基于 UDDI 的服务查找方法的研究

摘要

随着 Internet 的发展，基于 Internet 的 B2B 电子商务也不断发展。Internet 为各种实体提供了前所未有的机会，使他们利用 Internet 获得了空前的经济回报，体现了 Internet 的巨大价值。然而，目前大多数的电子商务的应用和基于 Web 的商业服务在处理方式上各不相同。如何将这些应用低代价、方便地连结在一起，从而实现大范围的跨企业实体的商务应用系统的对接成为一个十分重要的问题。

Web Service 这一概念的出现，为这个问题提供了一个解决方案。它的主要目标就是在现有的各种异构平台的基础上构筑一个通用的、平台无关的、语言无关的技术层，各种不同的平台之上的应用依靠这个技术层来实施彼此的连接和集成。

但是，在 Web 服务的检索上，已有的规范本身的功能还不很完善，为了提高服务查找的性能，这里提出了一种基于语义信息的服务代理社区查找机制。

首先根据语义信息将整个服务社区空间划分成几个概念域。每个域由一个域代理负责管理，每个域代理内还注册了若干个本地代理，这些域代理服务器以及本地代理服务器共同组成了服务代理社区。

另一方面，为了实现基于语义的查询，要求用户在发布服务的时候要输入一些必要的语义信息，这些语义信息经过预处理，转换成系统内标准形式后进行存储，然后，利用系统采集的其他信息来共同的完成对于服务的语义化描述。

经过系统的语义化处理后，对于用户的服务查询请求，就可以进行优化。同时，利用语义方案，我们可以实现服务调度方式的改进。

在域代理级和本地代理级进行查询操作时，采用语义信息检索方法，实现精确到操作级的服务定位，同时可进行一定程度模糊查询。

关键字：服务查找，电子商务，Web 服务合成，语义信息，UDDI，Ontology，Agents，虚拟企业

Study on Web Service Discovery Based on UDDI

Abstract

With the development of the Internet, Internet-based B2B e-business has developed rapidly. Internet has provided all kinds of entities with enormous opportunities, and brought them unprecedented commercial profits, which has embodied the great value of Internet. Most e-business applications and Web-based commercial services are working in different ways. How to combine these applications with low costs and convenience as well as implement the inter-enterprise commercial applications connection is a big problem.

With the appearance of Web Service, we have found a resolution to this problem. The main purpose of Web Service is to erect a general, platform irrelated, and language irrelated technical level on the existing heterogeneous platform. All applications on different plats can cooperate and integrate by this level.

But the existing rules for Web Service discovery is still not enough. To enhance the performance of the service retrieval, we have presented here a agents based with semantic information analyzing service discovery algorithm.

First, the whole service society is divided into several concept domains according to their semantic information. Each domain is managed by a Domain Broker(DB). And there are still some Local Broker(LB). All these DBs and LBs have established the service society.

At the same time, to realize the semantic-information-based discovery, the users are forced to input some necessary semantic information while they are publishing the services. Then the semantic information will be pretreated and converted to the standard storage form of the system. On the other hand, the system will collect other information to complete the semantic description of the services.

The virtues of this semantic treatment lie in several aspects. The first is that we can optimize the treatment of the user's discovery. Second, we can promote the attemperment of the composite services. With the help of the semantic information analyzing technique, discovery result can be so accurate that we can find an operation that matches the user's demand, and we can get some artificial intelligence to get all

the related results and get rid of those irrelated ones.

Key Words: discovery; e-business; Web Services Composition; Semantic Information; UDDI; Ontology; Agents; Virtual Enterprise;

声明

本人声明所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作过的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本人签名：鲁男

日　　期：2004.1.6

第一章 绪 论

1.1 引言

过去十年的对 IT 产业/COM 的"疯狂投资"的时代已经过去了，那是一个实验的年代。而现在，整个业界跨入了务实的阶段，当今电子商务发展的重心已经完全从过去的.COM 的模式转向到传统企业的电子商务化的进程中来。既然是企业的电子商务化，模式是否崭新是次要的，而是否能为企业带来经济利益则是主要的。在规划企业的电子商务应用的时候，企业管理人员和系统架构师更多的关注是：该电子商务应用是否能为企业带来直接的经济收益；是否有利于削减掉某方面的开支成本；是否能够优化资源使用。这些完全是由企业的商业利益驱动的。在这一轮的电子商务发展中，技术完全是为商业服务的，任何脱离商业需求的"新"技术必然是毫无用武之地的。

在 IT 投资锐减的日子里，系统架构师们小心翼翼、广泛考证，在对企业自身运作机制的务实的仔细调研中，总结出了一些(比较少量的，只有 7 种)当前最有价值实施的电子商务应用，它们是：

(1) 企业门户**(Portal)**：企业门户与一般信息门户有本质的区别，企业门户主要是为企业的重要客户、合作伙伴和自身的员工服务的。它应当具有个性化(这里的个性化并不仅仅是页面)，应当提供一系列的在线服务，客户、合作伙伴和员工们可以使用企业门户获得必要的知识/信息，通过企业门户与企业应用进行交互及事务处理。

(2) 网上连锁商店**(Storefront)**：为了拓展产品和服务的市场，拓广销售渠道以及增加销售额，企业应当建立具有自身品牌标识的网上连锁商店。所谓网上连锁商店，并不是说使用各种语言在各个国家分别建立网上商店，这只是其中的一个形式，更多的方式应当是将企业的网上商店能够加入到各种各样的网上实体中，比如门户网站、行业交易市场(e-Marketplace)、都市引擎等，使企业的销售渠道遍布整个 Web 空间。

(3) 集团内联网**(Intranet)**与知识库**(Knowledge Base)**：集团的全球内联网能够使企业的雇员可以在全球范围内进行有效的交流和协作，充分利用企业的全球资源，以提升整体的生产力。集团的知识库能够为员工的协作

提供丰富有效的工作中所需要的知识，以最大可能地提高员工的单位产出。

(4) 供应链(**Supply Chain**)管理：为提升企业的整体竞争力，企业往往需要保持并提升自身与其供应商的关系，采取流水线形式的采购方式并尽量减少运作成本，而要做到这一点，则必须要创建私有的交易通道和供应链关系的电子商务应用才能达到这一目标。

(5) 客户服务(**Customer Service**)：通过建立这样的面向客户的服务门户或自助式销售网站能够实现跨区销售，提升客户的亲近程度和满意程度，并减少服务成本。

(6) 分销(**Distribution**)管理：建立分销管理应用，能够使企业迅速地拓展分销渠道并挖掘新的市场机会。同时，企业还能裁减培训成本、服务成本和产品分销成本，并减少仓储费用。

(7) 提供 **ASP(Application Service Provider)**服务：通过在 Web 上部署 ASP 服务，企业能够获得新的额外的收入。而提供的 ASP 中的 A(Application)应当是企业核心竞争力的数字化表现，一般情况下，其范围可能就包含了前面提到的 6 种电子商务应用中的 5 种：企业门户、网上连锁商店、供应链管理、客户服务以及分销管理。

为了实施这些电子商务应用，不外乎几种手段：由自己的 IT 部门具体计划并实施；外包给软件公司或解决方案提供商计划并实施。当然，解决方案或实施计划中可能会包含平台软件或专用软件模块的采购。然而，无论自身的 IT 部门还是外包的解决方案提供商，其给出的实施计划都是应用正式运营前的。一旦应用被部署之后，由于商务环境和商务需求的不断改进和不断变化，这些电子商务应用不可避免地需要被修订、被更新，以符合新的电子商务流程，目的使企业获得最大的商业利益并保持竞争力。

在这些应用更新的可能中，下面三个可能是最主要的也是最常发生的：

- 经常会增加新的电子商务应用，这常常会每几个星期或每几个月发生一次；
- 经常会对电子商务的流程进行更改，这常常每周或每几天发生一次；
- 经常应用户的需求而进行更改，这甚至每个小时都会发生，尤其是当需要为每个客户、每个合作伙伴或每个企业员工都定制其首选的电子商务应用的时候。

毫无疑问，e 化的企业必须直面这一问题的挑战，经常的应用更新是当今电子商务应用部署所面临的最大问题，如何提升企业的响应能力，削减响应开支，

提升企业的竞争力，是所有的e化企业必须面对的问题[柴01]。

电子商务需要摆脱独立解决方案的实现模式，需要舍弃复杂系统连接的实现方法。一个有效的电子商务应用绝对不应该是仅仅基于程序员以及那些复杂的代码的。对于电子商务而言，传统的由程序员主导的由里向外的开发模式应当被由用户主导的由外向里的开发模式所取代。冗长的串行的开发循环应当被即时的、快速的应用装配所取代。同时这样的应用应当具备高可定制性。探究其商业本质，应是来自经过时间考验的商业技术概念："即时制造"以及"规模可伸缩"等概念。因此，我们需要做的就是将传统的商业概念延伸到电子商务中去。

基于XML技术的Web服务正是解决这一问题的最佳手段。Web服务的使用将改变目前的开发模式和应用部署的费用规模。各种Web服务分别实现了一定的电子商务功能，通过将各种电子商务的Web服务进行组合和集成以创建动态电子商务应用。Web服务能够统一地封装信息、行为、数据表现以及商务流程，而无需考虑应用所在的环境是使用何种系统和设备。

但是，伴随着Web服务的大规模出现，也同时出现了一个十分严重的问题，那就是对于Web服务的查询问题：

首先，面对于这样数量庞大的服务群，怎样能够从中找到用户需要的服务，就变得十分困难，在这里，用户需要的是能够满足自己的需求的服务，同时，用户还希望被作为查询结果传递回来的服务的质量尽可能的好，最好是其中表现最出色的服务；

其次，对于一些遗产系统内的应用而言，由于他们在设计的时候没有遵循新近出现的Web服务规范，也就是说，他们没有采用现在通用的Soap, WSDL等规范，因此就不能享受Web服务带来的诸多好处。所以，这类服务在查询的时候就出现了问题，因为他们不能够像标准的Web服务那样被使用，从严格的意义上说，他们并不是Web服务。但是，从完成的功能角度讲，他们已经具备了作为Web服务的各种基本的条件，因此，我们面临的第二个问题就是怎样让这些遗留服务同标准的Web服务一样被查询和使用。

1.2 研究Web服务查找的意义

研究Web服务查找的意义在于它可以回答上述的两个问题。

首先，我们可以建立一种有效的机制来帮助用户快速的获得可以满足其需求的服务，在这里，我们通过建立一个Web服务发布中心，把所有出售的服务的

信息都集中到该注册仓当中，同时在该注册仓的基础上建立一种索引，利用这个索引，我们可以用户提供令其满意的结果；

其次，我们可以建立一种可行的机制来帮助那些在遗留系统当中部署的"准Web 服务"获得标准的 Web 服务所具有的各种特性，并以此来满足这些服务被检索到的要求。

由于 UDDI 通常被通俗的语言降格为一个电话目录，因此如此重要的这么一个中央商业注册中心的价值很容易被忽略。事实上，UDDI 注册中心如同 Internet 一样，是为广泛的、分布式的商务媒介环境服务的。

作为一个辅助的说明佐证，让我们来看看 1994 年以前的 Web。在斯坦福大学的两个学生决定公布并不断的更新他们所寻找到的所有网站地址之前，我们是如何使用 Web 的。Yahoo!，令 Internet 用户在网站上寻找并获得信息的方式有了一个根本性的改变。Yahoo! 的网站目录服务对 Internet 的影响仅仅次于 NCSA Mosaic 浏览器对 Internet 的影响，我们至今仍在使用这两项技术和方法。

在 Yahoo!出现之前，发现信息需要花费大量时间，并且依赖于用户对访问的网站的第一印象的简单认知（可能依赖很大的运气成分）。这与今天通过偶然的机会来发现 Web 服务的方式是十分相似的。人们在实现并连接远端系统时必须遵循在线下制定的协议，并且依据技术说明文档使他们的计算机及软件和远端的计算机和软件(Web 服务)进行交互。

UDDI 承诺将解决这一实现上的瓶颈，并将显著地加强基于 Web 的软件和其他软件的连接能力。就如同 Yahoo!显著地提高了 Web 用户查找信息的效率一样，UDDI 的注册中心和语汇集将大大提高基于 Web 的应用和商务流程进行集成的效率。进而，代表企业财富的技术和商业人员将能被解放出来，并将他们的精力集中于解决战略上的问题。当电子商务朝着机器对机器的直接交流的方向发展的时候，对自动化的商务过程的有效发现就成为应用实现中非常重要的实现机制了。

1.3 当前的查询技术—UDDI 的发展状况

当前这个领域内，对于服务发现的研究主要就是 UDDI。UDDI 是一个公共的注册表，旨在以一种结构化的方式来保存有关各公司及其服务的信息。通过 UDDI，人们可以发布和发现有关某个公司及其 Web 服务的信息。这些数据使用标准的分类法进行分类，因此可以按分类来查询信息。最重要的是，UDDI 包

含有关公司服务的技术接口的信息。通过一套基于 SOAP 的 XML API 调用，用户可以在设计时和运行时与 UDDI 进行交互以发现技术数据，从而调用和使用这些服务。通过这种方法，UDDI 可以用作基于 Web 服务的软件系统的基础结构。

为何使用 UDDI？为何需要这种注册表？当我们面对具有数千甚至数百万个 Web 服务的软件系统时，将面临以下的严峻挑战：

1.3.1 如何发现 Web 服务？

针对这个问题，我们首先要回答下面的几个问题：

(1) 如何按照某种合理的方式分类信息？

(2) 对本地化有什么影响？

(3) 对专用技术有什么影响？如何保障发现机制的互操作性？

(4) 当应用依赖于某项 Web 服务时，如何在运行时与该发现机制进行交互？

UDDI 的出现正是为了应对这些挑战。为了解决这些问题，许多公司，其中包括 Microsoft、IBM、Sun、Oracle、Compaq、HP、Intel、SAP 以及三百多家其他公司（请参阅 UDDI: Community（英文）以获得这些公司的完整列表），共同制定了一种基于开放式标准和非专用技术的规范。该规范的 Beta 版于 00 年 12 月发布，正式产品于 01 年 5 月推出。它是一个全球业务注册表，建立在多个运营商节点上，用户可以通过这些节点免费搜索和发布信息。

通过 Web 服务的这种基础结构，能够以一种通用的、与供应商完全无关的方式找到有关 Web 服务的数据，而且数据一致并且可靠；使用可扩展的分类系统和标识，用户可以进行精确的分类查询；运行时， 基于 UDDI ，Web 服务可以被集成到应用程序中去。因而大大繁荣了 Web 服务软件环境。

1.3.2 工作原理

UDDI 数据存放在运营商（即承诺运营一个公共节点的公司）节点上。这种公共节点遵循 UDDI.org 组织管理的规范。目前已经建立了两个遵循 UDDI 规范版本 1 的公共节点：一个属于 Microsoft；另一个属于 IBM。HP 也承诺将建立一个遵循规范版本 2 的节点。数据寄存运营商之间必须能通过安全通道复制数据，从而为整个 UDDI 云团提供数据冗余。将数据发布到一个节点上后，通

过复制，就可以在另一个节点上发现这些数据。目前，每隔 24 小时就进行一次复制；在将来，由于有更多的应用程序要依赖 UDDI 数据，复制的时间间隔还将缩短。

值得一提的是，对于数据寄存运营商实现其节点的方式，不存在一些专用的要求，只要节点遵循 UDDI 规范即可。例如，Microsoft 的节点 http://uddi.microsoft.com/default.aspx（英文）完全用 C# 写成，并运行于 .NET 公共语言运行时环境下。其代码基础充分利用了 .NET 系统类提供的本地 SOAP 支持和序列化。在后端，Microsoft 运营商节点使用 Microsoft SQL Server 00 作为其数据仓库。而 IBM 使用其他技术来运行其节点。但是，这两个节点的行为是相同的，因为它们都遵循相同的一套基于 SOAP 的 XML API 调用。客户端工具可以和这些节点进行无缝的交互操作。因此，UDDI 公共云团是一个最佳方案，它展示了 XML Web 服务模型如何跨异类环境进行工作。

为了了解 UDDI，下一步我们来看看 UDDI 中存储的数据及其存储结构。UDDI 相对来说是轻量的，它被设计为“注册表”，而不是“储备库”。两者之间的差别很微妙，但却很重要。注册表将用户重定向至资源，而储备库则完全是一个信息库。我们以 Microsoft Windows 注册表为例：它包含基本设置和参数，但最终把应用程序引导至资源或二进制代码。基于 Prog ID 搜索 COM 组件时，将引导至一个 Class ID，然后通过 Class ID 再引导至二进制代码本身所在的位置。

UDDI 的行为与之类似：与 Windows 注册表一样，它依靠全局唯一标识符 (GUID) 来搜索并定位资源。UDDI 查询最终指向一个接口（.WSDL、.XSD 和 .DTD 文件等等），或指向其他服务器上的实现（例如 .ASMX 或 .ASP 文件）。UDDI 因此可以回答以下问题：

- “已经发布了哪些基于 WSDL 并是为指定行业建立的 Web 服务接口？”
- “哪些公司已经为其中一个接口写好了实现？”
- “目前提供的 Web 服务（以某种方式分类）有哪些？”
- “某个公司提供了哪些 Web 服务？”
- “如果要使用某个公司的 Web 服务，需要与谁联系？”
- “某个 Web 服务的实现细节是什么？”

1.4 本文解决的主要问题及组织结构

本文以东北大学申请的国家高技术研究发展计划（863 计划）课题——基于 ASP 模式的支持中小企业动态联盟的使能服务系统的研究——为背景，研究探讨了虚拟企业中基于私有 UDDI 中心集合的 Web 服务的发现算法与机制。

本文主要研究了虚拟企业内部的服务和外部的第三方服务在虚拟企业系统中的查询，同时为了解释说明查询的实现机制，还包含了注册、集成、调用和代理等方面的内容，并且利用了当今世界上先进的规范标准来实现。

本文的组织结构如下：第二章介绍了 Web 服务的查询的研究及发展状况，分析了 Web 查询的特点，以及当前该领域的状况，并简单介绍了目前处于主流地位的查询机制 UDDI，并分析其不足，给出本文所涉及的方案的优势；第三章介绍了为实现本方案的实现的相关技术，比较当今流行的系统基础平台，介绍了本系统应用的新技术和规范；第四章介绍整个系统的体系构架和设计；第五章给出了一个中间代理的设计，这个代理负责将遗产系统中的 Web 应用包装成标准的 Web 服务，同时，还负责收集大量的服务的调用信息，作为对服务进行综合评价的一个重要的依据；第六章，基于服务社区的 Web 服务的语义化处理；第七章总结了本方案的优点及有待改进的地方，同时对 Web 服务的查询解决方案的发展方向做了分析和展望。

第二章 Web 服务查询的研究与发展

2.1 Web 服务查询的简介

所谓的"Web 服务"，它是指由企业发布的完成其特别商务需求的在线应用服务，其它公司或应用软件能够通过 Internet 来访问并使用这项在线服务。

Web 服务将逐渐成为电子商务应用构建的基础体系架构。例如，一个公司通过另一个公司所提供的服务，直接在 Internet 上发送一份订货单。又例如，某一个服务用于计算某一尺寸或是重量的货物通过某种方法(如海运)运输一定距离需要花费多少费用。

大家可能认为，对于现有的商务合作伙伴，我已经知道他有一个已知的电子商务入口，那还有什么需要发现的呢？当然这是建立在一个默认的假定情况"信息都是已知的"这样一个基础上的。而且无论在哪种情况下，当需要找出哪些商业伙伴可以提供什么样的服务时，快速地发现并找到答案仍然将会变得十会困难。其中一个可选的方法是使用电话和每个合作伙伴进行联系，然后找出合适的对象。对于一个提供 Web 服务的商业实体来说，需要配备具备相当技术能力的专业人员去满足这样随机的服务发现需求显然是不合适的。

另一个解决该问题的办法是在公司的每个网站上放置一个 Web 服务的描述文件。这样，至少那些依靠已经注册的 URL 来工作的网络爬虫程序能够发现并为它们建立索引。可是这种通过"robots.txt"来定位 Web 服务的方法完全依赖爬虫程序的能力。这种分布式的机制是可扩展的，但它缺少一种机制来保证服务描述格式的一致性，也不能便捷的跟踪不断发生的变化。

这里提供了一种基于分布式的商业注册中心的方法，该商业注册中心维护了作为系统内用户（也就是联盟内用户）的一个企业以及该企业提供的 Web 服务的全球目录，而且其中的信息描述格式是基于通用的 XML 格式的。

2.2 Web 服务发布管理中心应具有的职责与功能

首先，说明一下 Web 服务发布管理中心应具有的职责与功能：

(1) 依据用户的输入条件查询特定的服务（或服务集合）；

(2) 查找实现了基于公共抽象接口定义的服务；

(3) 依据一致的划分模式或划分体系来查找提供特定类型服务的服务提供商；

(4) 确定一个具体的服务支持的安全与传输协议；

(5) 保存一个 Web 服务的技术信息以及更新信息；

2.3 基于 UDDI 的 Web 服务查询与传统的搜索引擎的区别

存在于现今 Internet 上的具有代表性的商业目录只是罗列了企业的基本信息，例如名称，地址，联系方式，一些产品和服务信息，还有必须的产业分类法。他们没有列出企业所提供的 Web 服务的编程入口描述。

UDDI 是被设计成为这些数据的一个集中的共享资源，它可以为现有的目录和搜索引擎服务。这也预示着，大部分的客户和商业实体，将仍然可以继续使用现有的搜索引擎和企业目录，作为他们首选的方法来浏览在 UDDI 商业注册中心中注册的企业信息。UDDI 商业注册中心是一个基础模块，它是为企业发布信息和服务描述、发现企业提供的信息和服务以及完成企业之间的商务流程的互操作等服务的。

2.3.1 什么是搜索引擎

搜索引擎（Search Engines）是一个对互联网上的信息资源进行搜集整理，然后供你查询的系统，它包括信息搜集、信息整理和用户查询三部分。

搜索引擎是一个为你提供信息“检索”服务的网站，它使用某些程序把因特网上的所有信息归类以帮助人们在茫茫网海中搜寻到所需要的信息。

早期的搜索引擎是把因特网中的资源服务器的地址收集起来，由其提供的资源的类型不同而分成不同的目录，再一层层地进行分类。人们要找自己想要的信息可按他们的分类一层层进入，就能最后到达目的地，找到自己想要的信息。这其实是最原始的方式，只适用于因特网信息并不多的时候。随着因特网信息按几何式增长，出现了真正意义上的搜索引擎，这些搜索引擎知道网站上每一页的开始，随后搜索因特网上的所有超级链接，把代表超级链接的所有词汇放入一个数据库。这就是现在搜索引擎的原型。

随着 Yahoo!的出现，搜索引擎的发展也进入了黄金时代，相比以前其性能

更加优越。现在的搜索引擎已经不只是单纯的搜索网页的信息了，它们已经变得更加综合化，完美化了。以搜索引擎权威 Yahoo!为例，从 1995 年 3 月由美籍华裔杨致远等人创办 Yahoo!开始，到现在，他们从一个单一的搜索引擎发展到现在有电子商务、新闻信息服务、个人免费电子信箱服务等多种网络服务，充分说明了搜索引擎的发展从单一到综合的过程。

然而由于搜索引擎的工作方式和因特网的快速发展，使其搜索的结果让人越来越不满意。例如，搜索"电脑"这个词汇，就可能有数百万页的结果。这是由于搜索引擎通过对网站的相关性来优化搜索结果，这种相关性又是由关键字在网站的位置、网站的名称、 标签等公式来决定的。这就是使搜索引擎搜索结果多而杂的原因。而搜索引擎中的数据库因为因特网的发展变化也必然包含了死链接。

怎样才能使搜索引擎精确地为人们提供相关的信息应该是它以后发展的方向，而不是只求综合服务。

2.3.2 什么是统一描述、发现和集成协议 UDDI?

统一描述、发现和集成----UDDI(Universal Description, Discovery, and Integration) 是一个全球范围的可使商户在因特网上列出其自身的基于 XML 的商务注册体。 它的最终目标是能使企业在万维网上彼此找到并让其电子商务系统能互操作， 从而达到在线交易的流水化。UDDI 经常被比做电话簿的白页、黄页和绿页。 这一项目允许商户把其业务按名字、产品、位置或其提供的网上服务进行组织。

Microsoft, IBM 和 Ariba 领导着 UDDI，该项目成员目前包括 130 家公司，包括世界上的一些大名鼎鼎的公司。Compaq, 美国快递, SAP AG, 和福特汽车公司 都提交过计划给 UDDI，像 HP 公司它们自己就有基于 XML 的称为 e-speak 的目录体系， 现在已被集成进 UDDI 里。

该组织不把他们自己称为标准实体，但他们确实提供了网上服务集成的框架。 UDDI 规范利用 W3C 和 IETF 的诸于 XML, HTTP, DNS 等协议，它们还采用了建议中的 SOAP 协议的早期版本中对交叉平台编程的消息指南。

统一描述、发现和集成协议(UDDI)是一套基于 Web 的、分布式的、为 Web 服务提供的信息注册中心的实现标准规范， 同时也包含一组使企业能将自身提供的 Web 服务注册以使得别的企业能够发现的访问协议的实现标准。

Web 服务是下一代的 WWW，它允许在 Web 站点上放置可编程的元素，

使得能进行基于 Web 的跨企业的分布式计算和处理。 UDDI 商业注册中心的创建目的就是为促进企业的 Web 服务的发展及为企业发现适当的 Web 服务。

2.3.3 二者的区别

二者的区别主要表现在对信息的处理上，UDDI 规范定义了发布和发现 Web 服务的方法，为企业和他们所提供的 Web 服务进行注册。服务以及实体的信息集中在数据中心里，可以进行全面而且实时的比较、搜索、替换、更新等操作。而且 Web 服务是以通用的XML 格式进行描述的，XML描述是具有语义信息的，所以可以进行高级的查询。

而搜索引擎（Search Engines）是一个对互联网上的信息资源进行搜集整理，然后供用户查询的系统，它包括信息搜集、信息整理和用户查询三部分。他在发现信息的过程中主要依靠网络爬虫的检索，形成的结果存在数量庞大，准确率低等问题。

2.4 现有的查询解决方案

为了解决第一个问题，一系列解决方案出台了，首先要说 UDDI[BE02][ABC02]规范，UDDI 规范试图通过建立一系列的规范，来实现对网络服务的统一描述、发现和集成。目前主要提供基于如下三种模式的查询[BED02]：

(1) 浏览模式。类似于 Web 浏览的方式的查询方法，用户首先查询一个他熟悉的企业，通过输入该企业的名字中的一两个字来实现，然后，用户在浏览该企业的 Web 服务列表，察看是否存在用户需要的服务，此过程一直持续到找到用户需要的信息为止；

(2) 钻探模式。建立在用户已经获得了注册于 UDDI 注册中心中的一个企业或服务的 ID 的基础上，用户利用已知的 ID，在 UDDI 注册中心对所需信息进行详细地查询；

(3) 激活模式。在用户已经获得了远程服务的详细信息，并建立了与远程服务的联系后，在实际调用时发现远程服务不可用，此时，用户的调用程序自动利用缓存的远程服务的信息来提出查询请求，向 UDDI 中心查询

关于该服务的更新情况（主要指 *get_bindingDetails* 一类的操作），并根据新获得的绑定信息来重新建立与远端程序的连接。

三种模式中只有第一种查询方式可以提供给用户根据主观意图进行查询的能力，但是这种能力十分有限。首先，该查询是根据用户输入的关键字进行查询的，它隐含要求用户对自己所查询的信息领域十分熟悉；其次，通过该查询方式查询到的信息具有片面性，不完全性，因为不同的人对于同一概念在理解上总是或多或少的存在偏差，此外，它还要求用户作大量的工作才能够获得想要的数据。

第二种查询方式更像是第一种方式的一个中间步骤，它是在用户已经获得了关于一个企业、服务的信息的 ID 的前提下执行的进一步的查找。

而第三种查询方式则充分体现了在 Web 环境下利用服务间的松散的耦合性进行动态的替换已有服务（尤其在此服务失效的情况下）的思想，但是，其自动化程度还不够。例如，存在一些实现同一接口类型的上下文无关的服务，虽然可能由不同的提供商提供，但是实现的功能相同，理论上可以彼此替换，但这种查询替换并没有在 UDDI 中体现出来。

所以说，在 UDDI 规范身并没有提供一个十分有效的查找方法，其中提供的查找能力还不能够满足用户的需要，于是有人提出利用基于本体论的语义信息来描述服务，并在这种含有语义信息的描述的基础上来查询服务，这种方案的代表就是 DAML[PKR02]。DAML 虽然提供了一些比较新的思路，可是它本身还很不完善，而且在改进体系结构上基本没有什么作为，同时也缺乏有实力的组织的支持，所以，当前，UDDI 在这一领域里还占据着绝对的领导地位。

另一方面，一些学者试图通过改进系统的体系结构来实现对于整个查询的改进，这方面的一个代表性成果就是 Sangum 系统[JH01]，这个系统利用多节点对服务信息进行管理，同时加入人工智能的解决方案，但是它并没有成为一个真正成熟的技术。同时，UDDI 自己也认识到自己在体系结构方面的不足，在其发布的 UDDI 规范 3.0 中，着重引入了对于多注册节点的支持[JM02] 。

2.5 新的解决方案

本文基于 UDDI 规范，提出一种基于领域划分的代理社区结构，并在此结构上添加语义信息，以增强 UDDI Web 服务注册管理器的功能。

为了达到上述的几个要求，本系统的设计侧重一下两个方面：

(1) 从系统的体系结构上进行改变，通过建立一个合理的系统体系，改变服

务的查询流程，来达到提高查询的效率与性能的目的；

(2) 从服务信息的记录的加角度出发，通过添加必要的语义信息，增强查询的准确率与查询结果的相关性。

在实现上，需要将二者紧密地结合在一起，一方面改变结构会影响查询的动作。例如，通过分域处理。

2.6 服务查询的现存的解决方案

本文基于 UDDI 规范，提出一种基于领域划分的代理社区结构，并在此结构上添加语义信息，以增强 UDDI Web 服务注册管理器的功能。

首先，说明一下 Web 服务发布管理中心应具有的职责与功能：

(1) 依据用户的输入条件查询特定的服务（或服务集合）；

(2) 查找实现了基于公共抽象接口定义的服务；

(3) 依据一致的划分模式或划分体系来查找提供特定类型服务的服务提供商；

(4) 确定一个具体的服务支持的安全与传输协议；

(5) 保存一个 Web 服务的技术信息以及更新信息；

为了达到上述的几个要求，本系统的设计侧重一下两个方面：

(1) 从系统的框架结构入手，建立一个有益于查询的系统体系；

(2) 从服务信息的内容入手，创造一个有益于查询的描述环境。

在实现上，二者是密不可分的，首先，通过对服务的描述信息进行基于语义的分析，将整个问题空间划分成若干个语义域，每个域内只记录该域所属的服务的信息，例如，股票报价服务，可以划分在金融领域内，这样在机械制造领域内就没有这类服务，采取这种方式，在查询管理上就十分方便，快捷，而且准确；其次，从服务提供者发布服务描述信息开始，到服务使用者发出查询请求为止，整个过程中，对所有的信息都进行基于语义的处理，这可以保证用户获得他们真正想要的服务，而不是一些垃圾信息；在每个域内，还设立了若干个代理器（本地代理 **LB**），利用这些代理器，可以为用户提供更周到完善的服务。对于拥有多个区域的系统，各服务中心必须保证区域之间的数据同步，即当某个区域内数据发生变化时，局部服务控制中心的复制服务保证准确地把信息复制到其他区域，保证全局数据的一致性。

第三章 实现的相关技术

3.1 国内外现有的技术基础

在当今的时代，技术是商业创造价值的基础，但是，这里所说的技术不是一个独立的概念。在现在的社会里，每种新技术的采用都来不开大量的相关领域内的技术的支持，例如，当前的无线通信技术已经逐渐地进入到 3G 时代，但是，在这个转变的过程里，不仅仅需要解决无线通信领域内的技术问题，还需要解决大量的诸如用户数据管理、网络服务的提供等问题。相似的，Web 服务的引入同样需要大量的周边技术。这些周边技术就包含了 Web 服务所必须的一些基础性的支持：

（1）可靠的网络硬件环境和 IT 技术保证，如：技术实力雄厚的 ASP 网站支持，Internet 宽带网的支持，基于 Web 技术的应用软件支持，Internet 安全技术支持，低廉的网络通讯费用 等等， 为提供 ASP 服务提供了坚定的基础。

（2）成熟、开放的 Web 平台构架软件提供商的技术支持，如：Microsoft 的.net、SUN 的 J2EE 和 IBM 的 Web Sphere 平台等。

（3）Internet 通信的标准化（如浏览器的普及）和服务的社会化为企业方便地使用 ASP 服务商的服务提供了便利条件。企业只需要几部微机和安装一个浏览器软件，就可以使用各种软件。软件和系统的维护与系统的使用完全分割开来，客户无需任何管理和维护工作。这就从技术上保证了 ASP 模式的流行和成功。

（4）随着先进的软件技术和国际工业标准的出台，为基于 Web 服务的动态企业联盟提供了新的生命力。如 XML、SOAP、UDDI、WSDL 等工业标准使动态企业联盟及其 e_business 的实现提供了可行性：

专家预测，随着软件的发展和工业标准的推出，下一代的 e_business 将是开放的动态联盟企业的主体。Web 服务 概念是从面向应用到面向服务结构（SOA）的 Web 服务的进一步发展，SOA 结构为实现开放的、平台中立的、异构系统交互的 Web 服务提供了坚实的基础。

3.2 开发环境的选择

Microsoft .NET 是 Microsoft 的 XML Web 服务平台。.NET 包含了建立和运行基于 XML 的软件所需要的全部部件。其框架结构见图 3.1。

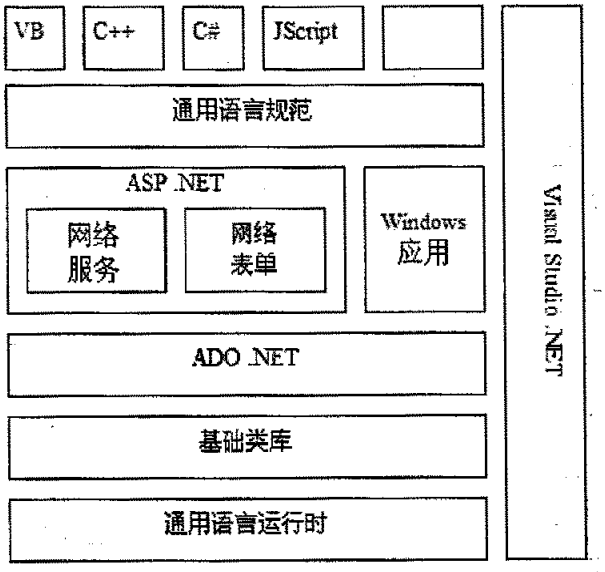

图 3.1 Microsoft.NET Framework 体系结构

Figure3.1 the Architecture of Microsoft.NET Framework

Microsoft .NET 解决了下面这些当今软件开发中的一些核心问题：

- 互操作性(Interoperability)、集成性(Integration)和应用程序的可扩展性(extensibility)太难实现而且代价很高。Microsoft .NET 依靠 XML(一个由 World Wide Web Consortium(W3C)管理的开放标准)消除了数据共享和软件集成的障碍。
- 无数具有相当竞争力的私有软件技术使得软件的集成变得非常复杂。而 Microsoft .NET 建立在一个开放的标准上，它包含了所有编程语言。
- 当终端用户使用软件时，他们总觉得不够简便。有时甚至感到很沮丧，因为他们无法在程序之间方便地共享数据或是无法对能访问的数据进行操作。XML 使数据交换变得容易了，并且.NET 软件可以使得用户只要一得到数据就能对它们进行操作。

- 终端用户们在使用 Web 的时候，无法对自己的个人信息和数据进行控制，这导致了个人隐私和安全泄漏问题。而 Microsoft .NET 提供了一套服务，使用户可以管理他们的个人信息，并且控制对这些信息的访问。
- .COM 公司和 Web 站点开发者们很难为用户们提供足够的有价值的数据，至少有一部分原因是由于他们的应用程序和服务无法很好地和其他程序和服务合作，只是一个不和外界连接的信息孤岛。而 Microsoft .NET 的设计宗旨就是为了使来自于多个站点和公司的数据或服务能够整合起来。

如同 MS-DOS 和 Windows 一样，.NET 将大大改变我们的计算领域。MS-DOS 使得个人电脑在商业和家庭中广为接受；Windows 增强了用户的图形界面，使其成为首选的与软件交互方式，最终使得图形界面成为个人电脑的主流。而.NET 则要把 XML Web 服务变成日后的主流计算模式。

3.3 新技术的应用

新的技术出现带来新的应用方式，网络服务的体系结构[IWS01]使构建网络服务的应用更为方便，只要遵循这个标准就能够架构出需要的服务体系，整个体系如图 3.2 所示。

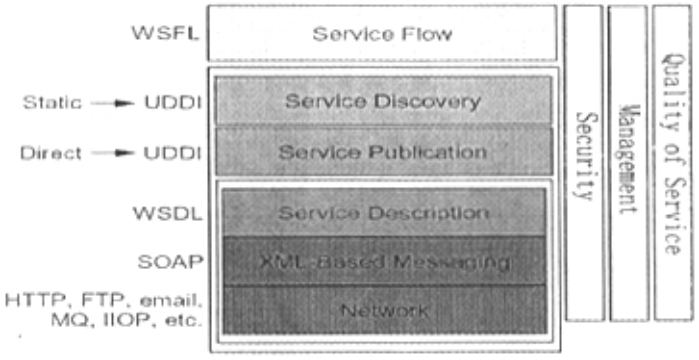

图 3.2 网络服务的体系结构

Figure 3.2 the Architecture of Web Service

3.3.1 网络服务（Web 服务）

网络服务（Web 服务）是基于网络的分布式应用程序的基本构造模块，而这

些程序是以平台、对象模板和多语言方式构建的。

网络服务是建立在象 HTTP 和 XML 之类的开放的 Internet 标准之上的，并且由此形成了可编程网络理念的基础。

通常说来，网络服务只是一个作为服务发行的简单应用程序。换句话说，它是可通过 URL 定位的自动将信息返回到需要它的客户端那里的一种资源。网络服务一个重要的特点是客户不需要知道一种服务是怎样实现的。

同组件一样，网络服务提供"黑匣子"函数，它可以被多次用而不用关心此服务是怎样实现的。网络服务还提供被称为契约的精确定义的接口，此接口描绘了所提供的服务。开发人员可以将远程服务、本地服务和定制代码组合在一起集成到应用程序中。例如，某公司可以使用如下服务组建一个在线商店：微软护照（Passport）服务用来验证用户身份、第三方个人化服务用来使网页匹配每一个用户的参数、信用卡处理服务、销售税服务、对每个运输公司的包裹跟踪服务，链接公司内部库存管理程序的内部目录服务以及少量定制代码，以使他们的商店能脱颖而出。

然而，网络服务与现在的组件技术并不相同，它不使用需要在服务器和客户机有明确的、同类型基本构架的具体对象模型协议，例如 DCOM、RMI 或 IIOP。尽管与具体组件技术紧密结合的实现在一个受控的环境中能很好地被接受，但它们在网络环境中变得不切实际。因为一个集成商业程序的参与者会发生变化，随着时间的推移，技术也在变化，所以在所有参与者间确保一个单一的、统一的体系架构就变得十分困难。网络服务采取了另外一种途径，它使用普便存在的网络协议和数据格式进行通信，如 HTTP 和 XML。支持这些网络标准的任何系统都支持网络服务。

而且，网络服务契约描述的是以术语报文形式提供的服务，这些服务是由网络服务生成和接受的，而并不描述服务是如何实现的。通过把重点放在报文上，网络服务模板对语言、平台和对象模板变得完全透明。这样，用任何一套编程语言、对象模型和平台的完全特性集，都可实现网络服务。网络服务可以在任何平台上，被任何应用程序所使用。只要用于解释服务容量、报文序列和所期望协议的契约得到认同，那么所实现的网络服务及网络服务用户就可相互不同，而不会影响会话另一端的应用程序。

网络服务模板对最小体系架构的要求很低，目的是确保网络服务在使用任何技术和编程语言的平台上实现和访问。对网络服务互用性的解决可以只依靠网络标准。然而，为了使应用程序更容易使用网络服务，简单地通过标准网络协议访问网络服务是不够的。当网络服务和网络服务使用者依靠标准的方式（如 XML）

表示数据和命令、表示网络服务契约、算出网络服务所提供的容量时，网络服务才会更加容易使用。

Web 服务是下一代的 WWW，它允许在 Web 站点上放置可编程的元素，能进行基于 Web 的分布式计算和处理。Web 服务的发展非常迅速，这个新规范（SOAP、WSDL 和 UDDI）的构建模块仅仅才出现了几个月，就已经对设计、开发和部署基于 Web 的应用产生了巨大的影响，软件产业的巨头和 Internet 时代的软件新贵们已经开始了在这个领域新一轮的竞争。

3.3.1.1 Web 服务的特点

Web 服务是封装成单个实体并发布到网络上供其他程序使用的功能集合。Web 服务是用于创建开放分布式系统的构件，可以使公司和个人迅速且廉价地向全世界提供其数据服务。

Web 服务是下一代分布式系统的核心，它具有如下特点：

互操作性：任何的 Web 服务都可以与其他 Web 服务进行交互。由于有了 SOAP（Simple Object Access Protocol）这个所有主要供应商都支持的新标准协议，因而避免了在 CORBA、DCOM 和其他协议之间转换的麻烦。还因为可以使用任何语言来编写 Web 服务，因此开发者无需更改其开发环境，就可生产和使用 Web 服务。

普遍性：Web 服务使用 HTTP 和 XML 进行通信。因此，任何支持这些技术的设备都可以拥有和访问 Web 服务。

易于使用：Web 服务背后的概念易于理解，并且有来自 IBM 和微软这样的供应商的免费工具箱能够让开发者快速的创建和部署 Web 服务。此外，其中的某些工具箱还可以让已有的 COM 组件和 JavaBean 方便地成为 Web 服务。

行业支持：所有主要的供应商都支持 SOAP 和周边 Web 服务技术。例如，微软的.Net 平台就基于 Web 服务，因此用 Visual Basic 编写的组件很容易作为 Web 服务部署，并可以被 IBM VisualAge 编写的 Web 服务使用。

3.3.1.2 Web 服务的体系结构

Web 服务是独立的、模块化的应用，能够通过网络，特别是 WWW 来描述、发布、定位以及调用。Web 服务的体系结构描述了三个角色（服务提供者、服务请求者、服务代理者）以及三个操作（发布、查找、绑定），如图 3.3 所示[蒋 01]。

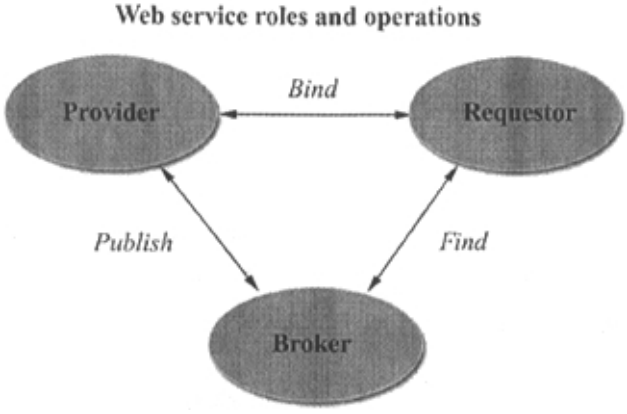

图 3.3 网络服务中的角色和操作

Figure3.3 the Roles and Operation of Web Service

服务提供者通过在服务代理者那里注册来配置和发布服务，服务请求者通过查找服务代理者那里的被发布服务的登记记录来找到服务，服务请求者绑定服务提供者并使用可用的服务。

在 Web 服务的世界里，三个操作都包含三个受到称赞却又截然不同的技术。发布服务使用 UDDI(统一描述、发现和集成)，查找服务使用 UDDI 和 WSDL(Web 服务描述语言)的组合，绑定服务使用 WSDL 和 SOAP。在三个操作中，绑定操作是最重要的，它包含了服务的实际使用，这也是容易发生互操作性问题的地方。正是由于服务提供者和服务请求者对 SOAP 规范的全力支持才解决了这些问题，并实现了无缝互操作性。

当开发人员开发新的应用时，可以通过 UDDI Operator 或 UDDI Search Engine 的 Web 界面在 UDDI Registry 上找到需要的 Web 服务；然后在 UDDI Registry 内，或通过 UDDI Registry 中的连接找到该 Web 服务的调用规范，该调用规范一般使用 WSDL 描述。开发人员可以使用开发工具或通过手动方式调用该规范，然后在自己的应用中加上该调用规范定义的 Web 服务调用。这样开发出的应用就可以通过 SOAP 来调用指定的 Web 服务了。

而对于具有自动集成相关应用的服务(Service)或应用(Application)，用户应用通过 SOAP 协议访问 UDDI Operator 或 UDDI Registry，找到需要的 Web 服务，UDDI Operator 和 UDDI Registry 会通过 SOAP 协议响应 Web 服务的调用规范和调用规范的链接，应用程序得到使用 WSDL 描述的服务调用规范文本，通过解析该描述文本，自动生成本地调用接口绑定，并将所需的调用参数适当绑定并完

成调用。

3.3.2 XML

可扩展标记语言，缩写为 XML[BPM02]，描述了一类称为 XML 文件的数据对象，同时也部分地描述了处理这些数据对象的计算机程序的动作。XML 是 SGML(标准通用标记语言[ISO 8879])针对特定应用领域的一个子集，或者说是 SGML 的一种受限形式。根据定义，XML 文件是合乎规范的 SGML 文件。

XML 文件由称为实体的存储单元组成，实体可以包含已析数据或未析数据。已析数据由字符组成，其中一些字符组成字符数据，另一些字符组成标记。标记中包含了对文件存储格式(storage layout)和逻辑结构的描述。XML 提供了一种机制用于约束存储格式和逻辑结构。

称为 XML 处理器的软件模块用于读取 XML 文件，存取其中的内容和结构。XML 处理器被设想为是为另一个称为应用的模块作处理。本规范从 XML 处理器应如何读取 XML 数据以及应向应用提供哪些信息的这两个方面，描述了要求 XML 处理器做出的动作。

3.3.3 简单对象访问协议 SOAP

SOAP[GHM03](Simple Object Access Protocol)是一个独立于传输协议的消息协议，每个 SOAP 消息是一个 XML 文档。尽管使用上常常是以请求-应答方式出现，SOAP 使用的仍然是单向消息。

SOAP 是在 XML 基础上定义的，完全继承了 XML 的开放性和描述可扩展性。SOAP 使用现有基于 TCP/IP 的应用层协议 HTTP、SMTP、POP3 等，可以获得与现有通信技术最大程度地兼容。SOAP 的消息路径机制和可扩充的 Header 和 Body 机制又为分布式计算提供了很好的支持。

SOAP 规范定义了 XML 消息的格式，但没有定义消息的内容和实际传输方式。

SOAP 为在一个松散的、分布的环境中使用 XML 对等地交换结构化的和类型化的信息提供了一个简单的轻量级机制。SOAP 本身并不定义任何应用语义，如编程模型或特定语义实现，它只是定义了一种简单的机制，通过一个模块化的包装模型和对模块中特定格式编码的数据重编码机制来表示应用语义。SOAP 的这项能力使得它可被很多类型的系统用于从消息系统到 RPC(Remote Procedure

Call)的延伸。

由于 SOAP 的主要设计目标是简明性和可扩展性，这就意味着有一些传统消息系统或分布式对象系统中的特性将不包含在 SOAP 的核心规范中。这些特性包括：分布式垃圾收集 (Distributed garbage collection)、成批消息传输/处理(Boxcarring or batching of messages)、对象引用 (Objects-by-reference)以及对象激活(Activation)。

3.3.4 Web 服务描述语言 WSDL

随着通信协议和消息格式在 Web 中的标准化，以某种格式化的方法描述通信变得越来越重要，其实现的可能性也越来越大。用 WSDL[CCM01]定义的一套 XML 语法描述的网络服务方式满足了这种需求。WSDL 把网络服务定义成一个能交换消息的通信端点集。WSDL 服务为分布式系统提供了帮助文档，同时该服务也可作为自动实现应用间通信的解决方案。

一个 WSDL 文档将服务定义为一个网络端点的集合，或者端口的集合。在 WSDL 里，端点及消息的抽象定义与它们具体的网络实现和数据格式绑定是分离的。这样就可以重用这些抽象定义：消息，需要交换的数据的抽象描述;端口类型，操作的抽象集合。针对一个特定端口类型的具体协议和数据格式规范构成一个可重用的绑定。一个端口定义成网络地址和可重用的绑定的联接，端口的集合定义为服务。

几家大的企业合力建立了 SOAP 标准。Web 服务描述语言(WSDL)向这种 Web 服务的提供商和用户推出了方便的协调工作的方法，使我们能更容易的获得 SOAP 的种种好处。对 Web 服务来说，我们必须先制定出指定接口的标准格式。SOAP 消息确实带有某些类型信息，因此 SOAP 允许动态的决定类型。但不知道一个函数的函数名、参数的个数和各自类型，怎么可能去调用这个函数呢？没有 WSDL，我们可以从必备文档中确定调用语法，或者检查消息。随便何种方法，都必须有人参与，这个过程可能会有错。而使用了 WSDL，我们就可以通过这种跨平台和跨语言的方法使 Web 服务代理的产生自动化。就像 COM 和 CORBA 的 IDL 文件，WSDL 文件由客户和服务器约定。由于 WSDL 设计成可以绑定除 SOAP 以外的其他协议，在这里我们主要关注 WSDL 在 HTTP 上和 SOAP 的关系。同样，由于 SOAP 目前主要用来调用远程的过程和函数，WSDL 支持 SOAP 传输的文档规范。

同时，由于一个 WSDL 文件实际上定义了一个实现的规范，所以每个遵从

特定的 WSDL 的服务在事实上声明了自己的一个功能上的分类，因此，一个 WSDL 可以被看作是一个专用的功能性的分类术语。这点在服务查询上很有用处。我们可以根据用户发布服务的时候声明的关于服务的功能性信息，也就是它的 WSDL 文档，进行基于性能的服务查询。

3.3.5 统一描述、发现和集成协议 UDDI

UDDI—Universal Description, Discovery, and Integration 的简称[BE02][ABC02]，是以后 Internet 上发布 Web 服务和 Business information 的一种标准。UDDI 是一套基于 Web 的、分布式的、为 Web 服务提供的、信息注册中心的实现标准规范[Dra01]，同时也包含一组使企业能将自身提供的 Web 服务注册，以使别的企业能够发现的访问协议的实现标准。

UDDI 的核心组件是 UDDI 商业注册，它使用一个 XML 文档来描述企业及其提供的 Web 服务。从概念上来说，UDDI 商业注册所提供的信息包含三个部分：

- 白页(White Page):包括了地址、联系方法和已知的企业标识。
- 黄页(Yellow page):包括了基于标准分类法的行业类别。
- 绿页(Green Page):包括了关于该企业所提供的 Web 服务的技术信息，其形式可能是一些指向文件或 URL 的指针，而这些文件或 URL 是为服务发现机制服务的。

所有的 UDDI 商业注册信息都存储在 UDDI 商业注册中心。通过使用 UDDI 的发现服务，企业可以单独注册那些希望被别的企业发现的自身提供的 Web 服务。企业可以通过 UDDI 商业注册中心的 Web 界面，或使用实现了"UDDI Programmer's API 标准"所描述的编程接口的工具，将信息加入到 UDDI 的商业注册中心。UDDI 商业注册中心在逻辑上是集中的，在物理上是分布式的，由多个根节点组成，相互之间按一定规则进行数据同步。当一个企业在 UDDI 商业注册中心的一个实例中实施注册后，其注册信息会被自动复制到其他 UDDI 根节点，于是就能被任何希望发现这些 Web 服务的人所发现。

UDDI 是以 XML 技术为基础的。XML 是一种传输数据的简单模式。通过这一模式，任何数据都 可以在 Internet 上自由的运送，而不管阅读者是什么 System，什么配置模式，其中 XML 数据的传输最重要的是以 soap 的形式。而 UDDI 则是利用了 soap 来进行数据的交流，任何人或任何公司，只要在 Internet 上注册了自己的一些 information(比如 Business information)，则不管在什么时候，它的这些 information 将被其他人或公司知道，从而得到一些商机。

UDDI 规范还定义了相应的 API 和事件请求、响应的标准以及 UDDI 中心的复制方式。任何人都可以利用这个规范建立 UDDI 中心或兼容的 UDDI 中心。

3.3.6 Ontology 技术

Ontology 一词源于西方，在哲学中译为“存在论”或“本体论”，是关于知识（Knowledge）和知晓（Knowing）的理论。组织知识的分类本质上是“Ontology”的问题。在知识分类中，Ontology 更多的是与 Taxonomy（分类法）、Classification（分类）、Index（索引）等词汇具有相通性，但无论在计算机软件设计、人工智能还是在对知识管理知识分类的研究中，Ontology 具有更加实际、深入的应用。我们在研究知识分类过程中，将 Ontology 译为“本体分类”，即指由人们认识事物的本征值（描述事物基本特征的最小单位）形成的、将事物分为不同类别的方法。Ontology 是知识分类的核心与基础，其具体内容在技术篇中将有详细介绍。

斯坦福大学的 Tom Gruber 先生在知识共享的环境中将 Ontology 定义为“概念化的规范（A Specification of A Conceptualization）”，就是说，Ontology 是用来描述概念和概念之间存在的相互关系的，是作为概念定义体系（Set-of-Concept-Definitions）的用法，与哲学中的理解不同。他给出 Ontology 的定义是：在人工智能系统中，用于描绘标准知识的共享和重用，定义普通词汇、描述领域知识的词汇规范（包括等级、关系、功能和其他主题等）被称为“Ontology”。Tom Gruber 先生设计了能够知识共享和重用的“Ontologies”。在一定语境中，Ontology 是对本体事项做的规范，从实际的角度考虑，他们选择用书写一个本体来作为一系列标准词汇的定义，尽管这不是说明概念的惟一方法，但是，它在人工智能软件（例如，读者和上下文语义的独立）中是很好的知识共享工具。

不同的组织具有不同的 Ontology，各个组织对自身 Ontology 的认识也是不同的。根据组织自身特点制订符合本组织的 Ontology，是实施知识管理的基础。我们根据知识的属性、组织的基本特征与共性，结合前人论证的观点，并从大量的实践中提取出以下具有普遍意义的组织知识的分类体系。

组织知识按照知识的特性一般分为：

（1）关于事件、对象的知识（基本信息知识）；

（2）关于程序和结构的知识（业务流程知识）；

（3）关于人、社会伦理和道德的知识（企业文化知识）；

（4）技能、经验、直觉（员工 Know-how 知识）；

（5）关于相互间关系的知识（协调性知识）。

知识的"波粒二相性"决定了作为实体的知识具有一定的静止性，作为过程的知识则体现出一定的动态性。知识之间的划分并没有绝对界限，因此，在以上所描述的组织知识的分类中，同样存在一定的交叉性。

在此分类中，基本信息知识对于一般的企业来说包括企业员工基本信息、制度、文档、文件、培训信息、客户信息、知识产权信息及库房管理信息。基本信息知识是比较容易结构化、标准化的知识。

业务流程知识是组织运作过程中逐渐沉淀下来的流程性的知识，是组织运作的核心。流程知识是组织知识中最核心的部分，组织在自身积累的过程中，流程知识的日渐标准化的程度直接体现了组织成熟与成长的程度。对于以项目管理为主的企业，业务流程知识的核心就是项目管理，包括项目开发、项目管理和人力资源管理.

第四章基于服务社区的 Web 服务的语义化处理系统的总体设计思想

从模块的层次上讲，系统可以被分为两个层次(如图 4.1)，一方面是处于

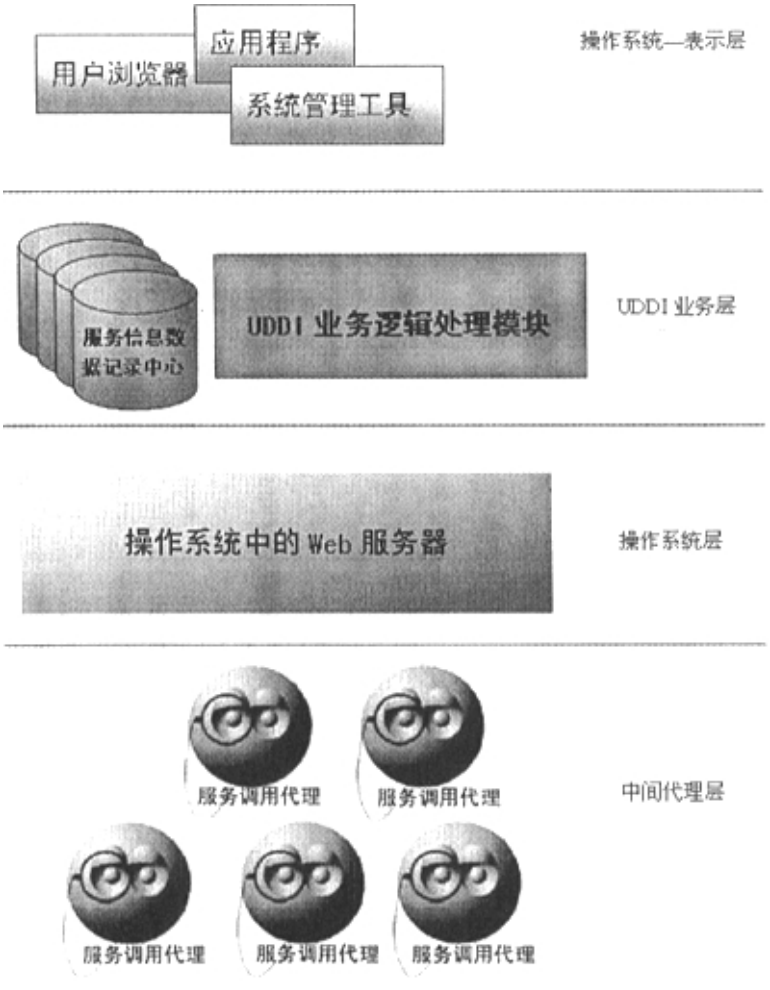

图 4.1 系统的整体结构

Figure4.1 the Architecture of the System

底层的网络数据处理层，在这里就是中间代理层，在这个层次中，系统需要处理用户的调用请求，实现一流服务的包装等功能；另一方面是处于高层的 UDDI 业务处理层，在这个层次中，系统处理所有与 UDDI 想过的业务，例如服务信息的发布，服务信息的查询，语义索引的建立、维护、更新等等。

但是，中间代理层是严格遵循业务层的指示运行的，是业务层的基础；业务层同时也需要代理层为其收集数据，实现其业务功能。

下面，从这两个方面分别说明。

4.1 中间代理

中间代理负责统一的服务调度，屏蔽了服务的具体的访问点，实现了服务的统一映射，为用户提供了位置无关的服务调用；如果结合遗留服务包装器，则还可以屏蔽服务的实现细节，同时，充分利用遗留系统。

服务信息采集

利用中间代理，我们可以采集大量关于服务的信息。这部分经过代理社区采集的信息可以为服务的供需双方提供增值服务。

首先，对于服务的需求者而言，他们除了希望获得服务以外，更希望自己获得的服务在同类服务中是最好的，这里提到的最好，包含了多个方面的内容，包括的服务的响应时间，服务的质量，服务的价格，服务获取前的等待时间等等。我们的系统对于这些服务信息进行整理，对同类型的服务进行综合评价，并排序；这样用户就可以获得一个基于系统标准的服务排名列表；当然，用户也可以根据自己的偏好选择排序方式，进行个性化选择。

其次，对于服务的提供商而言，我们采集的信息可以作为他们改进自身服务质量的依据。因为我们采集的信息里包含同类服务中其他服务的表现情况。从这些信息当中，服务提供上可以知道自己的市场占有率，自己的竞争对手的表现情况，市场的变化，竞争对手对于服务的改进，以及对于自己的优势等等。

此外，我们的系统可以利用采集的信息，有针对性地建立不同的领域性的代理。

4.1.1 实现遗留系统的标准化访问

4.1.1.1 引言

企业在自身的发展过程中必然会累积大量的按照功能划分的基于 Client/Server 架构的传统分布式服务系统（以下称传统服务），这些服务多是独立平台、独立系统，处理方式大多采用黑箱操作。因此具有以下缺点：

- 必须通过特定的 Client 端程序访问
- Client 端如果是不同的平台，需要不同的执行体
- 对 Server 的访问一般而言不能够穿透防火墙
- 已有的系统不能够方便的集成到企业的信息系统中

随着企业的不断发展和生存环境的不断变化，这种按照功能划分的分布式服务系统已经成为了企业管理的负担，严重影响了企业对市场的快速响应。为了解决这一问题，企业迫切需要集成企业中的各个功能体并重组它们的业务流程，从而为用户提供更快捷、方便和高质量的服务。同时，随着以 Web 为中心的经济模式的确立，越来越多的企业开始拓展其在 Internet 上的业务，并寻找可以使它们的业务流程线性化的计算机技术。XML 技术的发展和 Web 服务的诞生给这些企业带来了契机：企业可以建立以 Web 服务为基础的服务体系，从而可以方便地与其他系统甚至其他企业的信息系统进行集成。Web 服务主要有以下优点：

- 可以通过 HTTP（HyperText Transport Protocol）和 SOAP（Simple Object Access Protocol）传输，不受防火墙的限制
- 传输中的数据采用 XML 格式，方便不同系统间自动集成
- 可以通过 UDDI[BE02][ABC02](Universal Discover, Description and Integration 方便的发布，定位与查询
- 可以实现客户端对 Web 服务的动态绑定。客户端可以在查询 UDDI 的基础上，动态选择要绑定的服务，并与服务交互

由上可见，为了实现基于 Web 服务标准的企业应用集成（EAI），有必要将企业的传统服务转换成 XML Web 服务。然而，毋容置疑的是，由于原有的传统服务与其客户端是紧密耦合的，而 Web 服务和其服务访问点是松散耦合的，

因此进行两者之间的等价转换代价非常巨大：整个 Server 系统必须全部重写，并且，原有的 Client 系统将无法继续使用。这种转换尽管可以满足 EAI 的需要，但任何企业都无法接受。本文提出传统服务包装的观点，通过使用.NET 平台的反射（Reflect）和代码树（CodeDom）等特性，在传统服务之上包装出可通过 HTTP 和 SOAP 访问的 XML Web 服务，使得 Web 服务和传统服务间是紧密耦合，Web 服务和其它服务访问点是松散耦合，这样，既满足了 EAI 的需要，又可以是企业原有的系统继续良好地运作。

4.1.1.2 传统服务包装模型

1） 传统服务的运行模式

作为紧密耦合的应用，传统服务需要程序员来告诉它如何找到对方、如何互相通信：服务端开放并监听某个预先定义好的端口，并规定一定的输入/输出数据流格式（或称作协议），客户端通过 Socket 连接服务端所在的 IP，向该端口发送请求数据，然后接受服务端的返回数据。服务端可以接受如下四种基本的数据传输类型：

- 请求/响应（Request/Response）：客户端发出请求，服务端处理请求，并发回响应。
- 恳求/响应（Solicit/Response）：客户端接受并处理服务端的恳求，然后响应。
- 通知（Notification）：客户端接受服务端发出的通知信息，但不响应。
- 单向(One-way)：客户端发出请求，服务端处理请求，但不响应。

这与 WSDL(Web 服务 Description Language)中定义的消息传输类型一致。而服务端可以接受的数据流可以有以下几种情况：

- 普通字节流：多个数据按照规定好的顺序转换成字节流。
- 对象经二进制序列化后的字节流：数据流是保存了对象格式和对象内容的字节流。
- 简单字节流：由单个数据组成的字节流。

对于由二进制序列化的对象组成的字节流，由于保存了对象结构信息，因此很容易被反序列化成对象，进而被序列化成 XML 格式的数据；而简单字节

流也很容易被序列化成 XML 格式的数据，所以后两种情况我们就不加以详细讨论。

2） 服务包装模型

基于传统服务的运行原理，我们设计了如图 4.2 的服务包装模型。在服务包装模型中，包装后的 Web 服务处于中心的位置，它不但应该满足部署在防火墙之外的客户端应用和浏览器应用，还应该可以和企业中的其他应用集成，甚至和跨企业的应用进行集成。同时，企业原有的防火墙内的客户端应用应该还能够发挥原有的作用。

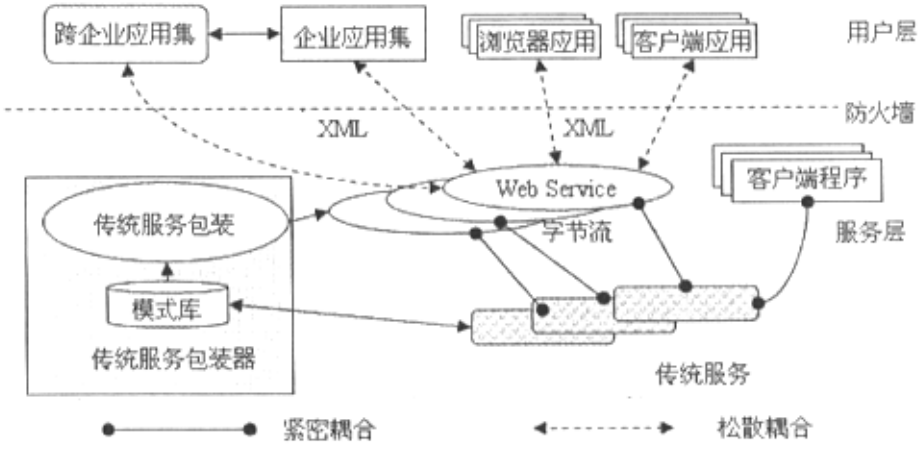

图 4.2 遗留服务包装模型

Figure4.2 the Legal Services Wrapping Module

在该模型中，由于传统服务传输的是无格式的字节流信息，而为了方便进行企业间的应用集成和支持下一代的 Web 应用，需要使用通过 XML 描述的格式清晰的元数据信息。因此，必然需要在两者之间做一个映射，使得用户层到服务层数据流能够根据当前的上下文自动转换。

4.1.2 信息采集模块

服务信息的采集在服务的查询上具有十分重要的意义，这个模块主要是集成在中间代理当中来实现。在具体实现上，主要是将信息统计模块插入到代理模块当中，在网络层与代理的业务逻辑层之间截获用户的调用数据流，并分析该数据，以获得所需的数据。这好像会代来用户数据内容的安全问题，但是，

基于所有的 Web 服务的调用与响应都是适用的 Soap 数据报文，而且，其头部的内容是明文传输的，只有数据包内容是加密的，所以，我们可以利用明文部分获得所需要的信息。注意，为了使用户的服务调用请求得到执行，代理服务器是知道用户请求的服务的信息的，所以，综合这两部分信息，就可以获得对于服务信息的统计。其模块结构如图 4.3 所示：

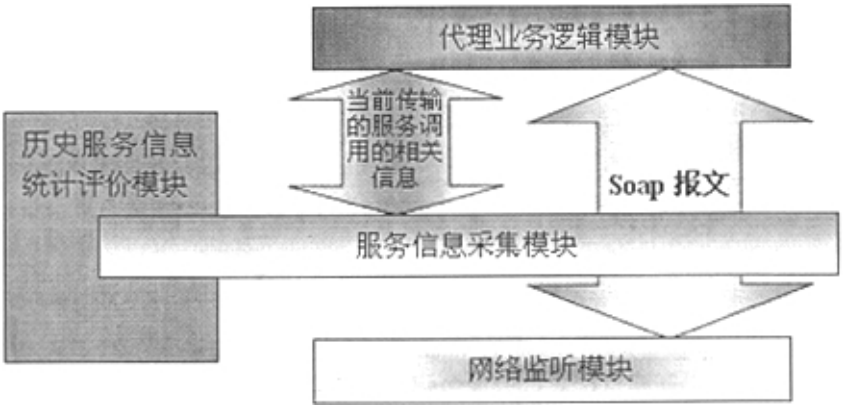

图 4.3 服务信息采集模块与调用代理的逻辑关系

Figure4.3 the Relationship between Information Collection Model and the Agents

4.2 基于服务社区的 Web 服务的语义化查询解决方案

4.2.1 系统总体结构

从用户的角度来看，查询系统是一个虚拟的整体，一切数据都存储在这个整体

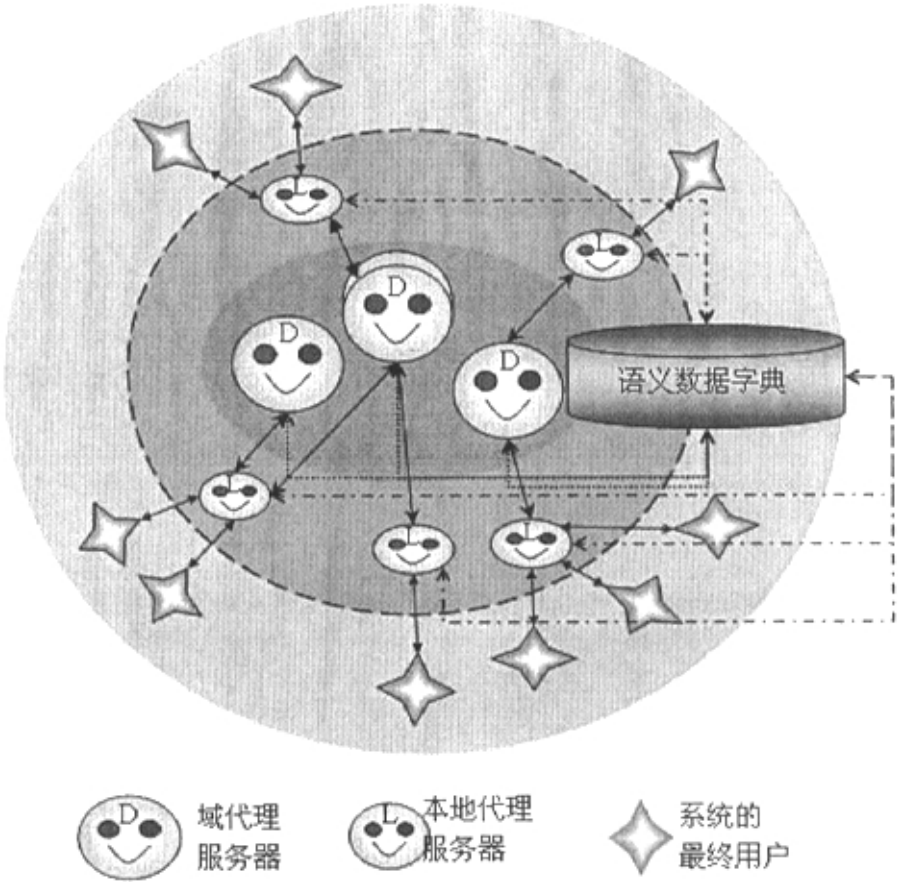

图 4-4 服务信息查询社区结构

Figure 4-4 the Architecture of the Services Information Discovering community

注：图中位于虚线椭圆圈内的为系统内部空间，这是一个包含多个分布式部署的节点的两层的信息管理空间。位于中心的是域代理节点，没个域代理节点可以管理多个本地代理节点，每个本地代理节点也可以在多个域代理节点内进行注册。

的系统之内，所有的操作都集中到服务管理社区（**Service Management Society----SMS**）里统一执行，用户被隔离在系统之外，通过各个接入点(AP)进入系统，再从这些接入点获得各种有用的信息。对于系统的整体空间，我们称之为全域空间，它是系统对整个客观世界进行的抽象，存储了我们的系统需要使用的所有的信息[鲁 03]。

在此模型内部，整个系统实际上被表示成为一个分布式的社区，这个社区被分成两层。顶层（中心层）是域代理服务器层（Domain Broker Level），下面

的是本地代理服务器层(Local Broker)，如图 4.4 所示，图中位于虚线椭圆圈内的为系统内部空间这是一个包含多个分布式部署的结点的两层的信息管理空间。如果将用户考虑进来，就形成了最底层的系统最终用户层。

系统内部建立分布式的层级关系带来了巨大的查询效益。首先，用户的注册信息按照域空间划分，查询的时候就可以将用户的查询请求直接定位于相关的域空间内，不仅缩小了查询时的目标空间，而且有助于进行精确的区域定位，提高查询的准确率；其次，限制用户通过 LB 进入社区，有助于将用户常用的服务信息的本地化，通过将用户常用的数据信息在本地进行缓存，可以进一步提高查询效率与精度；再次，用户的请求在提交到 LB 的时候，会经过标准化处理，转换成系统内部使用的标准形式，减轻了 DB 在处理请求的时候的计算负担；最后，用户通过不同的 LB 进入系统，利用不同的代理来完成各自的需求，可以将用户的请求分散处理，降低系统内出现瓶颈的几率；

同时，通过引入域空间的概念，执行分域处理的原则还可以增加系统的灵活性**[HWJ01]**。首先，可以方便地把一个企业、组织，甚至是另一个注册仓连接进系统空间，作为社区的一个域代理器节点。其次，可以将一些本来不支持 UDDI 规范的服务注册平台引进此系统，这样系统用户就可以使用基于 UDDI 开发的查询客户端查询 DAML 规范下注册的服务。

采用这种层级式的分布式处理原则还可以进一步提高系统的性能，首先，可以提高域空间之间联合查找的效率，平衡系统的负载，建立一个镜像域机制，所谓的镜像域，是指通过将一个域空间分解成若干个镜像子域(原始信息域的一个信息分片)，并将这些镜像子域分散部署到其他域空间的代理服务器上，这些分散在其他域代理服务器上的镜像子域的总和就可以形成一个（或多个）与原始的信息域等价的镜像域。这样，在进行查找的时候，同时需要考虑这种镜像分布，一旦某个域代理服务器负载过高，即可以通过镜像域来平衡负载；其次，极大地提高了系统的稳定性，由于整个域空间内，每个域都有至少一个镜像域，所以一旦某个节点崩溃，我们都可以暂时利用其镜像域维持该节点的功能，并利用镜像域内的数据恢复该故障节点。

对内的分布式层次划分的实现

在此系统内部，依据记录的数据的不同、所实现的功能的不同，以及在系统内所处的位置的不同，系统内的节点被划分为两大类，域代理（**DB**）节点以及本地代理（**LB**）节点。域代理器（**DB**）负责信息发布、存储、管理、查询，同时负责整个服务社区的管理，位于社区的顶层，是社区管理层的功能模块节点；本地代理**(LB)**作为用户进入社区的接入点**(AP)**，执行各种操作，包括本地

信息检索、本地信息存储、与域代理节点通信，用户的信息发布等等，处于社区的底层，是社区内的基本的功能节点。

系统的结构本身通常都可以说明一个系统的一些特点，本系统也不例外。通过建立一个包含了 DB，LB 和用户在内的三层的分布式结构，体现了本系统的几个主要的功能上的特点，那就是查询的精确性，灵活的扩展性，可靠的稳定性，负载的平衡性等。

同时，穿插于 LB 与 DB 之间，我们还引入了一个至为重要的模块，那就是语义分析模块，这个模块是我们系统进行查询的一个基础，利用这个模块，系统可以消除各类歧义，同时，还可以进行一定程度上的模糊查询。

此外，还引入了镜像域的概念来进一步的提高系统的性能。

4.2.2 本地代理模块（LB）

LB 处于社区的下层，是系统与最终用户接触的一个层次，是系统用户接入系统的通道，用户在注册后，可以利用 **LB** 进行各类操作。针对不同的用户对象，系统采用不同的注册机制，个人用户可以到一个公共 **LB** 去注册一个个人用户的身份；而组织用户可以申请获得一个独立的 **LB** 作为组织专有 **LB**。组织专有 **LB** 是由一个组织实体（如联盟）申请创建，处理组织内用户的服务发布、查询请求的本地级代理，其中的数据是联盟级私有的，所以一个组织专有 **LB** 是不会响应外界对于组织内私有数据的查询请求的。

一个 **LB** 不可以在同一个域**(DB)**中注册两次以上；但是一个 **LB** 可以在不同的域**(DB)**内进行注册。经过注册，LB 收到查询请求后的处理过程是，先在本地进行检索，如果失败就通过与之连接的域代理向整个社区请求[PPS02]。

此外，本地代理还可以以推**(Push)**的方式把用户新加入的服务的信息在系统内广播，让相关的 **LB** 记录缓存，减少以后的查询的工作量[Cam02]。

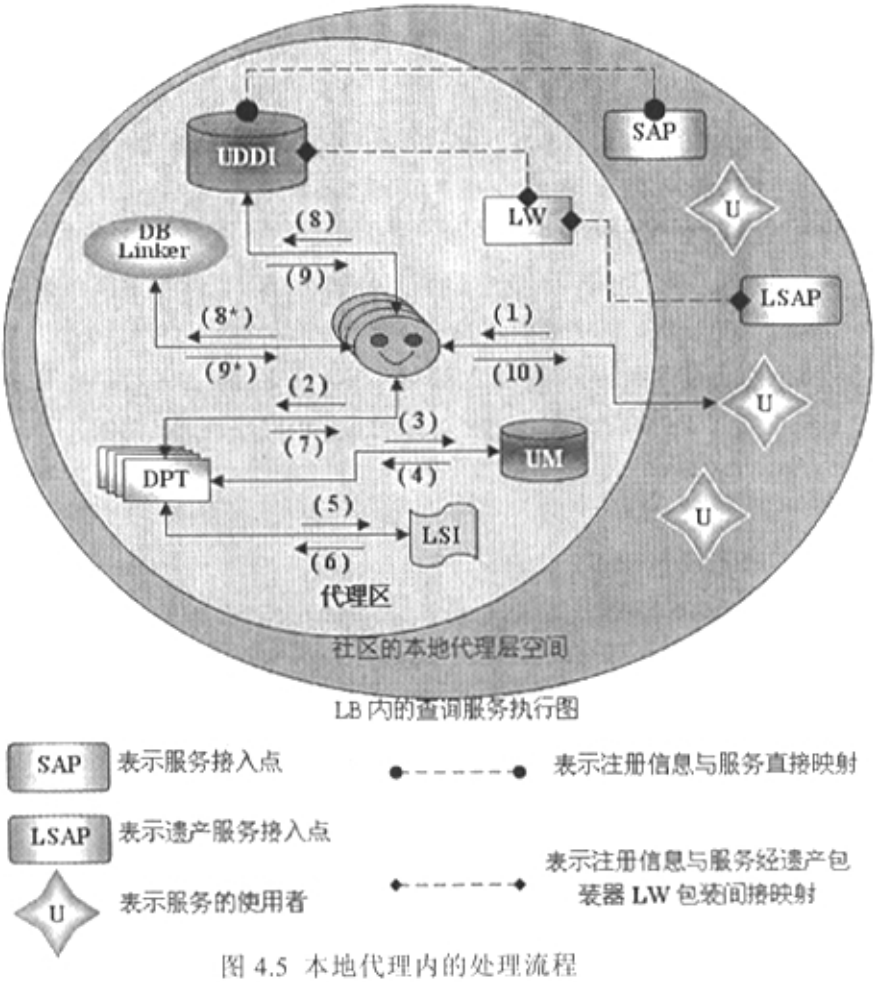

图 4.5 本地代理内的处理流程

Figure4.5 the Dealing Process in the LBs

4.2.3 域代理模块（DB）

DB 处于系统的最高层，是社区内的领导阶层。**DB** 之间需要相互合作来完成用户的请求，他们彼此之间需要通信。**DB** 层记录在社区内注册的全部公有服务信息，每个 **DB** 拥有一张全局数据分配表，以及社区内节点的信息。

DB 是社区的顶级节点，同时也是社区内数据记录的主体部分，它们可以执行绝大部分的查询任务，它们同时还要负责协调整个社区空间内查询服务。**LB** 是只是针对特定的用户群建立的，而且它的存在时间不稳定，同时它存储的数据量相对有限，最后它也不适合存储关于系统的信息。所以，系统引进域代理这一个层次，主要是为了社区的稳定的需要，因为一个社区要有一个稳定

存在的存储空间来保存那些面向公众的服务的信息，社区需要一个机构来管理整个系统的信息存储系统，系统需要能够完成用户的各种查询请求。

在用户进行查询的时候，如果 **LB** 无法满足用户的查询请求，**LB** 就会把该查询请求转移到其注册的一个域代理器（**DB**）中。因为每个域代理器（**DB**）中含有多个 **LB**，所以，在这里为每个 **LB** 创建一个专有的 **LB** 连接器（**LB Linker**），每个 **LB Linker** 只负责处理来自其对应的 **LB** 的查询请求，所以在查询上可以有更多的优化（例如，利用该连接器来缩小对应的 **LB** 查询时使用的语义空间的尺寸，利用历史使用信息对查询结果优化等），**LB Linker** 接下来将用户的请求转移给域代理服务器，由域代理服务器负责余下的处理，同在本地代理内的查询一样，用户的查询请求首先被预处理模块整理，利用全域内语义信息库生成基于域空间的查询方案，在这里需要用到两个功能模块,第一个是全局语义信息表（**GSI**），这个模块记录了社区范围内的语义信息，可以用于查询条件的规范化，另一个是全局节点信息表（**GNI**）,这个模块记录了社区范围内语义概念的分布情况，以及镜像域的分布情况，利用这两部分的信息，生成一个全局查询方案；域代理服务器获得查询方案之后，就会立即执行，查询方案的执行可能会有几个场地，首先是本地域的 **UDDI** 单元，这里包含了两个部分，其一为本地域的数据记录中心，另一个为从其他域镜像过来的镜像子域集合；其次为远程场地，这里也要分几种情况，第一，服务信息分布在远程服务器上，而且本地没有镜像子域，需要经过域连接器（**DB Linker**）来连接远程域代理节点，并获取查询结果，或者是当前的域代理服务器负载过重，需要像其他域代理服务器提请负载平衡请求，这主要通过负载平衡器（**Load Balancer**）来执行，负载平衡器（**Load Balancer**）会首先检索镜像域的分布信息，然后将此查询请求转移到自己的镜像域当中去。最后是通过域代理连接接口（**DLI**）连接的外引的信息注册仓（例如其他公司、组织建立的 **UDDI** 注册仓或 **DAML** 注册仓等），对于这类的注册仓内的信息，我们只提取其中的用于查询的信息，所以在用户想要获得详细信息的时候，我们就要回到原来的注册仓里去查询详细信息。

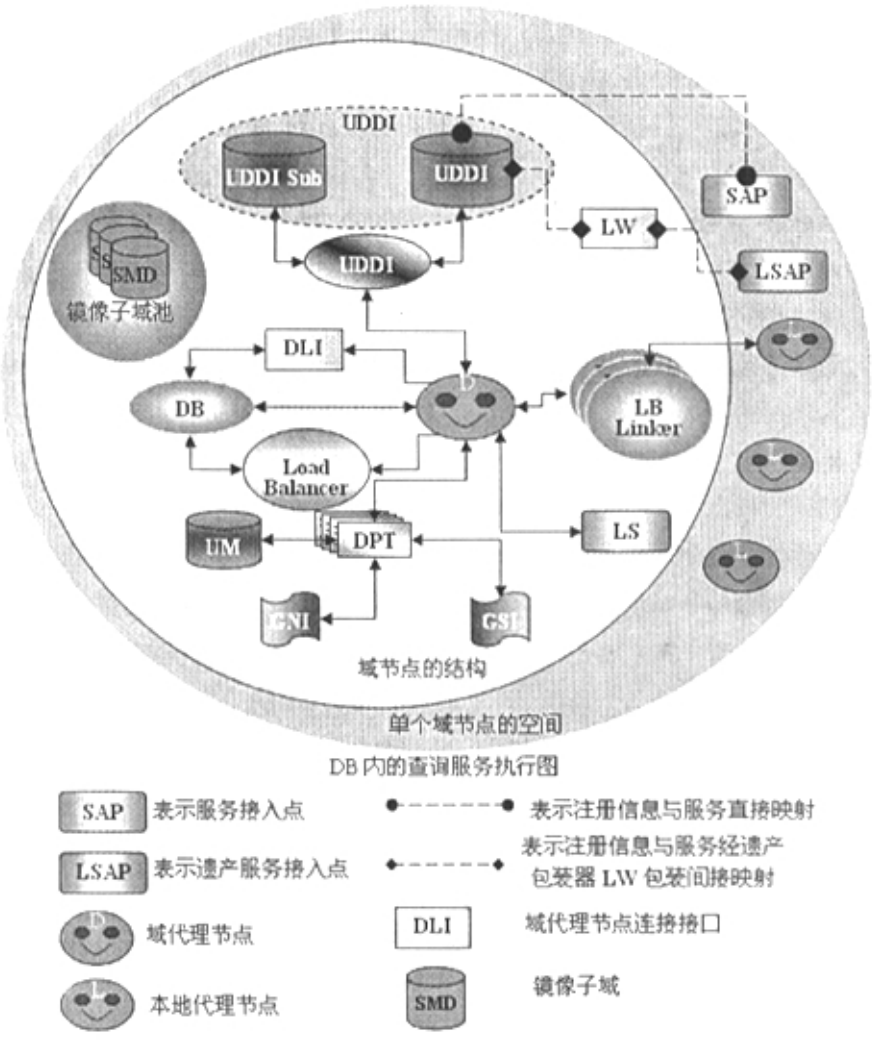

图 4.6 域代理内的处理流程

Figure4.6 the Request Dealing Process in DBs

4.2.4 基于语义域的精确定位

在社区内，现实世界被抽象成一个总的概念空间，我们周围的一切都可以用这个空间里的概念来描述；然后再依据语义概念上的联系与差异，将其划分成若干个不同的领域概念空间（简称域空间），每个域代理服务器节点就负责管理可以出一个域空间内的抽象概念描述的信息（这也是域代理节点的名字的

由来)。

通过这种域的划分，可以有效的减小目标数据空间，提高查询的效率。由于所有的数据都被约束到特定的语义域环境内，在进行具体查询的时候，就可以首先根据查询条件中的语义来进行一次基于域的过滤，然后再在缩小后的数据空间里查询。例如，有一个啤酒生产商，目前，他使用的瓶盖都是自己生产的，现在，为了缩减成本，提高效益，希望将瓶盖生产业务外包给一件专门的生产工厂。为此，他需要找到一个或几个专业的"啤酒瓶盖"生产商。他会搜索承办"瓶盖生产"业务的供应商，在这里，经过我们的"域空间"的过滤，系统会将这一次的检索直接转移到"制造领域"去，由负责该领域的域代理服务器负责处理该项查询。如果在该领域还有进一步的小粒度的划分，系统可能还可以将该查询转移到"小五金生产领域"。

由于每个节点都只专注于特定的领域，所以可以在内容上、性能上做得更好。因为每个特定的领域内的用户，他们在使用术语、查询习惯等问题上往往有相似的地方，所以可以针对特定的用户群建立特定的用户接口，以方便该用户群的使用；同时，特定领域内的用户的信息需求也往往比较集中，例如，经济领域内的用户往往比较关心股市行情，而交通与术领域内的用户有比较关心油料的价格等，所以系统在建立该领域服务的时候也要充分考虑到这部分集中的信息需求，为该领域内的用户提供这种增值的、便利的、用户可订制的、充分体现个性化的服务；再次，同领域内的用户可能需要建立一种便捷的、有效的、功能强大的交流方式，领域服务器完全可一位用户提供这种服务，比如说领域内的 BBS，或者为特定的用户群建立一种临时的会议交流场所等。

除了域代理节点外，本地代理节点同样体现出了这种利用语义概念对数据信息进行区域划分的系统特点。在本地代理级，每个本地代理节点会收集在该节点上执行过的所有的查询操作以及对应的执行结果，并根据该语义域对这些数据进行分区管理。由于每个本地代理节点连接的用户的在现实社会里身份基本相似，所以，每个本地代理节点内实际上记录了该特定的用户群常用的服务的信息，这实际上也形成了另一种形式的域。在用户通过本地代理节点进行查询的时候，系统会首先试着从本地代理的信息存储空间里获得信息。

4.2.5 镜像域

镜像域的引入是本系统的一个显著的特点，通过对每个域空间内的数据进行进一步的划分，可以将一个大的域分解成若干比较小的域，再将这些比较小

的域覆盖的数据分散地复制到若干个域代理节点上，通过这些附着在其他域代理节点上的镜像子域，可以形成一个或多个与原始的域等价的镜像域。

镜像域首先提高了系统的查询效率，当一个域代理节点发现收到的请求不能在本域内完成的时候，它会首先利用节点内的分析模块定位该查询可能获得满足的位置，然后检查本节点的镜像子域池内是否包含该位置，如果该镜像子域存在于当前节点内，则就直接在该镜像子域上执行此次查询，仍然是上面提到过的啤酒生产商，随着生产规模的扩大，他希望将自己的查品推广到一个新的地区，现在他需要查找一些啤酒分销商来代理他的品牌。他本身通过本地代理注册到了“食品加工领域”，所以他的请求会首先被发送到管理该域的域代理服务器去，该代理服务器分析他的请求后，决定该请求应该交由“服务领域”里的“销售服务”子域去处理，这时该代理服务器发现，在本地有“销售服务”子域的一个镜像，它就会利用此镜像子域内的数据，直接将此请求在本地处理，为该啤酒生产商提供一个可供选择的列表，这显然提高了查询的效率；

镜像域同时还提高了系统的负载平衡性，当一个域代理节点的负载过重的时候，该节点会根据一定的规则，选择性地将一些向该节点提出的查询请求转移到该域的镜像域内；在镜像域内，最先接到转移过来的镜像查询请求的域代理节点会首先分析该请求，然后再将该请求转移到可以实际执行该请求的镜像子域当中去，注意：一个域代理本身并不知道他的镜像数据的分布情况，否则在处理时的代价就太高，以至于体现不出镜像域的负载平衡的长处；

镜像域也极大地增强了系统健壮性，由于镜像域的存在，一旦某个域代理节点实效，系统可以利用该域的镜像域继续为向该域发出请求的用户进行服务；
灵活的扩展性

系统的扩展性来自于其特有的松散耦合的结构特点。在这个分布式的层级式设计里，每个域代理节点都只管理对应的域内的数据，在功能上彼此独立。一个域代理节点只有在收到一个属于其它域代理管理范围内的查询请求的时候，才会与其他的域代理节点进行交流，而这种交流也仅限于转发请求。所以，利用系统特有的域代理节点，可以轻易的附加或者剥离一个域的数据，同时又绝对不会影响系统的其他域的正常运作。当然，如果被剥离的域代理节点同时兼任社区领导人的时候，这个剥离过程里还会包含一个新的领导人的选举过程；

此外，利用系统对于域代理节点的这种松散的管理方式，系统外的信息注册仓也可以被轻而易举地引入系统。在这种情况下，外引的注册仓在行为上通常会与本系统有所差异，这就需要系统对这种外来的注册仓进行处理。首先，对该注册仓进行分解，根据系统内当前存在的域，对于该注册仓内的数据进行

分片，将每个分片内的数据复制到对应的域代理节点内，然后将该外引注册仓整个作为一个镜像子域的集合；对于无法处理的数据就不予处理，将它们作为补充数据部分，这部分数据只有在用户的请求无法满足的时候才会被访问，在这个补充数据集里无法利用语义处理带来的各种好处，只能执行最原始的匹配方式来进行服务信息的查询。

对于诸如企业、组织等拥有大量数据信息的实体而言，他们本身是服务的提供商，他们希望自己的服务能够被更多的人访问使用；同时他们也是服务的消费者，他们也希望可以利用其他服务提供商提供的服务为自己创造更多的利益。为了实现这个目的，他们可以申请成为一个本地代理，通过这个本地代理将自己的服务发布到社区，提供给其他人使用；同时通过这个代理在整个社区内搜索自己需要的服务，为自己创造价值。企业级用户申请成为本地代理可以获得很多好处：

(1) 减轻了实体自身的开发费用。实体内部的用户同样需要一个可以发布、查询服务的系统来共享、使用实体内的各种服务，通过申请一个本地代理的身份，实体就获得了一个功能完善的信息注册仓，此注册仓的开发维护费用是不需要提出申请的实体来承担的；

(2) 提高实体内私有信息的安全性，本地代理是可以区分本地私有信息与社区内共有信息的，对于那些仅限于实体内用户使用的信息，本地代理管理员可以将其声明为本地私有的，这样，除了连接到此本地代理的用户，其他人是无法获得这些信息的；而那些可以为大众使用的信息，则是不受访问限制的；

(3) 为实体带来了更多的机会，实体通过本地代理将信息发布出去，增加了被访问的机会；

(4) 为实体提供更多的信息，利用系统的集中性，可以为这些实体用户提供更多的同行业的比较信息，以及可用于改进自身服务质量的信息；

以前文提到的啤酒生产商为例，以该生产商为龙头，带动了周边一系列相关企业的成长，现在，为了实现共同高速发展的目的，这些企业决定组成一个联盟，在这个联盟内，加盟的企业可以共享联盟内私有的信息，可以使用其他人提供的只对联盟用户开放的服务，可以充分利用别人闲置的资源为双方谋求更大的利益；同时，他们还希望可以将自己能够提供的服务向公众发布，让更多的人可以访问。为了达到这些目的，他们只要向我们的系统申请成为一个本地代理用户，并在系统分配给他们的本地代理节点里发布他们的服务信息，系统可以阻止那些联盟内私有的信息外泄，同时还会将对外公开的信息发布到整

个信息社区里。在这个过程里，整个联盟没有花费人力、物力去开发一个信息发布与查询系统就获得了他们想要的效果，同时，他们也获得了一个巨大的用户群，那就是整个服务社区的全体注册用户，对于所有的用户而言，这显然是一件令人十分激动的事情。

如果上述的实体的数据量很大，就可以申请成为一个域代理（这种情况可能更多的适用于另一个服务信息注册中心）

关于系统灵活性的一个突出的表现就在于对用户无干扰的系统升级能力。由于我们的系统内采用了 DB 与 LB 分离的技术，同时引入了镜像域的概念，可以为我们提供一种黑箱升级的能力，当我们的系统需要对某个域代理进行升级更新的时候，可以利用该域的镜像域暂时维护其工作的正常进行，同时在后代对该域代理服务器进行维护、升级等操作，而这一切对于用户而言都是不可见的。

4.3 基于服务社区的 Web 服务的发布与查询

4.3.1 用 UDDI 进行注册

发布到 UDDI 是一个比较直接的过程。第一步是确定在 UDDI 上为公司及其服务建立模型所需的基本信息。之后便可以进行实际注册。这可通过基于 Web 的用户界面或编程两种方法完成。

步骤 1：为 UDDI 条目建立模型

步骤 2：注册 UDDI 条目

4.3.2 基于服务社区的 UDDI 改进方案

4.3.2.1 基于服务的语义信息的查找

系统中，一项十分关键的技术就是基于语义的查询。

首先，利用语义信息可以消除歧义。由于每个人对于客观世界的认识不同，对于同一概念的理解也会有所差异，表现在服务查找上，就是服务的供求双方对同一概念的描述的不同；利用带语义信息的分析处理，可以消除这种源自理解差异的歧义。具体地说就是，在提供商发布信息的时候进行一次标准化处理，将用户的输入用系统标准词汇、语法进行描述，然后将信息转化成为系统标准

格式进行记录；当用户进行查询的时候，对查询条件进行同样的标准化处理，这样，具有相同的语义概念的不同的服务描述（包括发布时的描述信息，以及查询时的约束条件）就会被统一成一种可匹配的描述。利用这种处理方式，由于用户的理解偏差形成的不精确的（甚至是错误的）查询就得到了一定程度上的修正；

其次，还可以实现一定程度的模糊查找。客观世界里的实体之间是存在联系的，而语义信息可以描述客观世界里的实体之间的联系，利用这一特性，可以实现一定程度的模糊查询。在语义信息节点为查询请求生成查询方案的时候，需要同时考虑实体信息之间的联系，并以此生成一系列的模板，利用这些模板，可以实现一定程度的智能查询，扩大查询范围。

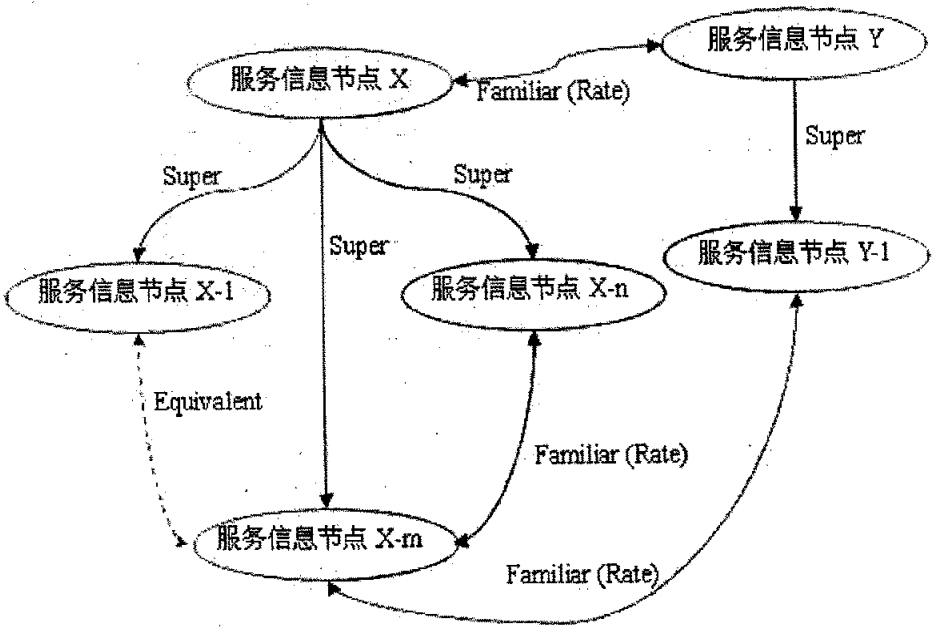

图 4.7 服务信息节点之间的网状关系

Figure4.7 the Net-Like Relationship Among Service Information Nodes

4.3.2.2 语义预处理

语义预处理实际上在发布服务的过程以及查询服务的过程里都要用到。在用户发布服务的时候，用户的查询请求首先被传递到语义预处理模块（SPT），这里的标准信息生成模块首先利用语义信息将用户的描述信息转化为系统内使用的标准描述，进行第一次消歧。具体说来 ，在这个过程里，用户的输入会

首先经过一个标准词汇词典，该词典记录了每个词汇的标准形式，通过这个词典后，就不存在不标准的描述了；然后，利用系统纪录的语义信息来描述实体间的联系，标记服务之间的联系，从而形成一个彼此关联的网状结构，网络上的每个节点就是一条服务的描述（针对于某个方面的描述），这个步骤主要是为了扩大查寻范围，以实现模糊查询的目的；第三，针对用户发布的数据，我们根据需要进行分片，每个分片实际上说明了服务在一个特定的方面的表现。

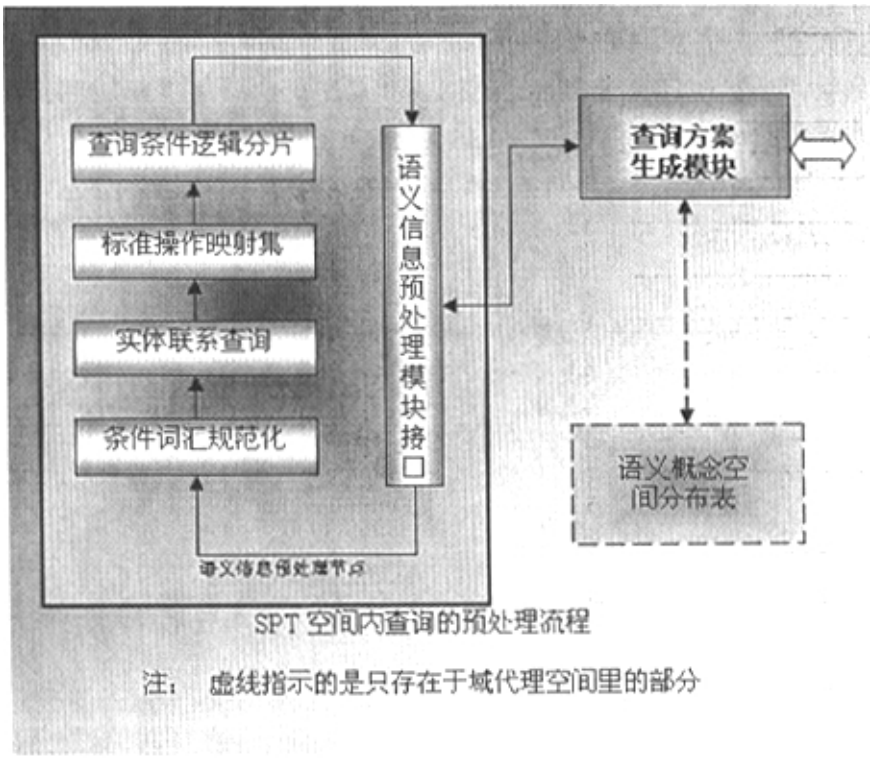

图 4.8 语义预处理

Figure4.8 the Semantic Pretreatment

在进行查询的时候，在每个代理节点内，与发布的过程很相似，用户的查询请求首先被传递到语义预处理模块（SPT），利用标准信息生成模块将用户的描述信息转化为系统内使用的标准描述，进行第一次消歧；然后，利用系统纪录的实体间的概念上的联系，并参照用户的近似度要求，从而形成一个彼此关联的网状的查询结构，网络上的每个节点就是一条服务的描述（针对于某个方面的描述），被该网络中的节点覆盖的描述就是满足用户的近似度要求的服务集合。这个步骤主要是为了扩大查寻范围，以实现模糊查询的目的；最后，用户的请求还会被划分成若干个分片，利用这些分片来优化查询。

4.3.2.3 基于语义的分域处理

我们的语义处理还包含了另一个方面，那就是对于整个目标空间进行分域处理。现实世界是可以通过有语义的描述进行抽象的，而现实世界里的实体又是存在千丝万缕的联系的，而这种联系反映在对客观世界的抽象上，就是语义概念之间的联系，我们可以将联系紧密的语义概念归类到一起，形成一个语义概念域。我们这里所谓的分域处理，就是指为每个这样的语义概念域建立一个管理服务器，然后，依据服务的描述，将服务划分到不同的语义区域当中去，进行分域管理。

利用这种分域处理原则，我们可以提高检索效率，降低总的检索代价，这是因为只要选择一个合适的域代理服务器，我们就可以把平均的检索空间缩小到系统的总的数据空间的 1/d（假定系统内包含 d 个域空间）。这显然减少了大量的无用比较，即可以提高效率，又可以缩减代价。

建立分域处理原则带来的另一个好处就是可以针对不同的域进行个性化处理。由于每个域只管理本域内的数据，所以其面向的用户群也是有针对性的，所以系统就可以针对不同的用户的群提供不同的增值服务。

4.3.2.4 建立语义索引

建立语义索引的目的同数据库中的索引的目的一样，就是为了提高查询的时候的效率与速度，同时上述的基于语义的分析为我们提供了一种建立索引机制的途径。

利用系统在用户的发布服务的时候的分片结果，我们可以在其上建立一种偏序关系，这种偏序关系实际上就是我们建立索引的依据

我们选择其中的几个分片作为建立索引的条件，例如服务的描述分片，我们根据对服务的功能描述的分析，将不同的服务分类，依据概念的关系建立一个查询索引，在这里采用了索引树作为服务的索引的记录结构，每个节点之中记录了与该节点管理的服务完全匹配的服务的描述，注意，在这里执行的是完全匹配原则，而且在不外加其他信息的情况下，在该分片管理的领域范围内，这些描述信息已经可以十分完备的描述这些服务了，这也就意味着，在这个分片上，同一个节点直接关联的服务，在该分片上是完全等价的。这就为服务的

动态替换提供了一种可能的解决方案。进一步说，在一个服务中的某个操作的具体执行是上下文无关的情况下，可以将该操作也归并到一个节点当中，在这种情况下，动态替换还可以到达操作的级别。

每个节点还会有若干个子节点，在描述的概念上，子节点是父节点的一个子集，是对父节点的一个更强的约束集，所以归属于子节点的服务一定也符合父节点中的描述的要求，但是，由于约束的加强，子节点中的服务可能只能完成父节点中关联的服务的部分功能，至于孙子节点，其完成的功能可能就会更少了；但是这种通过概念上的父子继承关系来约束的服务之间必然存在联系，这种联系就要通过等价因子来说明了。

我们将每个节点管理的所有服务称为一个等价连接空间内的等价服务。对于那些并不直接归属于某个节点，但却归属于其下属节点管理的服务，我们将上下两级节点管理的服务也称为等价服务，但是由于其描述上存在一定的差异，所以还要引入一个等价因子的概念来描述这种服务间的等价与相似的关系。至于节点之间的相似关系，也同样要由等价因子来描述。

根据上述描述的规则，不可避免的会造成某个描述同时是两个或者两个以上的节点的子节点中的服务描述部分，为了解决这个问题，在这里我们允许每个节点同时参与到多个等价连接空间里去，也就是说 我们允许多点继承。

在对服务的发布信息进行分片的同时，我们还要利用其中的一些常用的的分片来建立一个关于服务信息的索引结构。也就是依据服务的描述将服务进行分类，每个分类实际上说明了一个等价连接空间，在同一个等价连接空间之内的所有的服务都是等价或者相似服务。而我们建立索引的目的是使服务查找的绝大部分工作在服务发布的时候完成，并且只需要执行一次。

4.3.2.5 利用语义索引的信息检索性能分析

基于语义的索引带给我们的最大的好处就是，就是我们可以在不同的语义分片上进行有序的查找了。这里的关键是怎样进行合理的安排，来获得令用户满意的查询结果。

分片是实现查询的一个至关重要的部分。经过消歧处理后，就需要对用户的请求进行分片，分片依据各种服务描述的特性进行，例如，服务的有效地域，服务的价格等松散的约束条件要作为先进行分片的依据，这类分片说明的信息往往比较简单直观，进行匹配的时候所花费的代价也比较低；至于服务功能的

描述，作为严密的约束条件，它说明的信息比较复杂，进行匹配的时候花费的代价往往要大得多，所以需要放在后面。依据这种策略，服务的分片实际上说明了查询执行的方案。首先进行松散分片上的匹配，可以利用较少的代价检索较大的数据空间，经过这次检索，目标空间就会缩小，这时候再选择以较为严密的分片作为约束条件来检索，就可以实现既准确的定位、又花费较小的代价的查询。

假定我们现在的注册仓内有 10，000 条信息，用户在查询的时候，其条件被分解为三个分片，就是“价格（Seg1）”，“性能（Seg2）”，“功能（Seg3）”三个部分。在执行用户的查询请求的时候，我们就先利用 Seg1 对目标数据空间进行一次检索，这里要进行比较的信息只有一项，假定其花费的代价为 C，则检索全部的目标空间需要代价为：

代价 1 = C * 10，000 = 10,000C

然后用 Seg2 对得到的结果集进行第二次筛选，在这里要比较服务的平均响应时间和服务的稳定性，每次比较的代价为 2C，假定第一次检索后满足条件 Seg1 的信息有 10%的条目能够满足条件，即 1，000 条，则有

代价 2 = 2C * 1,000 = 2,000C

最后再用 Seg3 对余下的数据进行第三次筛选，由于这里涉及到将用户的输入约束与服务发布时的描述进行基于语义的比较，所以代价会很大，假定为 nC（ n >> 1 ），同时，假定经过第二次的筛选，有 10%的条目能够满足条件，则余下 100 条信息需要进　行进一步的比较，则有

代价 3 = nC * 100 = 100nC

所以，总代价 1 = 代价 1 + 代价 2 + 代价 3 = 10，000C + 2,000C + 100nC = 12,000C + 100nC

如果我们以 Seg3，Seg2，Seg1 的顺序执行上述的查询，其代价为

代价 1’ = nC * 10,000 = 10,000nC

代价 2’ = 2C * 1000 = 2,000C

代价 3’ = C * 100 = 100C

总代价 2 = 代价 1’ + 代价 2’ + 代价 3’ = 10，000nC + 2,100C

二者的比例为 r = 总代价 1 / 总代价 2 = （12，000C + 100nC ）/ (10,000nC + 2,100C)< 12 / 10n + 0.01 < 12 / (10 * 2) + 0.01 = 0.61（ 注意： n >> 1 ）

同样可以证明，以任何其他顺序执行查询的时候，所花费的代价都比 Seg1，Seg2，Seg3 这一顺序的代价大。

4.4 小结

首先，利用中间代理，我们可以将遗产系统当中的旧有的 Web 应用包装成标准的 Web 服务，这样，我们就可以采用统一的服务发布与查询机制来处理这部分遗留系统上的应用。同时，通过这些代理，我们收集了大量的服务的信息，包括服务的表现、服务的被访问频度等信息，利用这些信息我们可以对服务进行评价。

其次，通过建立一个等价服务社区，我们还可以进行区域处理。利用这个等价代理社区，我们将整个现实空间映射成为若干个彼此独立但是在逻辑上又有所联系的域，每个域代理负责一个特定的领域内的数据，这样，既可以提高系统的性能，又可以为系统用户提供更多的服务。

第三，我们利用语义信息建立了一个查询索引。这个索引不仅可以显著地提高查询的效率。而且，利用语义信息，查询操作可以精确到服务内的操作级上，这样的结果对于用户来说是很方便的，尤其是在服务动态合成的时候，服务合成模块[HJM00]只需要关心“我要什么”，而不必关心“我怎么做”。就可以得到一个满足某个标准接口的操作，至于合成过程里的歧义问题根本不需要考虑。

利用语义信息来描述服务是利用语义信息进行服务查找的前提，在一个服务的信息被发布到我们的注册仓中的时候，它首先被系统解析，然后存储为一种系统内自定义的格式，一种十分有利于查找的格式，为此，我们建立了一种基于 Ontology 概念的语义信息管理模型，在利用 Ontology 基本概念的模型的基础上，加入了服务信息描述因素，从而提高我们的系统在服务定位上的准确性。

第五章 基于服务社区的 Web 服务的语义化处理系统中关键模块的实现

对于系统的具体实现而言，这里采用了基于模块的方式，所以我们也以分模块的方式来逐个模块的说明。

5.1 中间代理的实现

在系统设计的阶段里，我们将中间代理定义为一个执行系统网络数据传输的基础结构，负责统一的服务调度，屏蔽了服务的具体的访问点，实现了服务的统一映射，为用户提供位置无关的服务调用等；结合遗产包装器，还可以屏蔽服务的实现细节。

5.1.1 代理服务器的实现

从系统结构的角度看，系统可以划分成两个模块，即本地代理（LB）模块与域代理（DB）模块。但是，由于每个节点中都包含一些相同的功能，所以，在实现上，系统可以分为三个模块，将这些相同的功能提取出来，作为查询系统内基本的节点组成结构(Node-Infrastructure)，本地代理专有结构(LB-Addition)，域代理专有结构（DB-Addition)。其中的 Node-Infrastructure 是域代理与本地代理都有的部分，主要负责节点内的各种基础的操作，如节点内查询，节点内语义分析等；LB-Addition 则负责本地代理内特有的功能，主要有用户身份认证，网络连接处理等，DB-Addition 在域代理内要负责用户验证，镜像域，社区管理，整体空间内的查找等。

在代理模块的实现上，充分考虑到本地级代理与域级代理在功能上的重叠部分，将这些重叠的部分提取出来，形成一个共有的模块，这就是两级代理的公共组件。在这里，主要实现了代理的网络功能部分，例如网络监听，数据转发等功能；以及用户管理，语义数据库查询、管理，服务信息的查询等。

class Node-Infrastructure

```
{
// 当前类当中使用的变量
User-Management UM;    //对用户的身份进行验证
SemanticDB SDB;    // 语义数据库
Service-Discover SD;    // 服务发现处理模块，采用工厂模式，对于本地和域代理两种不同的环境产生不同的查询实例。
// 当前的类当中使用的函数
Create-New-Agent ( );
}
// 用于用户的信息管理
class User-Management
{
Legalize( ); // 用户的身份认证
}
// 一个具有语义信息的数据库的管理模块
class SemanticDB
{
Create-Plan( 查询条件 ); // 利用语义信息库内的数据产生一个查询计划
}
// 服务查询的执行模块
class Service-Discovery
{
Plan-Carryout( ); // 执行查询计划
}
```

本地代理扩展部分的实现（伪代码）

两级代理中的公共部分只能够完成一些代理的基础性功能，但是每级代理中都还同时具有自己的个性化的东西。对于本地代理而言，它具有的一个个性化的东西就是代理内用户的认证，这个功能从代理的公共模块里继承，负责管理本地代理内注册的系统用户的登陆以及注销；创建处理每个用户请求的代理实例；创建查询方案；向特定的域代理求援等。

```
(网络监听模块接收到查询请求后)
Node-Infrastructure->Create-New-Agent ( ); //创建一个新的查询处理代理
```

```
Node-Infrastructure->User-Management-> Legalize( );   //对用户的身份进行验证
If( 身份验证通过 )
{
Node-Infrastructure->SemanticDB->Create-Plan( 查询条件 );   // 生成查询方案
If（ 该方案可以在本地执行 ）
{
Node-Infrastructure->Service-Discover->Plan-Carryout( );
// 返回结果
}
else
{
LB-Addition->Domain-Select（ ）;   // 当查询请求不能够在本地代理级获得满足的时候，就向域代理求援
LB-Addition->Turn-to-Domain( );   // 向被选择的域代理求援
// 返回查询结果
}
}
else
{
// 返回认证失败信息
}
```

域代理扩展部分的实现（伪代码）

在域代理这一级别内，同样有用户身份认证，但是这里认证的用户是本地代理的身份，也就是说，本地代理在接到用户的请求后，会首先进行认证，并尽最大的努力来实现用户的请求，一旦用户的请求得不到满足，就会代表用户向域代理申请服务，所以与代理见到的实际上是本地代理为用户的请求；创建查询方案，在这里，常建查询方案的过程里还要考虑数据的全局分布，利用域代理内的全局服务数据分布表来决定该服务的具体的执行的场地（请求数据的查询与分析，选择特定的域代理来执行）；平衡负载等。

```
(网络监听模块接收到查询援助请求后)
Node-Infrastructure->Create-New-Agent ( );
//对用户的身份进行验证
Node-Infrastructure->User-Management-> Legalize( );
If( 身份验证通过 )
{
// 生成查询方案
Node-Infrastructure->SemanticDB->Create-Plan( 查询条件 );
 If（ 该方案可以在当前代理执行 && 当前代理没有超负荷 ）
{
Node-Infrastructure->Service-Discover->Plan-Carryout( );
// 返回结果
}
else if(该方案可以在当前代理执行 && 当前代理超负荷 )
{    // 寻求负载平衡
DB-Addition->Load-Balance->Mirror-Select( );
DB-Addition->Load-Balance->Mirror-Carryout( );
}
else(该方案不可以在当前代理执行)
{
// 当查询请求不能够在当前域代理内获得满足的时候，就像其他域代理求援
// 首先将用户的请求根据语义定位到特定的域代理空间里去
DB-Addition->Domain-Disperse->Semantic-Position( );
DB-Addition->Domain-Select（ ）;
// 向被选择的域代理求援
DB-Addition->Turn-to-Domain( );
// 返回查询结果
}
}
else
{
// 返回认证失败信息
}
```

5.1.2 服务信息采集

在第四章里曾经说过信息采集模块式系统中采集服务信息的部件，利用这个模块可以截获服务的在被调用过程中的信息，包括相应时间，处理时间，可用性，出错率等，其关键技术就在于网络数据的截获，所以这个部件理所当然地被牵入到中间代理当中了，其处理过程可描述为：

```
while(true)
{
    getHttpPacket( );//获取 http 包
    extractSoapHeader( );      //从网络数据包当中获取服务的 Soap 包头
    analyzeSoapHeader( );      //解析 soap header, 获得调用参数
    sonsultAngent( );    //向 Agent 查询该调用的对应的信息
    recordInformation( ); // 存储统计信息
}
```

5.1.3 实现遗留系统的标准化访问

模块的总体架构如图 5.2 所示。传统服务提供者将其要对外发布的服务的 SAP, IS, OS 注册进入服务包装器，由服务包装器包装该传统服务并生成 Web 服务。

服务提供者可以选择服务的部署方式：部署在本地或部署在服务包装器所在的 Web Server 上（如果服务提供者不具备.NET 环境）。然而，如果选择部署在服务包装器所在的 Web Server 上，则必须对防火墙开放该传统服务所在的端口（port），否则，Web 服务无法与传统服务进行交互。部署完毕后，系统可以自动将包装好的 Web 服务发布到指定的符合 UDDI2.0 规范的注册中心，方便系统日后的集成。

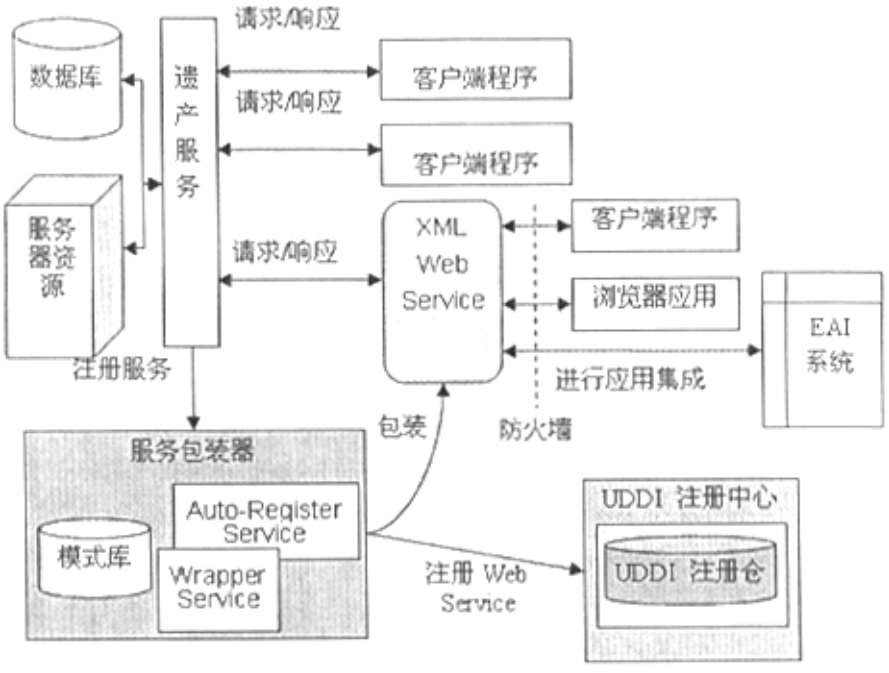

图 5-2 遗留服务包装模块的总体架构

Figure5-2 the Overall Architecture of Legal Services Wrapping Model

5.1.3.1 服务包装器设计方案

为了便于实现，我们在功能上将传统服务包装器分为三层，分别是：服务层，数据层和合成层。服务层主要处理服务提供者的提供的服务模式（SAP, IS, OS）；数据层将服务模式持久化到传统服务模式库中；合成层处理传统服务的 Web 服务形式的包装，主要包括四个的部分：流-类映射器（SCM），对象生成器（OG），流生成器（SG）和代码生成器（CG）。生成后得到一个 Web 服务和其自描述的 WSDL 文档。WSDL 文档中包含一个输入流的 XML-Schema 描述和一个输出流的 XML-Schema 描述。同时，提供一个可绑定的操作（Operation）：Invoke。

这样，一个典型的对传统服务进行包装的流程如下：

- 服务提供者绑定传统服务包装 Web 服务，注册其 SAP、IS、OS 到模式库中
- 传统服务包装器调用流-类映射器，生成输入输出类
- 传统服务包装器将对象生成器和流生成器绑定到生成的 Web 服务中（供

执行期使用）

包装后的 Web 服务的运行流程如下（以 Request/Response 模式为例）：

- 服务请求者通过 SOAP 访问该 Web 服务，提供请求的 XML 文档
- Web 服务调用流生成器，将经过 XML 反序列化生成的对象转换成字节流格式
- Web 服务绑定传统服务，发出请求的字节流
- 传统服务处理请求的字节流，然后返回响应的字节流
- Web 服务接收到响应的字节流后，调用对象生成器，生成响应的对象，然后经过 XML 序列化后被打进 SOAP 封套，返回给服务请求者

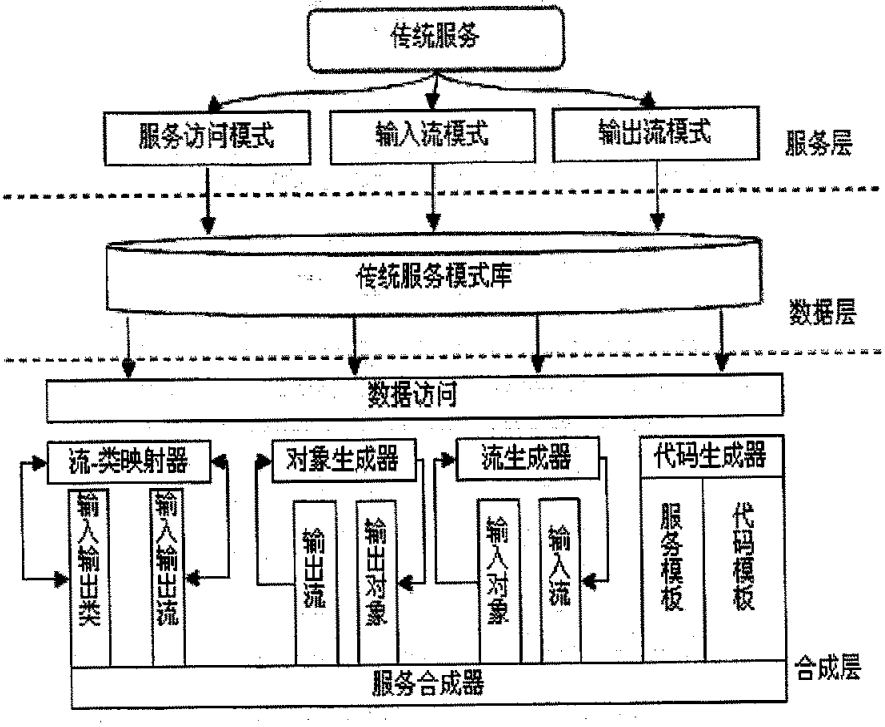

图 5.3 遗留服务包装器

Figure5.3 Legal Services Wrapper

5.2 关于 UDDI 的改进方案

5.2.1 增强 UDDI 注册功能

为了增强 UDDI 注册中心的功能，实现一定程度上的智能查找，实现结果记得准确性，在这里引入了语义概念来增强服务的查询性能。其语义处理部分的模块图如图 5.4 所示。

5.2.1.1 语义预处理—服务发布

查询的预处理实际上在发布服务的过程以及查询服务的里都要用到。在用户发布服务的时候，用户的查询请求首先被传递到语义预处理模块（SPT），这里的标准信息生成模块首先利用语义信息将用户的描述信息转化为系统内使用的标准描述，进行第一次消歧。

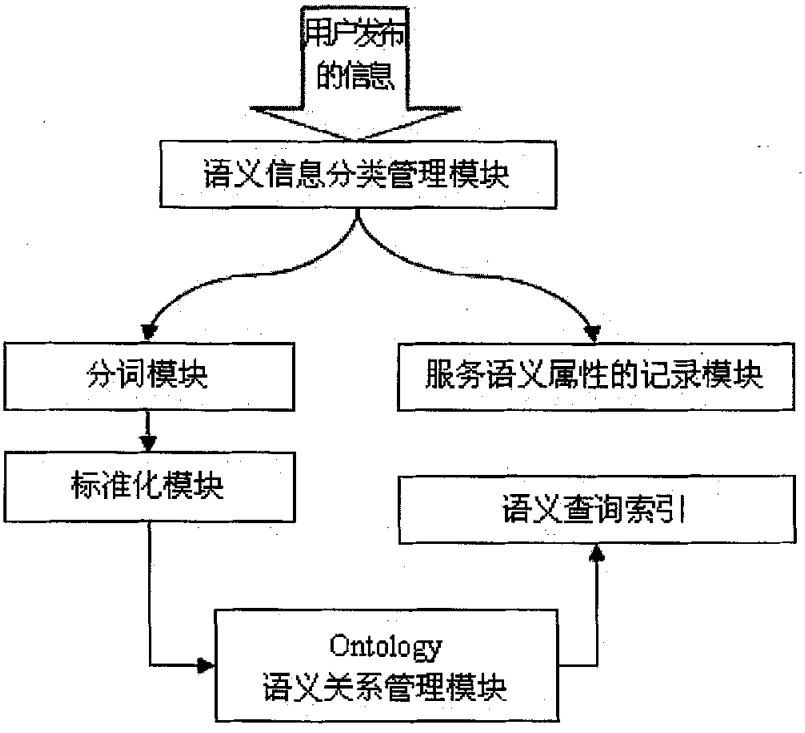

图 5.4 语义预处理在服务发布过程中的流程

Figure5.4 the Service Publishing Process in the Semantic Pretreatment Module

分词模块主要是将用户的输入分解，提取其中的关键性词汇，用来理解用户的对于服务的描述。分词是语义化处理的基础，决定了其后的各个阶段的处理的效果。**标准化模块**主要是对用户的输入进行处理。通过将非标准的描述转化为标准描述，可以减少系统的存储开销，降低查询时的各种消耗，提高查询的准确率。Ontology **语义关系管理模块**主要负责将前两个步骤中输入的数据按照语义的关系实行分类，为将来的检索提供排序依据。这个分类信息实际上就是一个建立索引的排序键。经过这个模块的处理后，每个服务的描述信息被最终归类到一个描述模版中，这个描述模版说明了一个服务所应具有的功能，我们称之为一个系统单词。

语义查询索引利用前面的三个步骤中的结果，在索引数据结构里建立关于服务的有序排列，这个排列中的分类信息说明了服务的功能，每个分类下记录的服务就是具有同样的服务功能的服务的集合。这里涉及到两个方面：

（1）**语义索引的设计**

从设计思想上说，这部分的关键在于如何将服务的描述部分与 UDDI 注册仓（或者其他的用于 WEB 服务查找的信息注册仓，例如 DAML 注册仓）当中的其他部分分离。也就是说，在 UDDI 注册仓之外重建一个有利于检索的数据仓，这个数据仓可以不使用数据库管理系统，可以是一个外建的查询树，其目的是能够在最短的时间内获取最好的结果。这里，最好的结果并不是说最符合用户需求的服务信息，而是尽可能让用户满意。

在建立索引的时候，使每个系统单词（就是经过 Ontology 信息分析管理模块处理后的结果）成为一个等价链接的入口，每次新加入一个服务，我们就利用已经存在的每个系统单词试着将它加到一个特定的等价链接空间里。

同时，在建立等价链接空间的时候还要遵循下列的一些规则：

- 每个系统单词只用来记录直接等价连接空间，也就是说，如果存在两个约束等价链接空间，其中一个空间的约束集为 Cs1，而另一个空间的约束集为 Cs2，两者满足关系： Cs1 为 Cs2 的子集，则两者的公共空间，只记录在与 Cs2 关联的空间记录里，对于这部分数据，只在 Cs2 中记录一条关于这两个约束集的等价信息，并引用记录在 Cs2 空间内的数据，对于 Cs1 空间内比 Cs2 空间多出的部分则在 Cs1 的入口表内另行记录；例如，对于两个约束集合，**S1**:C1,C2,C3,C4,C5 和 **S2:**C2,C3,C4，S2 肯定包含 S1，所以，S2 的直接链接当中不记录 S1 当中的服务信息，这部分服务信息，由 S1 中的间接等价引用指针予以记录；
- 每个新加入的服务都会拥有一个等价约束空间的记录，如果在已经存在

的模板集合内无法为其形成等价链接空间约束，就将其分配到一个新的等价连接空间里，这个等价连接空间具有对该服务的描述，说明了同类服务的功能，在这个过程里，我们创建了一个新的等价连接入口；

- 等价链接空间是由所有满足等价约束的服务的扩展服务信息记录单元组成的，其中包含关于服务的排序信息，也就是说，在等价连接空间里的服务列表示一个已经排好顺序的有序序列；
- 对于约束集，利用本体论数据仓来获取其 UUID；

（2）语义索引的实现

系统在实现的时候采用了一棵准 B 树形成的森林。此森林管理了以系统单

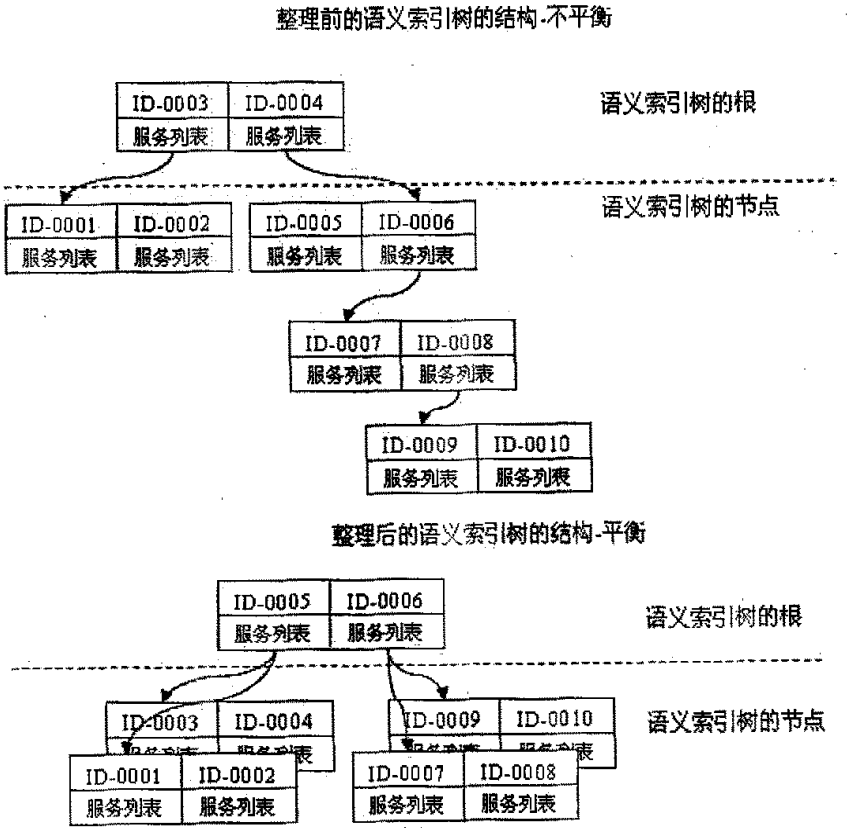

图 5.5 语义索引的插入与平衡

Figure5.5 Insertion and Balancing of the Semantic Index

词作为键值的树，之所以被称为准 B 树，是因为这并非是一颗标准的平衡树，相反地，可能有很长的树枝，也可能有很短的树枝。这主要是因为树中的节点可

能在任何时候被添加到这棵树中，为了不影响系统的运行，就允许这种不平衡性存在。但是，对于树中的每个节点而言，在进行查询时具有与 B 树相类似的行为特征，这主要是为了提高查询的效率。在每个系统维护点上，系统会执行必须的平衡操作，例如节点的分裂，旋转（借用 AVL 树的一些特征）等。下面是语义索引操作的一个示例的平衡状态图，见图 5.5。

这里还涉及到了一个要说明的问题，就是关于等价约束链接空间的建立时机，系统选择在新发现一个模板的同时建立一个语之对应的等价连接空间的记录节点，这包括两种情况。

在一个服务刚刚注册到信息仓的时候，如果该服务不能够被归类到某个等价连接空间的时候；

在用户进行查找的时候，用户的输入条件不能够被满足，但是从本体论数据仓当中可以查到该条件的一个超集，同时此超集还满足建立间接等价连接的条件的时候；

下面是语义索引的实现，图 5.6 为应用程序在某个时刻的运行状态，其中的 ELSID 为系统单词的 ID，每个 ServiceID 后的号码为一个注册在系统内的服务的服务号，图中显示了系统中有若干个服务列表入口，有的现在为空，说明与该入口描述的功能相同的服务还没有注册进来；注意在 ELSID8 引导的入口中有 8 个服务，这个服务拥有该系统单词所描述的服务功能，互为等价服务。

图 5.6 语义索引的实时运行图

Figure5.6 the Runtime Status of the Semantic Index

5.2.2 增强 UDDI 的查询功能

在处理用户的查询的时候，与发布的过程很相似，用户的查询请求首先被传递到语义预处理模块（SPT），利用标准信息生成模块将用户的描述信息转化为系统内使用的标准描述，进行第一次消歧；然后，利用系统纪录的实体间的概念上的联系，并参照用户的近似度要求，从而形成一个彼此关联的网状的查询结构。这个步骤主要是为了扩大查寻范围，以实现模糊查询的目的；最后，用户的请求还会被划分成若干个分片，利用这些分片来优化查询。

5.2.2.1 等价连接空间在查询时的使用

语义查询索引中记录的只有服务的 ID 信息以及服务的性能排序信息，所以

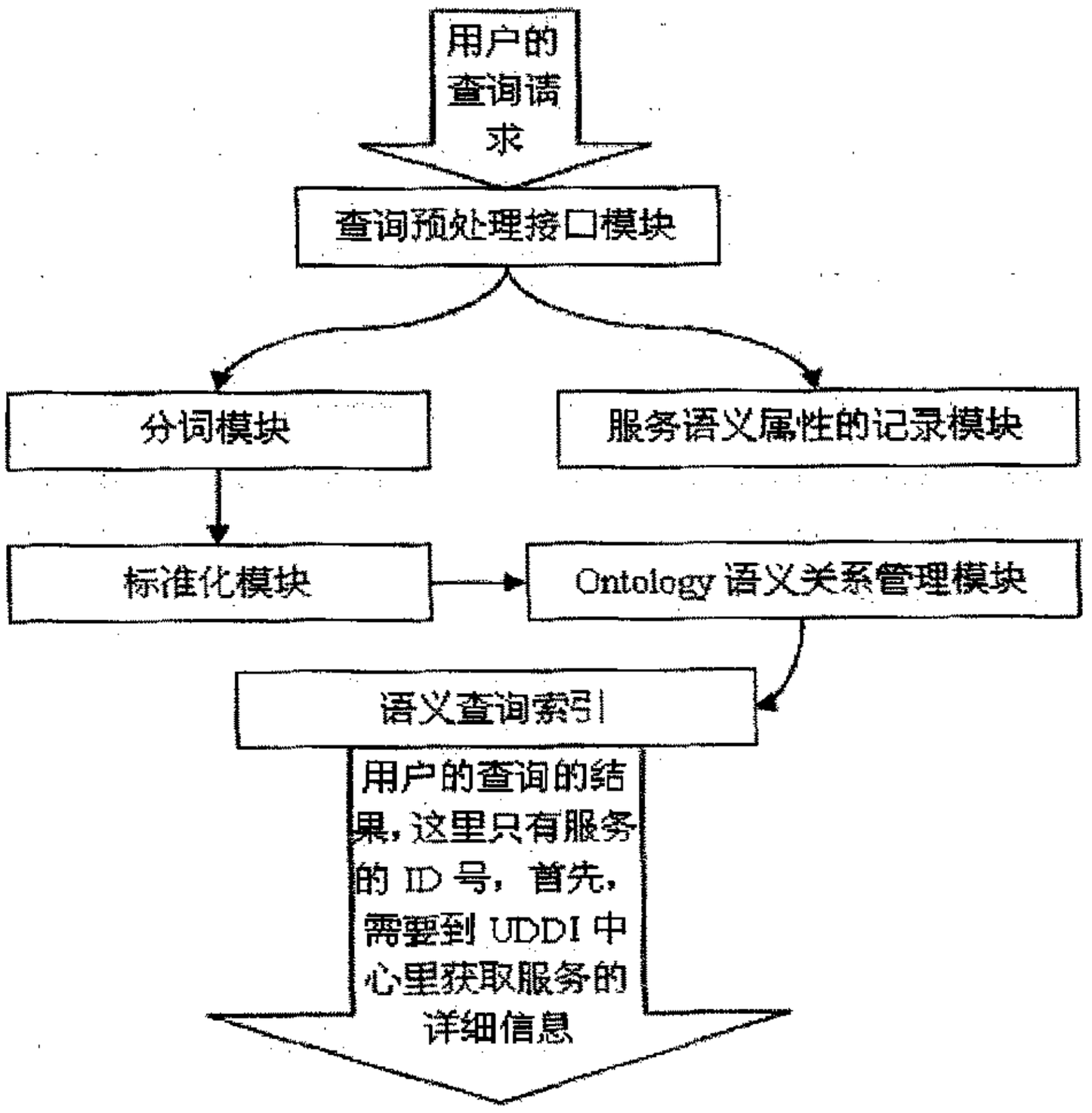

图 5.7 查询当中的语义预处理

Figure5.7 the Semantic Pretreatment in Service Discovery

在这里查询后获得的结果是一个排好序的服务 ID 的列表，还要依据用户的输入条件，结合服务的语义属性信息，进行一定的筛选，然后根据筛选后得结果到 UDDI 中心当中去寻取服务的详细信息返回给用户。

在这个过程里，有几点要说明的事项：

- 如果任何一个存在的模板都不能完整的包含一个查询约束集，则此服务在我们的注册仓内不存在，考虑向服务合成部分求助；
- 这种等价约束连接空间只是针对于服务的某个特定的方面而言的，在查找的时候，先利用等价连接空间缩小目标集，然后再在该集合内部进行进一步的查找，也就是依据用户的输入条件，再进行筛选的过程；
- 等价约束集实际上为我们的查找提供了一个对于约束的划分，所以我们需要进行经常性的维护，以提高其效率。

第六章 结束语

近年来，随着中国加入世界贸易组织，中国经济的繁荣发展，全球化经济成为中国企业面对的现实。为了适应全球的竞争，现代电子商务平台成为各个企业首选的运营方式，电子化带来的是效率的提高和低成本。虚拟企业平台作为电子商务平台的一种，迎合了小型企业为了达到同一目标而结成团体的需要。平台将为企业能在未来经济发展中占有一席之地提供技术和环境支持。

本文以东北大学软件与理论研究所申请的国家高科技发展计划（863 计划）项目"基于 ASP 模式的支持中小企业动态联盟的使能器服务系统"为背景，参考了国内外大量的相关文献，研究探讨了以 B/S 三层构架为基础的、支持虚拟企业商务合作的系统平台。 其中，如何获得有效的共享资源，为虚拟企业提供最佳的合作环境至关重要。本文基于 UDDI， 提出了基于社区的 Web 服务语义化的服务发现思想，主要完成了以下工作：

针对现有的查询方案的不足，提出新的应用方案，集中研究了语义化查询的优点。

利用基于语义索引的分域处理的思想设计实现了整个系统，系统采用模块化组件，除了必要的基本模块，其他模块都可以动态插入到系统中，提高了系统的可重用性和可扩展性。

服务调用模块采用国际标准的协议，提供 Web 浏览访问方式和接口访问方式。接口方式以 HTTP 协议为基础，利用 SOAP 协议进行远程调用。系统内服务和第三方服务全部注册在私有 UDDI 中心，方便用户查询。设计了代理调用系统，方便系统对全局服务调用的控制。

目前，Web 服务领域的研究还在继续，关于 Web 服务的发布与查询也在同步地进行。文中设计的基于语义信息的分域处理、建立语义化索引等模块的设计与实现还有待于在实践中不断修改、完善。分布式对象合作模型将提供更高级的代理调用系统以适应其他模块的需求。在不久的将来，我相信本套系统将会更加完善，更加满足用户的需求。

参考文献

[BE02] T. Bellwood, D. Ehnebuske, etc. UDDI Version 2.03 Data Structure ReferenceUDDI Committee Specification, 19 July 2002, http://uddi.org/pubs/DataStructure-V2.03.Published-020719.htm , 2002

[ABC02] R. Atkinson, R. Brendle, P. Conley,etc. UDDI Version 2.03 Replication Specification UDDI Committee Specification, 19 July 02, http://uddi.org/pubs/Replication-V2.03.Published-020719.htm , 2002

[BDE02] D. Bryan, V. Draluk, D. Ehnebuske, etc. UDDI Version 2.04 API Specification UDDI Committee Specification, 19 July 02, http://www.uddi.org/pubs/ProgrammersAPI-v2.00-Open-010608.pdf, 2002

[BPM00] T. Bray, J. Paoli, C. M. Sperberg McQueen, etc. W3C (World Wide Web Consortium) Extensible Markup Language (XML) 1.0, October 00. http://www.w3.org/TR/00/REC-xml-001006, 2000

[GHM03] M. Gudgin, M. Hadley, N. Mendelsohn, etc. SOAP Version 1.2 Part 1: Messaging FrameworkW3C Recommendation 24 June 03, http://www.w3c.org/TR/SOAP, 2003

[CCM01] E. Christensen, F. Curbera, G. Meredith, etc. Services Description Language (WSDL) 1.1W3C Note 15 March 01,http://www.w3.org/TR/01/NOTE-wsdl-010315, 2001

[蒋01] 蒋松，白利强. **Web** 服务——下一代的 **WWW** http://www.tech521.com/techData/data/1254.asp, 2001

[柴01] 柴晓路. 架构 Web 服务：为什么需要 Web 服务？, http://www.huihoo.com/xml/dw/webservice1.html, 2001

[孟01] 孟军，王宝学. 精通 ASP.NET 网络编程，人民邮电出版社，2001

[Dra01] V. Draluk, D. Web Service: an Overview, Proc. of the 27th VLDB Conf., Roma, Italy, 637-640 http://www.vldb.org/conf/2001/P637.pdf, 2001

[GH00] E. Gamman , R. Helm , Design Patterns , 机械工业出版社，2000

[IWS01] IBM Web 服务 Architechture Team. Web 服务 Architecture Overview: The Next Stage of Evolution for e-Business, http://www-106.ibm.com/developerworks/web/library/w-ovr/, 2001

[孙 01] 孙三才，许薰尹. 精通 C# 与 ASP.NET 程序设计，中国青年出版社，2001

[Baa01] Mark Baartse. ASP 与 XML 高级编程，清华大学出版社，2001

[申 02] 申德荣，于戈等. 基于 ASP 模式的支持中小企业动态联盟使能系统的研究[J]. 计算机科学, 2002.10

[战 02] 战立明，申德荣等. 基于 UDDI 规范的分布式对象共享模型的研究[J]. 计算机科学,2002.29(8.增.B)

[陈 02] 陈天，战立明等. 虚拟企业内基于 UDDI 的分布式对象共享管理模型[J]，小型微型计算机系统，2002

[McF1998] C. McFall. An Object Infrastructure for Internet Middleware, IBM on Component Broker. IEEE Internet Computing, 1998;2(2)

[鲁 03] 鲁男,申德荣,于戈.基于语义的 Web 服务社区查找方法的研究，网络化制造与大规模定制学术会议论文集，589-595，2003

[PKR02] M. Paolucci, T. Kawamura, T. R Rayne, etc.Semantic Matching of Web Servic Capabilities. http://www.daml.org/services/ ISWC02-Matchmaker.pdf，2002

[JH01] A. Jagatheesan and S. Helal. Sangam: Universal Interop Protocols for E-Service Brokering Communities Using Private UDDI Nodes. http://www.harris.cise.ufl.edu/projects/publications/ Sangam.pdf，2001

[JM02] K. Januszewski, E. Mooney,. etc. UDDI Version 3 Features List. http://www.uddi.org/pubs/uddi_v3_features.htm

[HWJ01] S. Helal, M. Wang, A. Jagatheesan, etc. Brokering Based Self Organizing E-Service Communities， http://csdl.computer.org/dl/proceedings/isads/01/1065/00/10650349.pdf

[PPS02] T. R. Payne, M. Paolussi, R. Singh, etc. Communicating Agents in Open Multi Agent Systems http://www.daml.org/services/ WRAC02.pdf

[Cam02] C. Campo. Directory Facilitator and Service Discovery http://www.fipa.org/docs/input/f-in-00070/f-in-00070.pdf

[NCW00] W. Nagy, F. Curbera, and S. Weerawarana, The Advertisement and Discovery of Services (ADS) protocol for Web Services Simplifying the announcement of available Web Services to inquiring software agents http://www-900.ibm.com/developerWorks/cn/web/ws-ads/index_eng.shtml

[HMJ00] S. Helal, M. Wang and A. Jagatheesan , SERVICE-CENTRIC

BROKERING IN DYNAMIC E-BUSINESS AGENT COMMUNITIES

http://www.harris.cise.ufl.edu/projects/publications /BPtemp148.pdf

致 谢

两年半的硕士研究生的学习生活马上要结束了，怀着对未来生活的美好憧憬我就要离开朝夕相处的老师和同学了。回首这两年半时光，我不禁对研究所里浓厚的学习气氛，良好的研究环境，无私奉献的老师和团结友爱的同学产生了深深的眷恋。我深深庆幸自己能有机会来到这个朝气蓬勃的集体。

在即将毕业之际，我对各位老师和同学们的无私的关心和帮助表示衷心的感谢！特别要感谢我的导师于戈教授和申德荣副教授给予我的无私的教诲与帮助。

于戈教授对我的悉心教导和谆谆教诲，使我终生难忘；他渊博的知识，敏捷的思维，一丝不苟的治学态度给我留下了深刻的印象，将对我今后的学习和工作产生深远的影响。

申德荣副教授在学习和研究上也给予了我尽可能多的指导和帮助。她严谨的治学态度，精益求精的工作原则和废寝忘食的工作热情为我树立了榜样，不断的激励我孜孜不倦地追求完美。

实验室的其他老师和同学也在工作和生活中给了我极大的帮助。特别感谢战立明、陈天、杨丹、张荣、吴青泉、聂铁铮等同学在我遇到困难时伸出了援助的双手。

最后，衷心祝福东北大学软件与理论研究所越走越好，我将永远为你而自豪！

攻研期间论文发表情况

1．鲁 男,申德荣,于 戈, 聂铁铮，陈 天. 基于语义的 Web 服务社区查找方法的研究,第一届网络化制造与大规模定制学术会议论文集，2003,10,pp589-595

2．战立明、申德荣、鲁男、于戈，基于 UDDI 规范的分布式对象共享模型的研究，计算机科学(增刊)2002，Vol.29(8)，pp145.147

3．张 蓉，申德荣，于 戈, 鲁 男等，基于.Ontology 的 Web 服务查找和合成技术研究，计算机集成制造系统—CIMS,2003,Vol.9(10)pp921-925

攻研期间科研情况

国家 863 计划项目“基于 ASP 模式支持企业动态联盟使能系统的研究”（项目编号：2001AA415210） 2001.10-2003.10，50 万元，项目主要参与人。