序与用户对 QoS 指标重要性进行选择的顺序对应，把这些实数按顺序组成一个向量 $media=(m_1,m_2,m_3,m_4,m_5)$，其中 m_i $(i=1,2,3,4,5)$ 是第 i 项 QoS 属性对应的三角模糊数量化的结果；

(4) 对得到的向量 $media=(m_1,m_2,m_3,m_4,m_5)$ 进行归一化处理，即得到动态权重向量。

4.5.2 组合 QoS 权重计算

组合 QoS 权重照顾了用户的主观偏好，又能兼顾领域特性的客观要求；假设只考虑 5 个指标的 QoS，且静态权重向量表示为：

$$s\_weight=(s_1,s_2,s_3,s_4,s_5) \tag{4.5}$$

其中 s_i $(i=1,2,3,4,5)$ 为第 i 项 QoS 指标对应的静态权重；

动态权重向量表示为：

$$d\_weight=(d_1,d_2,d_3,d_4,d_5) \tag{4.6}$$

其中 d_i $(i=1,2,3,4,5)$ 为第 i 项 QoS 指标对应的动态权重；

对于静态权重向量和动态权重向量的组合，本文采用积的方法得到组合权重向量，并对其归一化，得到最终的组合 QoS 权重向量表示为：

$$c\_weight=\left(\frac{s_1\times d_1}{\sum_{i=1}^{5}s_i\times d_i},\frac{s_2\times d_2}{\sum_{i=1}^{5}s_i\times d_i},\frac{s_3\times d_3}{\sum_{i=1}^{5}s_i\times d_i},\frac{s_4\times d_4}{\sum_{i=1}^{5}s_i\times d_i},\frac{s_5\times d_5}{\sum_{i=1}^{5}s_i\times d_i}\right) \tag{4.7}$$

通过公式(4.5)、(4.6)、(4.7)可计算出组合 QoS 权重向量。

4.6 基于 QoS 的服务选择方法

对于一个 Web 服务组合来说，基于 QoS 的服务选择要综合考虑多个服务的各项 QoS 属性，使之满足用户给出的总体 QoS 约束，这是一个多目标优化问题。遗传算法 GA(Genetic Algorithm)作为一种智能优化算法，因为其独有的优势在多目标优化问题中得到广泛应用；本节以 Web 服务 QoS 评价标准——用户 QoS 满意度入手构造遗传算法解决基于 QoS 的服务选择问题。

4.6.1 问题描述

本文借鉴文献[58]的方法将基于 QoS 的 Web 服务选择问题描述如下：只考虑 5 个 QoS 指标，它们分别是 Cost、res、rel、use 和 vis；假定 Web 服务组合由 n 个抽象服务类构成，则 Web 服务组合的各项 QoS 指标可以按照公式组(4.8)计算得到：

Cost 计算： $Cost=\sum_{i=1}^{n}Cost_i$，其中 $Cost_i\ (i=1,2,...,n)$ 为第 i 个 Web 服务对应的 $Cost$；

res 计算： $res=\sum_{i=1}^{n}res_i$，其中 $res_i\ (i=1,2,...,n)$ 为第 i 个 Web 服务对应的 res；

rel 计算： $rel=\prod_{i=1}^{n}rel_i$，其中 $rel_i\ (i=1,2,...,n)$ 为第 i 个 Web 服务对应的 rel；

use 计算： $use=\prod_{i=1}^{n}use_i$，其中 $use_i\ (i=1,2,...,n)$ 为第 i 个 Web 服务对应的 use；

vis 计算：$vis=\prod_{i=1}^{n}vis_i$，其中 $vis_i\ (i=1,2,...,n)$ 为第 i 个 Web 服务对应的 vis； (4.8)

用户的 QoS 约束通过不等式组(4.1)描述。

4.6.2 算法构造

染色体编码方式的选择、适应度函数的选择、终止条件的设定是构造遗传算法 GA 的重要环节，直接决定了构造的算法性能优劣，下面分别讨论本文对这几个环节的构造：

(1) 染色体编码

采用二进制分段编码方式：对于包括 n 个子服务的 Web 服务组合，染色体划分成 n 段，每一段都是相应子服务类预留位置。同时，采用自适应长度的编码，具体做法是：

设置每一个子服务的编码段长度为 $L_i = \lceil N_i \rceil$，其中 N_i 是第 i 个抽象服务类中所包含的具体 Web 服务个数； Web 服务组合对应的编码长度为 $L = \sum_{i=1}^{n} L_i$ 。

这样做的好处是，既能够覆盖整个搜索空间，又能够尽量减少资源开销，提高算法效率。

(2) 适应度函数

从用户 QoS 满意度入手设计适应度函数，这样可以使得算法搜索的方向朝着全局最优用户 QoS 满意度的方向进行；本文选取的适应度函数就是用户的 QoS 满意度，即：

$$obj(x) = f(ws) \tag{4.9}$$

由前几节所介绍的相关公式可知，式(4.9)是可求解的。

(3) 算法终止条件的设定

用户 QoS 满意度达到阈值要求时算法终止；如果达不到阈值，若发现群体中一定比例的个体已经是同一个个体，则结束算法迭代。

(4) 其他设定

采用轮盘赌方法实现选择；构造两点交叉，交叉率设置为 0.5；对染色体每一位二进制数值按概率执行变异操作，变异概率设为 0.015.

4.7 基于 QoS 的服务选择实验

4.7.1 实验背景

本实验基于第三章第 3.5 节介绍的示例二。用户的功能需求 tourism 得到满足，而且基于功能层面的服务发现成功构造了候选服务集。在候选服务集中，S^*={taxi, tram, hotel, package, buffet}，用抽象服务类来表示就是{S_1, S_2, S_3, S_4, S_5}；由图 3.10 给出的抽象服务类与具体 Web 服务对应关系可知，5 个抽象服务类对应的具体 Web 服务个数分别是 22 个、7 个、4 个、4 个和 13 个，进而可知算法搜索的解空间大小为 22×7×4×4×13=32032。

4.7.2 实验过程

首先进入图 4.2 所示的系统界面，点击“我是专家”进行静态权重 *s_weight* t 的设置：

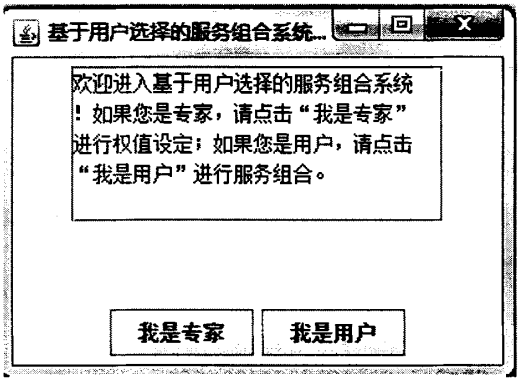

图 4.2 系统初始化界面

接着按照图 4.3 所示的界面进行专家打分：

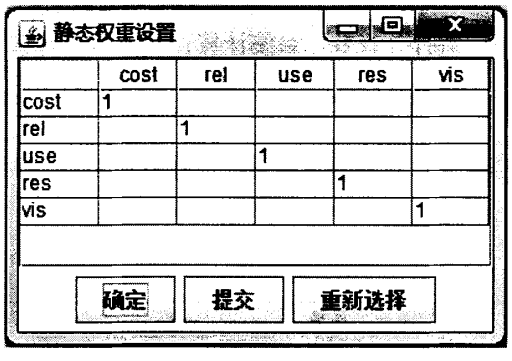

图 4.3 专家打分界面

专家根据领域的经验完成打分，提交完成静态权重设置，如图 4.4 所示：

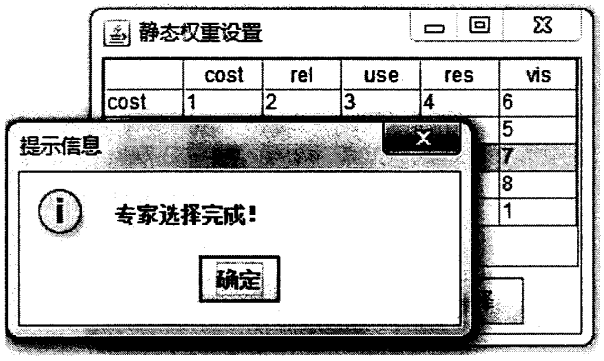

图 4.4 专家打分提交界面

系统需对专家打分后形成的判断矩阵进行一致性检验，即比较一致性比率 CR 是否小于 0.1；如果满足条件，系统将完成静态权重的计算；否则将要求重新输入；此次实验的静态权重设置结果如图 4.5 所示：

```
CR=0.0029448137824296984
0.39177400980141763
0.2508292082505555
0.18402125871726333
0.13494515378321595
0.0384303694475476
```

图 4.5 一致性比率和静态权重

由图 4.5 知，CR<0.1，故打分有效，系统计算得到的静态权重向量为：

$s\_weight = (0.392,0.251,0.184,0.135,0.038)$，此为保留 3 位小数的结果。静态权重设置之后，系统将进入动态权重设置界面，由用户选择各项 QoS 指标的重要性，如图 4.6 所示：

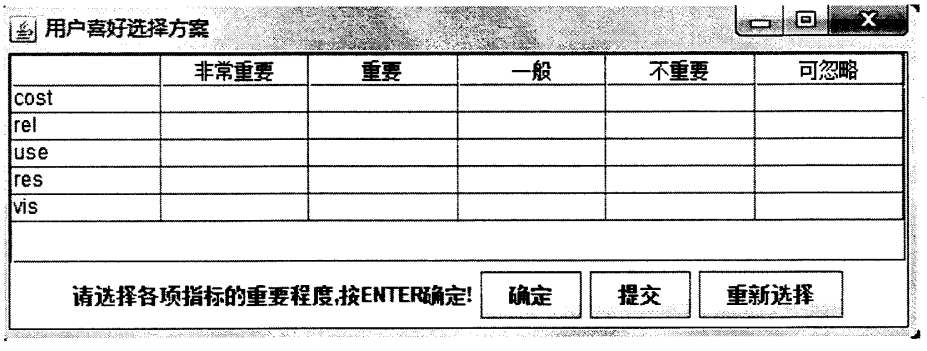

图 4.6 用户对 QoS 指标重要程度选择

在用户选择完毕后，系统会计算出动态权值如图 4.7：

```
userweight0:0.17391304347826086
userweight1:0.391304347826087
userweight2:0.30434782608695654
userweight3:0.08695652173913042
userweight4:0.04347826086956521
```

图 4.7 系统计算的动态权值

由图 4.7 可知，系统计算的动态权值向量为：$d\_weight = (0.174,0.391,0.304,0.086,0.043)$，同样为保留 3 位小数的结果；

系统会根据计算得到的 d_weight 和 s_weight，利用公式(4.7)计算得到组合 QoS 权重向量 c_weight。

接下来用户输入其 QoS 约束，如图 4.8 所示：

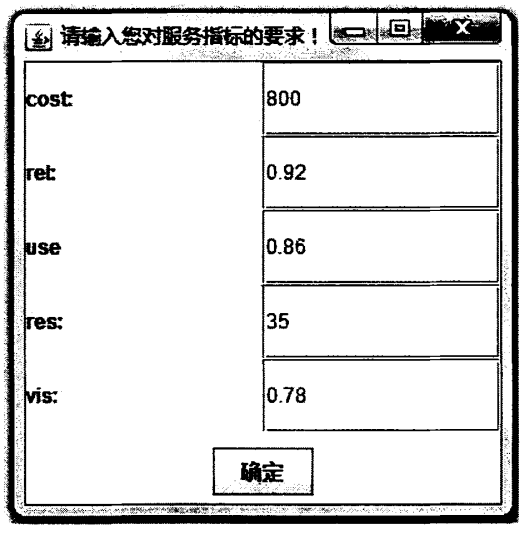

图 4.8 用户 QoS 约束

用户 QoS 约束输入完毕之后，系统将运行本章 4.6.2 节构造的遗传算法，运行结果如图 4.9 所示：

图 4.9 所描述的信息分别是遗传算法每一次迭代过程中，种群中最优个体和最劣个体的适应值。

```
第1代: 0.9498893665155198; worst:0.8546219960526336
第2代: 0.9485538570176961; worst:0.8546219960526336
第3代: 0.9333591828192115; worst:0.8546219960526336
第4代: 0.9175418551415035; worst:0.8546219960526336
第5代: 0.9160600239548803; worst:0.8546219960526336
第6代: 0.9177108444305385; worst:0.8527797516396177
第7代: 0.9227048845466562; worst:0.8527797516396177
第8代: 0.9160213178513967; worst:0.8353929307080388
第9代: 0.923403912789125; worst:0.8353929307080388
第10代: 0.9253656906915603; worst:0.8353929307080388
第11代: 0.922996252037571; worst:0.8353929307080388
第12代: 0.9354494626614487; worst:0.8353929307080388
第13代: 0.9374530280091105; worst:0.8353929307080388
第14代: 0.9398165591124776; worst:0.8353929307080388
```

图 4.9 遗传算法迭代示意图

限于篇幅原因，图 4.9 并未完全列出遗传算法的迭代全过程，最后服务选择的输出结果如图 4.10 所示：

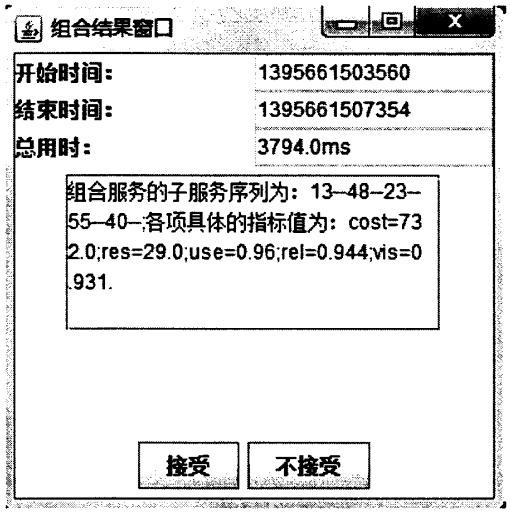

图 4.10 服务选择结果示意图

至此，基于 QoS 的服务选择过程已完成；图 4.10 中显示了选择用时、抽象服务类对应的具体服务序号、以及组合 Web 服务的各项 QoS 指标；如果用户对选择结果不满意，可以点击“不接受”，通过设置不同的动态权值和 QoS 约束进行再选择。

4.7.3 实验分析

该实验展示了本章所论述的整个基于 QoS 的服务选择过程;在实验的结果中我们不难发现，虽然并不是用户的每一项 QoS 约束都能够得到完全满足，但是权重越高的 QoS

指标被满足的可能性越高、被满足的程度也越好。这充分显示了该算法在提高用户满意度方面的积极作用。

我们对该算法重复做了多次试验，图 4.11 描述了当其他权值打分固定不变，某单一 QoS 指标权重打分从 1 到 10 变化时，搜索结果中 QoS 指标满意度的变化规律；为了不失一般性，我们选用 cost 和 vis 分别作为消极 QoS 指标和积极 QoS 指标的代表。

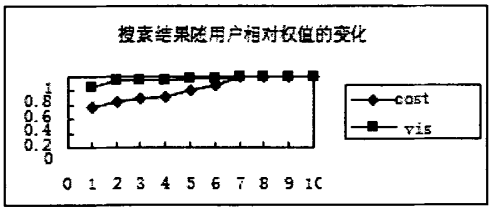

图 4.11 QoS 指标满意度随输入权重的变化规律

从图 4.11 不难看出，当某一项指标的权重较小时，在算法的输出结果中，对应的 QoS 指标满意度会比较低；当该项的权重增加时，对应的 QoS 指标满意度逐渐提高；这说明算法的过程是可控的，用户可以结合自身的需求和算法结果，以调整权值的方式有效地选择符合要求的服务。

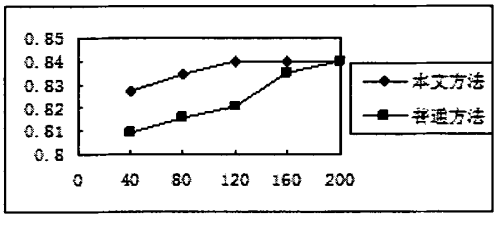

图 4.12 用户满意度随算法迭代次数的变化规律

图 4.12 给出了用户满意度随遗传算法迭代次数的变化规律。不难看出，相对于传统的以全局 QoS 最优为目标的优化算法，本文所提的算法具有更快地收敛速度，能够更有效率找到用户满意度高的 Web 服务。

4.8 本章小结

本章在第三章工作的基础上，全面介绍了基于 QoS 的 Web 服务选择方法。提出了组合 QoS 权重的分解方式，即分解为静态权重和动态权重，并分别给出了两种权重的计算方法；定义了 Web 服务的 QoS 总体评价标准——用户 QoS 满意度的概念，并以此为出发点，构造用以实现基于 QoS 服务选择的遗传算法 GA。本章的最后通过一个实验全程演示了文中所介绍的所有概念、理论和应用，最后分析了算法的有效性和优势。

第五章 基于服务内容的 Web 服务选择和组合选择模型

本文在定义 3.2 中给出了 Web 服务的的描述模型，即$W=\{F,Q\}$，其中F为 Web 服务的功能属性，Q为非功能属性。依据该模型的描述，本文在第三章和第四章分别介绍了基于功能层面的 Web 服务发现和基于 QoS 的 Web 服务选择，实现了候选服务集的构造和以此为基础的服务选择。但在某些情况下，前面的工作似乎不能完全满足用户的需求。还是拿第 3.5 节中的示例二来说，用户在选择了 hotel 这一功能的服务后，他可能会对服务的内容有自己的要求，比如房间的类型、房间内的设施等。如何描述用户的这类需求并用于 Web 服务选择显得很有必要，本章将就此问题展开讨论。

5.1 Web 服务描述模型的扩展

在某些特殊的场合下，除了 Web 服务的功能和服务质量 QoS 之外，用户还有可能会关心 Web 服务能够提供什么样的服务内容信息，他们同样希望能够将自己对这类信息的偏好作为服务选择的部分依据。然而，这类信息往往反映服务的运行时信息，无法在定义 3.2 的模型中进行描述。语义 Web 服务技术，比如 OWL-S(Web Ontology Language for Services)，可能通过其 precondition 等属性描述这些细节信息，但源于语义 Web 本身及其应用的局限性，这种描述不具有一般性和实用性。

因此，为了描述用户对 Web 服务内容方面的需求，需要对 Web 服务的描述模型进行扩展，定义 5.1 描述了扩展后的 Web 服务描述模型。

定义 5.1 扩展后的 Web 服务可用一个三元组表示，即$W=\{F,Q,C\}$，其中：

(1)F表示服务 W 的功能属性；

(2)Q表示服务 W 的非功能属性；

(3)C表示服务的内容描述。

通过在 Web 服务描述模型中引入元素 C，我们希望能够为 Web 服务内容描述提供一个形式完好的、统一的方法。

5.2 描述服务内容的 CoS 模型

为了实现基于服务内容的服务选择，必须首先构造服务内容的描述模型，我们将其记作 CoS(Content of Service)；本节主要介绍表示服务运行时内容信息的 CoS 的相关概念、性质和操作等。

5.2.1 CoS 的概念

定义 5.2 对于一个给定的 Web 服务 W，它的 CoS 是一个二元组，即 $CoS_W = \{S_j, item\_list)\}$，其中：

(1) S_j 表示 Web 服务 W 所属的抽象服务类；

(2) $item\_list$ 是一系列内容项的集合，它的形式为 $item\_list = (item_1, item_2, ..., item_n)$，每一个内容项 $item_i$ $(i = 1,2,...,n)$ 都表示服务 W 某一方面的内容信息，n 该服务所包含的内容项的数目。

为了便于理解，这里以 3.5 节中 tourism 服务中关于宾馆预订服的服务来举例说明。我们已知提供 hotel 服务的抽象服务类为 S_3，假设该类服务的内容项包括 room_type、air_conditioner、wakeup_serv 等 3 项，那么提供宾馆预订的某一 Web 服务 ws 的 CoS 可以表示为：

$CoS_{ws} = \{S_3, item\_list_{ws}\}$，其中服务内容项集合的形式为：

$item\_lsit_{ws} = (room\_type, air\_conditioner, wakeup\_serv)$。

在 $item\_list_{ws}$ 中，第一个内容项表示宾馆预订服务 ws 能够提供什么样的房间类型，第二项和第三项分别表示了它能够提供的特殊服务，比如时候有空调、是否提供叫醒服务等。

5.2.2 预定义 CoS 模型

从定义 5.2 不难看出，Web 服务的 CoS 模型的形式取决于它所属的抽象服务类；换句话说，同一抽象服务类对应的所有具体 Web 服务具有相同形式的 CoS 模型，它们彼此之间的区别仅仅在于内容项的取值不同。因此，我们可以针对每一个抽象服务类，定义唯一由该抽象服务类决定的 Web 服务的 CoS 形式，称之为预定义 CoS 模型。

定义 5.3 预定义 CoS 模型是针对一个抽象服务类 S_j 的 CoS 共享形式模型；它描述了抽象服务类 S_j 所对应的所有具体 Web 服务 CoS 的形式，包括 *item _list* 的 3 个方面的描述：

(1) *item _list* 中的内容项数 n；

(2) *item _list* 中的内容项名称；

(3) *item _list* 中内容项的顺序。

假设抽象服务类 S_j 的预定义 CoS 模型中，*item _list* 共有 n 个内容项，这些内容项的名称和排列顺序为 $(item_1, item_2, ..., item_n)$，则 S_j 对应的具体 Web 服务的 CoS 可通过对 *item _list* 中各内容项赋值得到。表 5.1 显示了抽象服务类 S_j 对应的一个 *item _list* 赋值：

表 5.1 S_j 的 item_list 赋值

item_name	$item_1$	$item_2$	....	$Item_n$
value1	v_{11}	v_{12}	....	v_{1n}
value2	v_{21}	v_{22}	....	v_{2n}
value3	v_{31}	v_{32}	....	v_{3n}
....	....	....	....	....

从表 5.1 可以看出，每一个内容项都可能会对应着多个赋值；而且，这些内容项之间不是独立的，因此对一个 *item _list* 的赋值只能以向量的形式给出，我们称之为赋值向量。显然，赋值向量具有和 *item _list* 相同的维数，且每一维的取值类型由 *item _list* 中对应的内容项所决定的。

5.2.3 内容项的数据类型、主项和项的排序规则

我们只考虑具有以下三种数据类型的内容项：

(1) 字符串类型：约定具有字符串类型的内容项是可枚举的；

(2) 布尔类型：具有布尔类型的内容项取值为 0 或者 1；

(3) 数字类型：具有数字类型的内容项取值为实数。

在 *item _list* 的所有内容相中，定义第一个内容项为主项；主项是 *item _list*，可以作为赋值向量的索引。

为了方便存储和查找，根据内容项的数据类型来安排预定义 CoS 模型中各内容项的顺序，即项的排序规则：

(1) 首先安排主项；

(2) 主项之后放置各布尔类型的内容项；

(3) 最后是其余字符串类型项和数字类型项。

主项是可枚举的，因此对它的赋值只能从一个给定的集合中选取；一旦选定了主项的赋值，布尔项赋值的不同组合就构成了赋值向量的不同分类，这在 CoS 的初始化和更新操作中具有重要意义。

图 5.1 显示了 3 个布尔型项的赋值组合对赋值向量的分类情况。

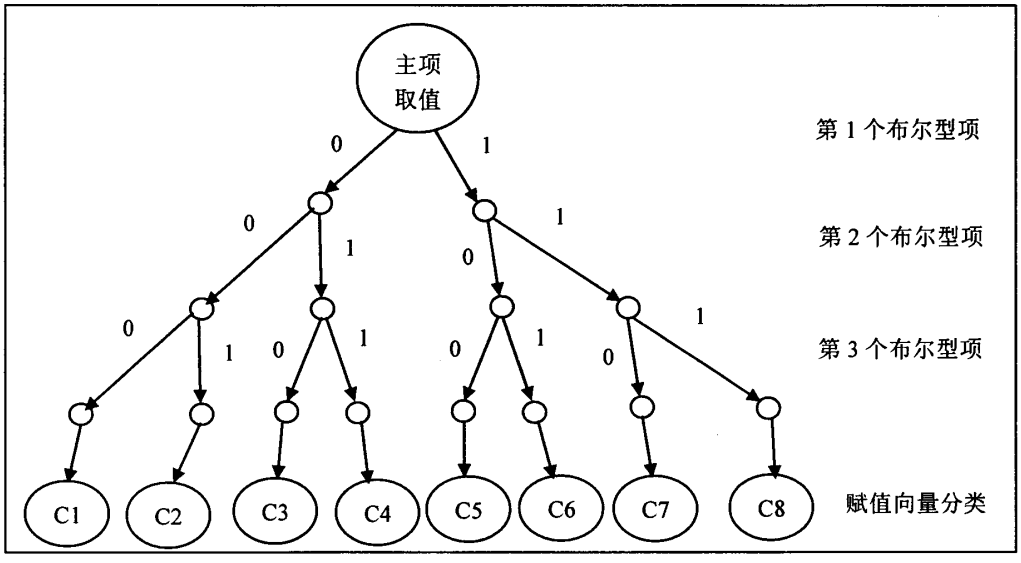

图 5.1 具有三个布尔型项的 CoS 赋值向量分类

仍以宾馆预订服务来举例，在这个抽象服务类的预定义 CoS 模型中，room_type 为主项，air_conditioner 项和 wakeup_serv 项均为布尔型项；假设主项的取值为 Single_room，那么它对应四种布尔项的取值组合，即 (0, 0)、(0, 1)、(1, 0) 和 (1, 1)，所以赋值向量共有 $2^2 = 4$ 类，它们分别是：

第一类：$value\_vector_1 = (Single\_room, 0, 0)$；

第二类：$value\_vector_2 = (Single\_room, 0, 1)$；

第三类：$value\_vector_3 = (Single\_room, 1, 0)$；

第四类：$value\_vector_4 = (Single\_room, 1, 1)$。

一般地，如果在预定义 CoS 模型中具有 n 个布尔型项，那么对应的赋值向量分类数为2^n。

5.2.4 CoS 的性质

前面介绍了 CoS 的定义和预定义的 CoS 模型，本节总结概括 CoS 的性质如下：

(1) 抽象服务类依赖性：一个 Web 服务拥有什么形式的 CoS 仅仅取决于它所属的抽象服务类，这也是提出预定义 CoS 模型的依据所在；

(2) 内容项间非独立性：在 CoS 的一个赋值向量中，一个内容项的改变往往会影响到其他项；这也是必须用向量的形式给 CoS 赋值的原因；

(3) 跨类不可公度性：服务的 CoS 依赖于它所在的抽象服务类，因此，属于不同抽象服务类的 CoS 内容指标之间具有不可公度性，不可以简简单单对其进行比较或者运算；

(4) 赋值可变性：是对某一具体的 Web 服务而言的，随着 Web 服务的更新或升级，其对应的 CoS 赋值向量可能会发生相应的改变；

(5) 模型可变性：是对某一抽象服务类而言的，指的是预定义 CoS 模型的可变性，该模型的变化往往是随着业界的标准发生变化而改变的；一般来说，模型改变的频率相对较低。

总结来说，CoS 是抽象服务类依赖的、内容项关联的、跨类不可公度的描述模型，它可能会随着具体 Web 服务的更新改进或业界标准的变化而改变。

5.3 CoS 的初始化和更新

本节介绍 CoS 的初始化和更新方法。只有对 CoS 进行初始化并保持有效的更新，才能够真实反映 Web 服务的内容，为用户进行服务选择提供依据。

5.3.1 CoS 的初始化

服务的提供者负责给出 CoS 的初始化赋值向量，CoS 的初始化在服务注册的时候同步完成；对于抽象服务类 S_j，假设其预定义 CoS 模型为 CoS_j，则图 5.2 表示了 S_j 类 Web 服务的 CoS 初始化过程。

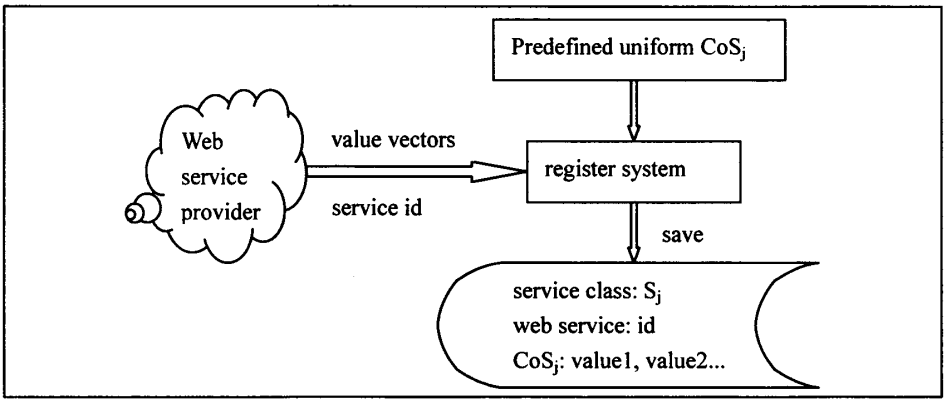

图 5.2 Web 服务的 CoS 初始化

Web 服务提供者在注册其 Web 服务时，需同时提供用以初始化该 Web 服务 CoS 的赋值向量，即图 5.2 中的 value vectors，之后系统通过赋值向量，生成每一个 Web 服务的初始化 CoS。

5.3.2 CoS 的更新

在初始化之后，每一个 Web 服务都拥有属于自己的符合预定义 CoS 模型的 CoS 值向量。但是当 Web 服务升级或更新后，先前的 CoS 值向量可能已与现在的 Web 服务内容不相符合；另外，用户在调用一个 Web 服务之后，可能会将其真实感受到的服务内容反馈给系统，而这些反馈信息存在和系统记录 CoS 不相同的可能性；因此需要对 CoS 提供更新的机制。负责提供 CoS 更新的是服务提供者或者是服务用户。

图 5.3 是抽象服务类 S_j 中 Web 服务更新 CoS 的过程。

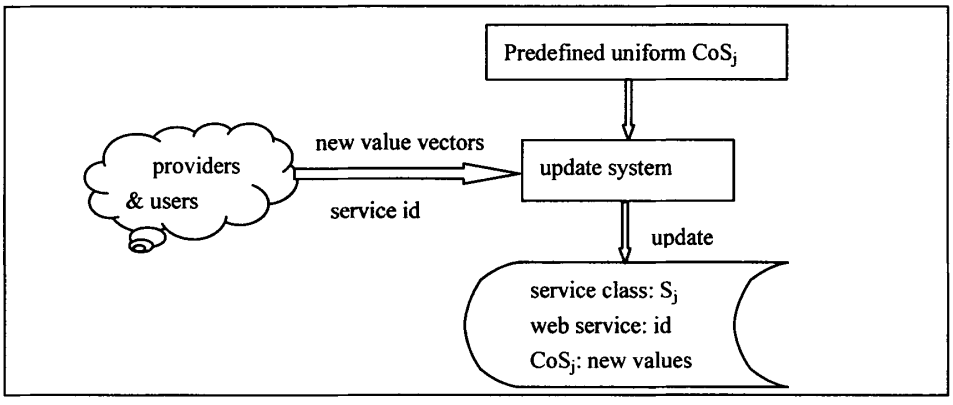

图 5.3 Web 服务 CoS 更新

CoS 的更新相对于 CoS 的初始化过程要复杂一些，下面给出 CoS 更新的操作步骤：假设抽象服务类 S_j 对应的 M 个 CoS 赋值向量全部存在集合 vector_set 中，且服务提供者或用户给出的新赋值向量为 v，则：

Step1. 解析赋值向量 v，得到其主项，记为 v.main_item；

Step2. 找出 vector_set 中所有主项为 v.main_item 的向量，并放入集合 set_1 中；

Step3. 考察赋值向量 v 的布尔项赋值组合，得到其所属的赋值向量类；判断该类是否已经在 set_1 中：如果是，执行 Step4；否则转向 Step5；

Step4. 用 v 的值更新已有的 CoS 向量，转向 Step6；

Step5. 把向量 v 加入到 value_vector 中，执行 M=M+1，转向 Step6；

Step6. 返回更新后的 vector_set 和集合中向量的个数 M，过程结束。

5.4 CoS 在服务选择中的应用

前面介绍了 CoS 的概念、预定义 CoS 模型的概念，CoS 的性质以及 CoS 的初始化、更新，本节在此基础上尝试构造一个记分法来实现基于 CoS 的服务选择。

5.4.1 问题描述

给定一个抽象服务类 S_j，它对应着 k 个具体 Web 服务；其预定义 CoS 模型为 CoS_j，用赋值向量 $value\_vector_{j,i}$ $(i=1,2,...,k)$ 表示抽象服务类 S_j 中第 i 个 Web 服务的 CoS 值。

另外，用户对服务内容的需求也必须表示成向量，我们称之为用户 CoS 需求向量，并用 $user\_vector_j$ 表示用户对 S_j 类服务的内容该需求；显然，$user\_vector_j$ 应该与 $value\_vector_{j,i}$ 具备相同的形式，它们都取决于 CoS_j 的形式。

有了这些假设，基于用户内容需求的服务选择问题可以描述为：在 S_j 的 CoS 赋值向量集合中，找到与用户 CoS 需求向量 $user\_vector_j$ 最为相似的赋值向量 $value\_vector_{j,i}\ (i=1,2,...,k)$ 以及该赋值向量所对应的 Web 服务。

显然需要定义向量间的相似性度量。

5.4.2 向量间的相似性度量

假设抽象服务类 S_j 对应的赋值向量 $value\_vector_{ji}\ (i=1,2,...,k)$ 和用户 CoS 需求向量 $user\_vector_j$ 分别具有以下的形式：

$$value\_vector_{j,i}=(v_1,v_2,...,v_n)$$

$$user\_vector_j=(u_1,u_2,...,u_n)$$

显然，$v_i(i=1,2,...,n)$ 和 $u_i(i=1,2,...,n)$ 都是对 CoS_j 定义中第 i 个内容项的描述，因此它们之间是可以比较的；定义项间相似度 $item\_match\_degree(i)$ 来描述两个向量间第 i 项的相似度，它的计算方法如下：

(1) 如果第 i 项的数据类型是字符型，因为我们在预定义 CoS 模型中规定字符型数据都是可枚举的，故而：

$$item\_match\_degree(i)=\begin{cases}1, \text{如果两个字符串相同}\\0, \text{其他}\end{cases}$$

(2) 如果第 i 项的数据类型为布尔型，则：

$$item\_match\_degree(i)=\begin{cases}1, \text{如果两个值同时为1或同时为0}\\0, \text{其他}\end{cases}$$

(3) 如果第 i 项的数据类型是数值型，则：

$$item\_match\_\deg ree(i)=\begin{cases}1, \text{如果} d_1=d_2\\ d_1/d_2, \text{如果} d_1,d_2>0 \text{且} d_1\neq d_2\\ d_2/d_1, \text{如果} d_1,d_2<0 \text{且} d_1\neq d_2\\ 0, \text{其他}\end{cases}$$

在项间相似度 $item\_match\_\deg ree(i)$ 的基础上，定义两个向量 vec_1 和 vec_2 间的向量相似度为 $vector\_match(vec_1,vec_2)$，它的计算方法如下：

$$vector\_match(vec_1,vec_2)=\frac{item\_match\_\deg ree(1)}{n}\times\sum_{i=1}^{n} item\_match\_\deg ree(i) \quad (5.1)$$

对公式(5.1)的解释如下：

该定义建立在 Web 服务对用户内容需求的主项必须满足的假设之上；公式(5.1)除去 $item\_match\_\deg ree(1)$ 因子之后，剩余的部分是两个向量对应的项间相似度的算数平均值；我们已经知道 $item\_match\_\deg ree(1)$ 是主项的相似度，它的取值只能是 0 和 1，该定义说明，只要两个向量间的主项相似度为 0，那么两个向量之间的相似度就是 0；如果主项相似度是 1，那么两个向量之间的相似度就是所有项间相似度的算术平均值。

5.4.3 Web 服务和 Web 服务组合的记分

首先介绍 Web 服务计分的概念。

定义 5.4 一个 Web 服务 ws 所对应的 CoS 赋值向量与用户 CoS 需求向量之间的相似度称为 ws 的记分，记作 $score(ws)$。

根据的定义，来自抽象服务类 S_j 的第 i 个 Web 服务 ws 记分为：

$$score(ws)=vector\_match(value\_vector_{j,i}, user\_vector_j) \quad (5.2)$$

在此基础上，可以定义 Web 服务组合的记分。

定义 5.5 一个 Web 服务组合 wc 所对应的所有子服务的记分的算术平均值，称为 wc 的记分，记作 $sum\_score(wc)$。

假设一个 Web 服务由来自 N 个抽象服务类的子服务构成，第 $j(j=1,2,...,N)$ 个抽象服务类对应的具体 Web 服务编号为 n_j，那么根据定义 5.5 有：

$$sum\_score(wc)=\sum_{j=1}^{N} score(value\_vector_{j,n_j}, user\_vector_j) \tag{5.3}$$

5.4.4 基于记分法的 Web 服务选择

由于 CoS 的跨类不可公度性，用户在给出自己的 CoS 需求向量的时候，必须按照服务类对应给出，而无法像 QoS 那样直接给出整体的约束条件。同时，来自不同抽象类的 Web 服务在 Web 服务组合中都是不可或缺的，因此无法衡量不同 Web 服务之间的重要性关系，这就使得我们不可能通过加权的方法给出总体的 CoS 评价指标。

尽管如此，我们还是希望能够对 CoS 指标提出一个评价准则，并以此为依据进行 Web 服务选择。这里讨论两个评价方法。

(1) Web 服务组合记分最大准则

该准则是指按照 Web 服务组合的记分能够取最大值的原则，从抽象服务类中选择对应的具体 Web 服务的方法。首先引入一个定理，这也是从抽象服务类选择 Web 服务的依据。

定理 5.1 一个 Web 服务组合的记分 $sum\_score(wc)$ 取得最大值的充分必要条件是，每一个抽象服务类 $S_j(j=1,2,...,N)$ 对应的具体 Web 服务 ws，其记分 $score(ws)$ 取得该类的所有服务记分的最大值。

定理 5.1 证明如下：

充分性：如果对于所有的抽象服务类 $S_j(j=1,2,...,N)$，都取其记分最高的 Web 服务 ws 来进行服务组合，显然所得到的 Web 服务组合记分能够取得最大值。

必要性：用反证法。Web 服务组合记分取得最大值，记为 $sum\_score(wc)_1$，假设有的抽象服务类没有取记分最大的 Web 服务。不妨设第 j 个抽象服务类中取了 ws_1，而该类服务中记分最大的 Web 服务为 ws_2，显然有 $score(ws_1)<score(ws_2)$，根据 Web 服务组合记分的定义有：

$$sum\_score(wc)_1=\sum_{i=1}^{j-1} score(value\_vactor_{i,n_i})+score(ws_1)+\sum_{i=j+1}^{N} score(value\_vactor_{i,n_i})$$

把第 j 个抽象服务类取的服务换成 ws_2，得到的 Web 服务组合记分为：

$$sum\_score(wc)_2 = \sum_{i=1}^{j-1} score(value\_vactor_{i,n_i}) + score(ws_2) + \sum_{i=j+1}^{N} score(value\_vactor_{i,n_i})$$

比较两式右端，只有中间项不一致，且由于 $score(ws_1) < score(ws_2)$，故而有 $sum\_score(wc)_1 < sum\_score(wc)_2$，这与 $sum\_score(wc)_1$ 是 Web 服务组合最大记分矛盾，故假设不成立；命题得证。

至此我们已经完成了对定理 5.1 的证明。由定理 5.1 可知，按照 Web 服务组合记分最大原则进行服务选择，就是根据用户提供的 CoS 需求向量，对每一个抽象服务类选取记分最高的具体 Web 服务的过程。

一般情况下，按照 Web 服务组合记分最大原则所进行的 Web 服务选择，其结果是唯一的；按照该原则进行服务选择的方法具有确定性，不能与其他的服务选择方法，比如 QoS 等结合使用，这也是该方法的局限性所在。

(2) 子服务记分约束准则

子服务记分约束原则要求用户对每一类子服务的记分给出约束的阈值，记分小于阈值的 Web 服务将被淘汰，不允许参与到 Web 服务组合中。

该原则相当于是对候选服务集的一次过滤，经过过滤之后，候选服务集中的具体 Web 服务集合规模会缩小；这样一来，可以对过滤后的候选服务集进行基于 QoS 的服务选择。

可见，按照子服务记分原则进行服务选择的结果是缩减了候选服务集，这既保证了用户对 CoS 的需求，同时又能够提高基于 QoS 的服务选择算法的效率。该原则可以与其他服务选择方法综合使用，以达到最优的服务选择效果。

5.5 基于 CoS 服务选择的例子

5.5.1 基于子服务记分约束原则的选择和 CoS 的更新

在基于 CoS 的 Web 服务选择方面，本文提供一个系统/用户交互界面，用户通过直接选择的方式表达自己的 CoS 需求。

为了方便描述，假设对于每一类抽象服务 CoS 我们都给一个统一的预定义 CoS 模型，即假定他们只有两个内容项：主项 type 的取值可以有 high、low 和 ave，内容项 price 类型为数值类型；Web 服务组合共有 3 个服务类。以下是用户通过系统进行基于 CoS 的选择过程。图 5.4 是对第一类 Web 服务主项的选择：

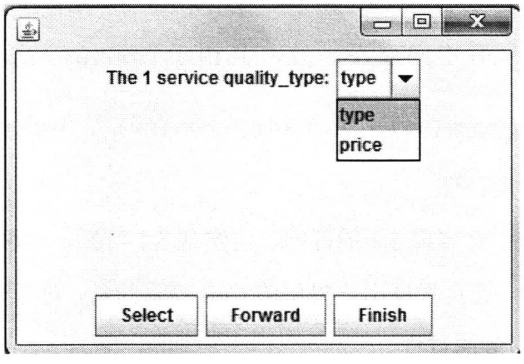

图 5.4 第一个服务主项 type

选择主项 type，点击“Select”进入 type 的值选择界面，如图 5.5 所示：

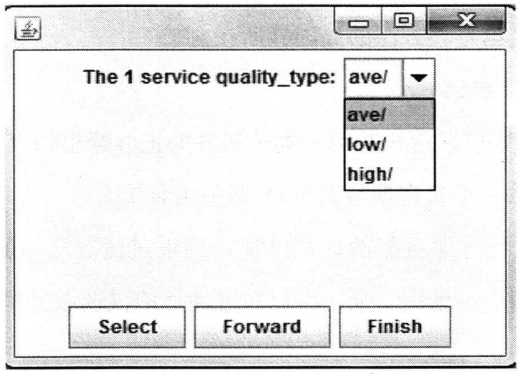

图 5.5 选择主项的值

接下来选择 price 项的值，如图 5.6 所示：

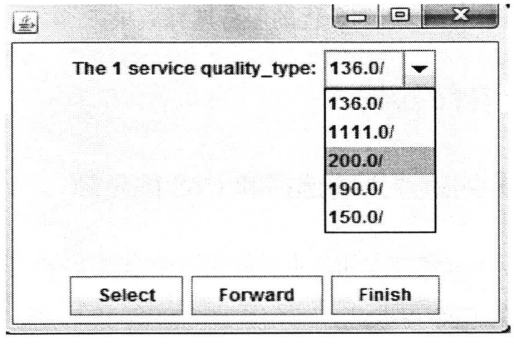

图 5.6 选择 price 项的值

该过程继续下去，直到所有的 Web 服务类选择完毕，如图 5.7 所示：

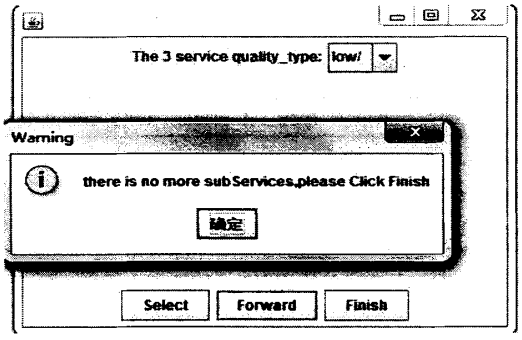

图 5.7 最后一类 Web 服务 CoS 选择

这样就完成了基于用户 CoS 需求的服务选择；在用户使用完服务之后，可以通过系统提供的反馈机制，来更新某一 Web 服务的 CoS，如图 5.8 所示：

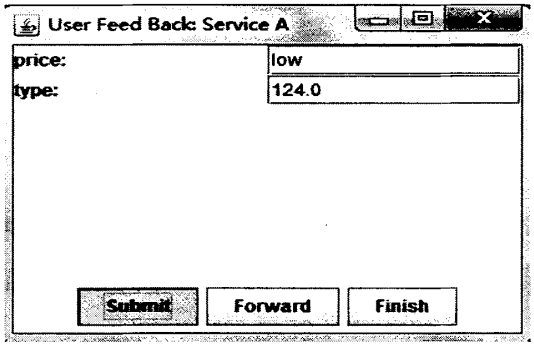

图 5.8 用户对抽象服务类 A 中某 Web 服务的 CoS 反馈

收到该反馈信息后，系统将按照 5.3.2 节中提供的方法进行 CoS 更新。

5.5.2 基于 Web 服务组合记分最大原则的服务选择

假设一个 Web 服务组合 tour 由来自 3 个抽象服务类的 Web 服务构成，它们分别是 hotel、transport 和 dinning；这三个抽象服务类预定义 CoS 模型中 item_list 的形式分别如表 5.2、表 5.3 和表 5.4 所示：

表 5.2 hotel 类对应的 item_list

item_name	room_type	wakeup_serv	air_conditioner	room_price
value1	single-room			
value2	double-room			
value3	standard-room			
value4	deluxe-room			

其中 room_type 是主项，wakeup_serv 和 air_conditioner 为布尔型项，room_price 为数值型项。

表 5.3 transport 类对应的 item_list

item_name	transport_type	t_price	time_consume
value1	taxi		
value2	bus		
value3	bicycle		

其中 transport_type 为主项，t_price 和 time_consume 是数值型项。

表 5.4 dinning 类对应的 item_list

item_name	dinning_style	package_support	free_tableware	cost
value1	western-style			
value2	fast-food			
value3	buffet			
value4	local-delicacy			

其中 dinning_style 是主项，package_support 和 free_tableware 是布尔型项，cost 是数值型项。

在系统中存储的三个抽象服务类对应的具体 Web 服务个数分别为 20 个、24 个和 17 个，它们的 CoS 都已经初始化；如果用户对每一个抽象服务类给出的用户 CoS 需求向量如下：

$$user\_vector_{hotel} = (standard\_room, 1, 1, 160)$$

$$user\_vector_{transport} = (taxi, 45, 50)$$

$$user\_vector_{dinning} = (local\_delicacy, 0, 1, 300)$$

则系统依据公式(5.1)、(5.2)和(5.3)求出每一个抽象服务类中最大记分的 Web 服务对应的编号以及对应的 Web 服务组合最大记分如下所示：

$$id\_array = [5, 21, 14]$$

$$sum\_score(tour)_{max} = 0.69$$

也就是说，系统根据 Web 服务组合最大记分原则，在 hotel 类选择编号为 5 的 Web 服务、在 transport 类选择编号为 21 的 Web 服务、在 dinning 类选择编号为 14 的 Web

服务，并计算以这三个 Web 服务构成的 Web 服务组合记分为 0.69；根据定理 5.1 可知这也是 Web 服务组合的最大记分。

5.6 关于组合选择模型的讨论

在前面我们分别介绍过基于 QoS 的服务选择和基于 CoS 的服务选择，那么这两种服务选择方式能否综合应用，又该用什么样的方式来组合它们呢？本节主要探讨这个问题。

5.6.1 组合选择模型之——层次模型

在 5.4.4 节介绍子服务记分约束准则的时候我们提到，基于该准则进行的服务选择，实质上是相当于对候选服务集进行再过滤，在满足用户 CoS 需求的同时，缩减了候选服务集中具体 Web 服务的数量。如果我们在此基础上，对缩减后的候选服务集再进行基于 QoS 的服务选择，那么该模型就是一个层次模型；用于 Web 服务选择的层次模型如图 5.9 所示。

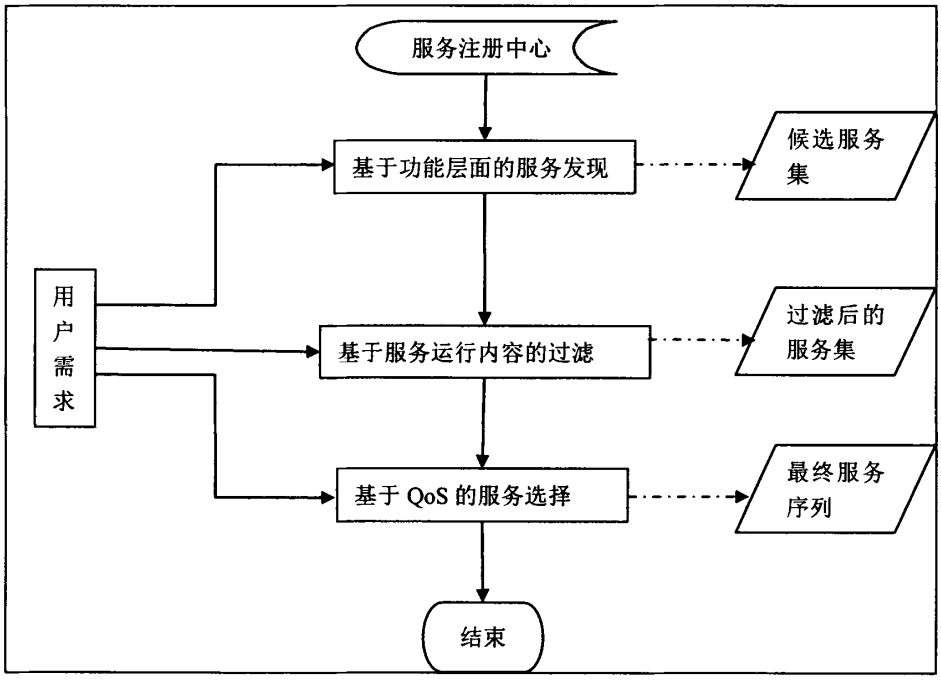

图 5.9 组合选择模型之层次模型图

本文的总体工作实际上是实现了一个图 5.9 所示层次选择模型；以下是关于层次模型优缺点的一些讨论。

(1) 层次模型最大的优点：

是能够综合利用基于 CoS 的服务选择和基于 QoS 的服务选择，通过对候选服务集的两次选择，获得最终用以实现 Web 服务组合的 Web 服务序列；同时，基于 CoS 的服务选择缩减了候选服务集的规模，这有助于基于 QoS 服务选择算法更快速收敛。

(2) 层次模型的缺点：

层次模型的缺点同样也是很明显的。因为是要进行分层次的选择，所以无论先进行哪一个层次的选择，都会在该层次选择中不可避免地忽略掉另一层的一些因素。比如图 5.9 在进行基于服务内容的过滤时，仅仅考虑到某一抽象服务类中 Web 服务的记分是否达到用户指定的阈值，这样做的结果可能会导致服务质量 QoS 非常好的一些 Web 服务被过滤掉；在极端情况下，如果过滤后的候选服务集中剩余 Web 服务的服务质量普遍偏低，那么在进行基于 QoS 的服务选择时，就有可能无法找到用户 QoS 满意度高的 Web 服务；反之亦然。

5.6.2 组合选择模型之——解析模型

鉴于层次选择模型的缺陷，结合第四章中对用户 QoS 满意度定义的启发，讨论解析模型实现的可能性。

首先要对图 4.1 中的层次模型进行扩展，是指包含 CoS 的描述，如图 5.10 所示。

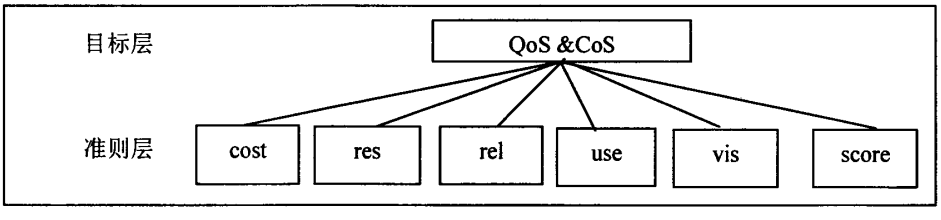

图 5.10 包含 CoS 描述的扩展层次模型

这样一来，我们就相当于把 Web 服务的记分看成是 QoS 的一个指标，这个拥有 6 个指标的“QoS&CoS”新指标可以完全仿效在第四章中关于 QoS 的方法进行处理，最后构造一个类似于 4.6.2 那样的算法进行组合服务选择。以下是关于该模型的一些讨论：

(1) 相对于层次模型，该模型最大的优势就在于同时考虑 QoS 和 CoS 指标，这样就不会造成顾此失彼的尴尬情况；

(2) 但是该模型的实验过程中有几个关键问题有待解决：

(a) 将描述用户 CoS 满意度的 Web 服务记分作为类似 QoS 的指标来处理是否合理；CoS 的跨类不可公度性是否会与此相冲突；

(b) 如何确定图 5.10 中各指标的相对权重：显然我们在第四章中关于组合 QoS 权重的计算方法显然不适合这个场景。

5.7 本章小结

本章首先介绍了基于 CoS 的服务选择，包括 CoS 的定义、性质、初始化、更新和度量方法，并通过例子说明了 CoS 在服务选择过程中的使用。结合本文第四章的工作，本文实际上相当于实现了一个基于 CoS 过滤和基于 QoS 服务选择的组合选择模型，而该模型是一个层次模型；本章的末节深入讨论了组合模型的优缺点，并以探求的姿态提出实现组合选择的解析模型，分析了该模型的优缺点、可能的实现手段以及需要解决的关键问题。

第六章 结论与展望

Web 服务技术作为面向服务体系架构 SOA 和面向服务计算 SOC 的一种实现，其发展日新月异，越来越多的企业选择将其业务流程封装成 Web 服务放在互联网上为用户提供服务；Web 服务组合技术的出现大大提高了现用 Web 服务的可重用性和使用灵活性，降低了开发成本，是 Web 服务的真正潜力所在。随着可用 Web 服务的增加，用户往往会面临幸福的烦恼；如何在保证服务质量和用户偏好的前提下，找到实现其业务逻辑需求的 Web 服务或 Web 服务组合显得尤为重要。

本文的研究工作是基于用户需求的 Web 服务发现与选择。从 Web 服务必须从三个层面上满足用户需求的角度出发，研究工作可以分为三个阶段：

第一阶段：实现基于功能层面的服务发现。这是整个研究过程的基础环节，反映用户最根本的需求；如果服务发现不能够满足用户的功能需求，那么其他工作将没有继续进行的必要性。本文从领域服务所提供的功能是可列的这一事实出发，提出了功能分解的概念和相关原理，在此基础上构造了领域服务功能树；利用领域服务功能树的叶子结点必然是抽象服务类所描述的功能这一性质，确定了领域服务功能树与抽象服务集之间的关联关系；在此基础上，通过对领域服务功能树的遍历和剪枝操作，实现了基于功能层面的 Web 服务发现和候选服务集的构造。候选服务集的构造是后续服务选择的基础，服务选择的过程实际就是对候选服务集进行过滤和再选择的过程。

第二阶段：实现基于 QoS 的 Web 服务选择。服务质量 QoS 是区别功能相似的一组 Web 服务的重要指标，也是对 Web 服务进行选择的重要依据。本文考虑 QoS 的 5 个技术指标，定义了用以描述 Web 服务 QoS 的标准——用户 QoS 满意度；QoS 满意度的计算需要对各 QoS 属性加权，本文从领域客观要求和用户主观要求两方面考虑，提出客观权重和主观权重的定义及计算方法，并以此获得组合 QoS 权重；在此基础上，以用户 QoS 满意度作为适应度函数，构造了基于遗传算法 GA 的 Web 服务选择方法。

第三阶段：实现基于服务内容的服务选择。首先建立描述服务内容的 CoS 模型，介绍了该模型的性质、初始化和更新方法；在此基础上，定义了服务内容的度量方法——记分法，并提出通过记分进行服务选择的两个准则——Web 服务组合最大记分准则和子服务记分约束准则，以及各自的优缺点和应用。

本文的研究工作基于用户的需求，从三个层面实现了 Web 服务发现与选择，取得了一定的成绩，但也还有很多不足之处。研究工作从整体上实现了一个基于层次结构的

组合发现、选择方法，即在实现基于功能层面服务发现的基础上，先利用基于 CoS 的服务选择对候选服务集进行过滤，之后对过滤后的候选服务集进行基于 QoS 的服务选择。这样做的优点很明确，但是缺陷也比较大，因为一个层次的选择往往会损害到另一个层次的质量。

基于此，本文最后提出了服务选择的解析模型，分析了该模型的优势和可能的实现方案，同时也指出该模型的实现需要解决的技术问题，即服务的记分能否被看做是类 QoS 指标，以及如果可以，如何确定该指标和其余 QoS 指标之间的权重关系等。本文作者相信，组合选择模型的优化必将能够最大限度提高用户的服务体验，并愿意为此付出自己的努力。

参考文献

[1] UDDI Version 3.0 OASIS[EB/OL]. http://www.uddi.org，2006-11.

[2] 施有松，侯晓霞. 基于 UDDI 的分布式发现发布策略及其实现[J]. 计算机应用与软件，2004，10(21)：28-29.

[3] 冯百明，刘兴武，李伟. 一种面向消费者的服务发现机制[J]. 计算机研究与发展，2003，40(12)：1787-1790.

[4] Aversano L, Canfora G, Ciampi A. An Algorithm for Web Service Discovery Through Their Composition[A]. Proc of the IEEE IST'1 Conf on web Service[C]. 2004.332-341.

[5] Paolucci M, Soudry J, Srinivasan N. A Broker for OWL-S Web Services[A]. Proc of the AAAI Spring Symp[C].2004.92-99.

[6] Klein M, Bernstein A. Searching for Services on the Semantic Web Using Process Ontologies [A] . Proc of the Int'1 Semantic Web Working Symp[C].2001.159-172.

[7] OWL-S Coalition.OWL-S Release [EB/OL]. http://www.daml.org/services/owl-s/1.1, 2006-11.

[8] 刘传昌，陈俊亮. 目标 Web 服务描述本体和服务发现模型. 计算机工程, 2007,18 期(187-189).

[9] 沈玮韡, 蔡鸿明, 姜丽红. 一种基于语义 Web 服务自动发现的实现. 计算机工程，2006,18 期(211-213)

[10] Stumme G. Using ontologies and formal concept analysis for organizing business knowledge [A]. Becker J, Knackstedt R. Wissensmanagement mit Referenzmodelloen-Konzepte fur die Anwendungssystem-und Orgnisationsgestaltung, Physica[C]. Heidelberg: [s.n.], 2002.163-174.

[11] Li Lei, Horrocks I. A software framework for matching based on semantic web technology [M]. New York, USA: ACM Press, 2003.331-339.

[12] Massimo P, Soudry J, Srinivasan N, et al. A Broker for OWL-S Web Services [A]. In: Proceedings of the AAAI Spring Symposium[C]. Palo Alto, California: [s.n.], 2004.92-99.

[13] 汪清明. 基于领域本体的 Web 服务动态组合模型. 计算机应用. 2009，Vol 29，No 7(1957-1959).

[14] 李曼，王大治，杜小勇，等. 基于领域本体的 Web 服务动态组合[J]. 计算机学报，2005,28(4)：664-650.

[15] 朱益琼，蔡鸿明，姜丽红. 基于领域本体的多层次服务综合匹配[J]. 计算机工程与应用，2007,43(10):128-131.

[16] 王树义，杨吴霖. 基于推理和相似度计算的语义 web 服务匹配策略[J]. 计算机应用与软件，2008，

25(8)：130-132.

[17] 彭晖，史忠植，邱莉榕，常亮．基于本体概念相似度的语义 Web 服务匹配算法[J]．计算机工程，2008，34(15)：51-53.

[18] 邱田，李鹏飞，林品．一个基于语义近似度的 Web 服务匹配算法[J]．电子学报，2009，37(2)：429-432.

[19] 白东伟，刘传昌，陈俊亮．一种增强语义精确度的 Web 服务匹配方法．北京邮电大学学报．2006，29(05)：40-45.

[20] Trastour D, Bartolini C.A Semantic Web Approach to Service Description for Matchmaking of Services.Proc.of the International Semantic Web Working Symposium (SWWS), 2001.

[21] 汤宪飞，蒋昌俊，丁志军等．基于 Petri 网的语义 Web 服务自动组合方法．软件学报，2007,18(12):2991-3000.

[22] 吴健，吴朝晖，李莹等．基于本体论和词汇语义相似度的 web 服务发现[J]．计算机学报，2005，28(4)：595-602.

[23] 艾未华，宋自林，魏磊等．基于领域本体的 Web 服务发现[J]．电子科技大学学报，2007，36(3)：506—509.

[24] 华进，钱雪忠．基于语义的 Web 服务发现模型研究[J]．计算机工程与设计，2008，29(9)：2394-2396.

[25] 刘奎，赵晓静．一种基于本体的 Web 服务发现框架[J]．计算机技术与发展，2008，18(2)，112-114.

[26] 张孝国，黄广君，郭洪涛，曹利红．基于语义的 Web 服务发现技术研究[J]．计算机应用，2008，28(4)：881-883.

[27] 吴红娟，叶飞跃，李霞，彭文涛．基于语义的 Web 服务发现核心技术研究[J]．计算机应用,2006，261 1)：2661-2663.

[28] B．Benatallah, M．Dumas，et a1．Towards Patterns of Web Services Composition．Patterns and Skeletons for Parallel and Distributed Computing．Springer Verlag,UK,2002(11),26-296.

[29] 刘书雷，刘云翔，张帆，等．一种服务聚合中 QoS 全局最优服务动态选择算法［J］．软件学报，2007，18(3) : 646-656.

[30] 莫振华，蔡鸿明，姜丽红．基于遗传算法的多 QoS 约束服务选择［J］．计算机应用与软件，2009，26(3) : 4-6.

[31] 胡焕耀，董渭清，符锐，等．面向 Pareto 最优遗传算法的服务组合方法［J］．西安交通大学学报，2009，43(12) : 50-54.

[32] CLARO D B，ALBERS P，HAO J．Selecting Web services for optimal composition［C］//Proc of

IEEE International Conference on Web Services, Workshop on Semantic and Dynamic Web Processes. 2005.

[33] 孙学胜，曹玖新，刘波，等. 基于多目标粒子群优化的服务选择算法 [J]. 东南大学学报: 自然科学版，2009，39(4) : 684-689.

[34] 夏宏，李增智.粒子群算法求解 Web 服务组合中基于 QoS 的服务选择. Journal of Beijing University of Posts and Telecommunications. Aug.2009, Vol.32, No.4:63-67.

[35] 刘莉平，陈志刚，刘爱心. 基于粒子群算法的 Web 服务组合研究 [J]. 计算机工程，2008，35(5) : 104-106.

[36] ALRIFAI M，RISSE T，DOLOG P，et al. A scalable approach for QoS-based Web service selection [C] //Proc of the 1st International Workshop on Quality-of-Service Concerns in Service Oriented Architectures. Berlin: Springer，2008: 190-199.

[37] 范小芹，蒋昌俊，王俊丽，等. 随机 QoS 感知的可靠 Web 服务组合 [J]. 软件学报，2009，20(3) : 546-556.

[38] 朱红宁，张斌. 基于 SPA 的 Web 服务选取方法 [J]. 计算机科学，2009，36(11) : 32-35.

[39] 叶世阳，魏峻，李磊，等. 支持服务关联的组合服务选择方法研究 [J]. 计算机学报，2008，31(8) : 1383-1397.

[40] ALRIFAI M，SKOUTAS D，RISSE T. Selecting skyline services for QoS-based Web service composition [C] //Proc of the 19th International on World Wide Web. New York: ACM Press，2010: 26-30.

[41] K. Channabasavaiah, K. Holley, E. M. Tuggle, Jr., "Migrating to a Service-Oriented Architecture", IBM developerWorks, December 2003, available at: http://www-106.ibm.com/developerworks/library/ws-migratesoa/.

[42] Ricky E. Sward, Jeff, Boleng. Service-oriented architecture (SOA) concepts and implementations [J]. ACM SIGAda Ada Letters - HILT '2012, 2012.12, 32(3):11-12.

[43] D. Tidwell, Web Services: The Web's next revolution. Available from: http://www.cn-java.com/download/book/wsbasics-a4.pdf.

[44] Microsoft Advertising APIs, What are Web Services? Available from: http://msdn.microsoft.com/en-us/library/bb126205.aspx.

[45] What are Web Services? Available from: http://java.sun.com/blueprints/guidelines/designing_webservices/html/index.html.

[46] W3C Working Group Note. Web Services Architecture Requirements.2004. Available from: http://www.w3.org/TR/wsa-reqs/.

[47] 龚玲等译. Web 服务：原理和技术/(荷) 帕派佐格罗著. 北京：机械工业出版社，2009.12.

[48] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI [J]. IEEE INTERNET COMPUTING, 2002, 6(2):86-93.

[49] 张正明,马炳先,相东明.基于 Petri 网的 Web 服务的创建与描述.系统仿真学报.2011.7,23(s1):19-25.

[50] PengCheng Xiong, YuShun Fan, MengChu Zhou. A Petri Net Approach to Analysis and Composition of Web Services [J]. IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART A-SYSTEMS AND HUMANS Y, 2010, 40(2):376-387.

[51] Wei Tan, Yushun Fan. Data-Driven Service Composition in Enterprise SOA Solutions: A Petri Net Approach [J]. IEEE Transactions on Automation Science and Engineering, 2010.7, 7(3):686-694.

[52] Farhan Hassan Khan, Saba Bashir, M.Younus Javed. QoS Based Dynamic Web Services Composition&Execution [J]. (IJCSIS) International Journal of Computer Science and Information Security, 2010, 27(2):147-152.

[53] 相东明,马炳先,张正明.基于语义的 Petri 网自动共享合成方法研究[J].系统仿真学报，2012, 24(11):2237-2242.

[54] 张正明. Web 服务的 PNML+OWL 描述及应用研究[D]. 山东济南：济南大学，2013.

[55] 张正明,马炳先,相东明.基于 Petri 网的 Web 服务注册方法的研究与实现[J].电信科学，2012.(28)11:86-91.

[56] K. Lcc, etc. QoS for Web Services: Requirements and Possible Approaches, 2003. Available from: http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[57] 崔纪鹏,马炳先,张正明. 面向用户服务质量需求的服务选择方法研究[J]. 计算机技术与发展,2012,22(12):38-42.

[58] 王勇,胡春明,杜宗霞.服务质量感知的网格工作流调度. Journal of Software, Vol.17, No.11, November 2006:2341-2351.

[59] SHUPING RAN. A Model for Web Services Discovery with QoS. ACM2003.

[60] 岳超源. 决策理论与方法[M]. 北京：科学出版社，2003.

[61] 王传玉. 改进 AHP 中判断矩阵一致性的一种新方法. Journal of Anhui Institute of Mechanical & Electrical Engineering. Vol.16, No.4, December 2001:47-50.

[62] 李登峰. 模糊多目标多人决策与对策[M]. 北京：国防工业出版社，2003:22-38.

致 谢

时光荏苒，三年的硕士学习生活转眼将过，回首三年的风风雨雨，内心深处，几多不舍，几多眷恋，几多希冀，几多感激。

我铭记这段经历中的真诚和感动，珍惜一起走过的师长、同学、亲人和朋友。

首先我要感谢我的导师——马炳先副教授。马老师学识渊博，德艺双馨，治学严谨而不失灵活，敬业负责又不乏平易近人。本文的所有工作，包括最后的写作成文，都是在马老师的悉心指导下完成的；在这三年里，无论是科研工作还是日常生活中，马老师的一言一行无不对我有着深刻的影响，从马老师那里，我学会淡定、从容地面对生活，学会严谨、求实地从事科研，学会真诚、谦逊地与人交往。无论多么华丽的辞藻也无法表达我对您的感激之情，我的导师！

我感谢济南大学信息学院的老师们给我的帮助。我忘不了李金屏老师的别出心裁，忘不了乔善平老师的兢兢业业，忘不了张景祥老师的幽默风趣，忘不了陈贞翔老师的严肃认真，忘不了王时春老师的勤勤恳恳。谢谢你们！

我感谢908实验室的同学和225宿舍的室友，你们是我在学校里的亲人，好兄弟、好姐妹！你们使我的生活变得丰富多彩，充满欢声笑语；你们埋头苦读的背影、迎难而上的飒爽英姿激励着我前行。感谢你们！

我感谢我的家人。无论何时何地，无论山高路远，我都感受到你们期许的目光和默默的支持，有你们的地方才是家——我内心深处永远的港湾！

我感谢工信部丁中博士，虽然素昧蒙面，但几次交流让我受益匪浅，衷心感谢！

我感谢济南大学给我提供这么好的学习环境，这么好的人生平台！

最后，我衷心感谢为我评审论文的老师们，你们辛苦了！

2014年4月

附 录

一、在校期间发表的学术论文

[1]面向用户服务质量需求的服务选择方法研究.计算机技术与发展.2012,22(12):38-42.(第一作者)
[2]Research on Service-Content-Based Web Service Selection Method. WISE2013 Workshops 2013, LNCS8182, pp.168-180.(第二作者)
[3]面向组合的 Web 服务间数据关联定位方法研究.(第二作者)
目前该论文已被《计算机科学》正刊录用。

二、在校期间参加的项目

主要参与导师的高校基金课题研究：国家自然科学基金（60903099）。

三、在校期间获奖情况

(1) 2011-2012 年度：研究生入学三等奖学金；

(2) 2012-2013 年度：三等奖学金、三好研究生；

(3) 2013-2014 年度：三等奖学金。

面向用户需求的Web服务发现与选择

作者： 崔纪鹏
学位授予单位： 济南大学