摘要

随着信息技术的不断进步，信息网络化和社会化进程的加快，信息化工程取得了一定的成绩，但是，在信息化的道路上也出现了一些新的问题，提出了新的需求和挑战。新的问题就是信息不能共享和整合，分布式资源不能有效利用。本文在对此问题出现的原因进行分析的基础上提出了将网格技术和本体论有机的结合，构建一个统一的基于语义的信息系统集成和互操作信息平台，实现信息共享，加快信息化的步伐。

本文讨论了一个以劳动力市场背景的支持数据集成的语义网格系统LF-Grid。文中首先根据系统的实际需求，给出了LF-Grid系统的整体结构框架，简述了各个组成模块的功能，具体讨论了此系统中基于语义本体的服务发现机制的相关的三个问题。

(1)网格服务语义描述的研究

主要研究如何定义和表示网格服务。服务描述旨在为服务提供者和服务请求者提供标准的方式来描述，是服务发现的基础。基于服务发现的需求，本文在分析了各服务描述语言的基础上，参考Web服务本体论OWL-S，根据WSDL 1.2版本的可扩展性，提出了在服务描述WSDL文档，扩展服务功能性信息的语义描述，为服务发现奠定良好的基础。

(2)UDDI语义扩展的研究

为了实现扩展后的服务描述的有效发布，本文利用UDDI的数据实体tModel的特点，通过把服务描述的语义信息以tModel的形式注册到UDDI中心，实现扩展服务描述在UDDI中心的使用。为更好地实现高效、准确的服务发现提供了基础。

(3)网格服务语义发现的研究

根据智能化和高效性的要求，研究怎样基于语义描述和本体论对广告服务描述与请求服务描述进行服务发现。服务的匹配是服务发现的一个关键问题。目前，为了提高服务发现过程中服务匹配的能力，许多方法都考虑有效利用本体论技术，对服务进行语义匹配。本文在扩展服务描述的基础上，提出了语义匹配原理及LF-MCH算法，对其做了详细的阐述。LF-MCH该匹配方法从本体概念的语言特征和背景特征两方面进行匹配，计算语义相似度。接着基于这个匹配算法，提出了LF-Grid 网格服务发现三阶段算法，充分利用了网格服务中存在的潜在的语义，

通过对服务功能的操作、输出和输入参数、前提条件和效果递进地进行语义相似匹配，很大程度上提高了服务检索的查准率和查全率。当存在大量功能相似、输入和输出相同的服务时，通过前提条件和效果的语义匹配筛选，实现最佳服务的发现。

最后，总结了全文并对系统的进一步完善提出了建议。

关键词：语义发现；语义描述；语义注册；语义网格；网格

ABSTRACT

With the increasing development of information technology and the quickening of the information networking and social process, the information project has made a certain improvement. However, some new problems, requirements and challenge appear on the information way. The new problem is that information can not share and be integrated, and the distributed resources can not be made use effectively. On the base of the analysis of the reasons for it, this paper puts forward that Integrating grid technology and ontology, and building a uniformed semantics based information system integrating and interoperating platform realize information sharing, and quicken the step of information.

In this paper, we discuss a semantic grid system LF-Grid to support the data integration on the background of labor force market. First, on the basis of the actual requirement of system, we give the architecture of LF-Grid, and describe the functionality of all modules. We detailedly discuss the relevant three problems about the ontology-based service discovery.

(1) research on semantic description of grid service

This section discusses How to define and express grid service. The purpose of service description is to provide a standard way to describe for the service provider and the service request, and it serves the discovery foundation. Based on service discovery demand, This article first analyzes each service description language ,and then refers to Web service ontology OWL-S, and according to the WSDL 2.0 editions may the extension, Proposes to extend the semantic description of service functional information in the WSDL document, in order to serve the discovery to lay the good foundation.

(2) research on extended semantics of UDDI

In order to realize the extended service description to effectively issue, This paper uses the UDDI's data entity tModel characteristic, Through registering the service description semantic information by the tModel form to the UDDI center, to realize the use of extended service description in UDDI center, so as to provide the foundation to realize highly effective, accurate serves the discovery.

(3) research on the semantic discovery of grid service

According to the requirement of intellectualization and highly effective, this section mainly studies How to carry on the service discovery based on the advertising

service description and the requested service description with the semantic description and the ontology. The service match is a key question of discovery. At present, In order to enhance the match ability in the service discovery process, many methods all consider to effectively use the ontology technology, and carry on the semantic match of the services. Based on extended service description, this paper proposes the semantic match principle and the LF-MCH algorithm and details it. Then, based on this match algorithm, It proposes three stages algorithms of LF-Grid grid service discovery, which Fully has used the latent semantics which in the grid service exists.

At last, we conclude the whole paper and put forward the planning of the next work.

Keywords: semantic discovery; semantic description; semantic registry; semantic grid; grid

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。本声明的法律责任由本人承担。

论文作者签名：李露　　日　期：2006.4.5

关于学位论文使用授权的声明

本人完全了解山东大学有关保留、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阅；本人授权山东大学可以将本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。

(保密论文在解密后应遵守此规定)

论文作者签名：李露　导师签名：　日　期：2006.4.5

第一章 绪论

1.1. 研究背景

随着信息技术的不断进步，信息网络化和社会化进程的加快，信息化工程取得了一定的成绩，实现了信息的对外公开。但是，在信息化的道路上也出现了一些新的问题，提出了新的需求和挑战。新的问题就是信息不能共享和整合，分布式资源不能有效利用。产生这个问题的原因主要有以下两方面：

一方面，目前信息化程度只是局限在很小的范围内，大多数信息平台中信息的组织和利用基本上还是在相对孤立的系统内进行的。性质或功能相似的一些信息平台由于数据结构、概念模型和软硬件环境的差别形成异构分布的信息孤岛，极大地阻碍了信息共享和应用。

另一方面，由于不同地区的人们对领域概念的认知不同和表述习惯的差异，导致对同一领域概念描述不一致，使描述具有地区局限性，从而产生描述上的差异，形成语义异构。因此不同地区的信息平台对同一个概念的语义解释往往有很大的差别，如果不考虑这种语义的差异，就会严重影响查询的准确率，不能满足用户的需求。信息平台存在的价值也就得不到充分的体现。

鉴于对原因的分析得知，解决信息共享问题，消除信息孤岛和知识孤岛，首先要定义统一开放的领域数据模型和服务模型，来解决平台异构性；而对于语义异构性，则需要采用基于语义的描述方式来解决。将这两种方法有机的结合，提供一个统一的基于语义的信息系统集成和互操作信息平台，实现信息共享，加快信息化的步伐。

在提供统一开放的领域数据模型和服务模型方面，网格技术起着非同寻常的作用。所谓网格[1]是构筑在Internet上的一项新兴技术，它把整个Internet整合成一台巨大的超级计算机，实现计算资源、存储资源、数据资源、信息资源、知识资源、专家资源的全面共享。网格的根本特征并不是它的规模，而是资源共享，消除资源孤岛。这种开放透明的共享资源模式决定了网格与信息平台结合的必然性。信息平台的研究重点是如何消除信息孤岛和知识孤岛，实现信息资源和知识

资源透明地共享。这种共享不是一般的文件交换与信息浏览，而是要把同一领域的信息平台连接成一个虚拟的社会组织(virtual organization)，实现在动态变化环境中灵活控制的协作式信息资源共享。基于网格的信息平台与基于web的信息平台最大的区别是一体化，即用户看到的不是数不清的门类繁多的网站，而是单一的入口和单一系统映像。一个用户需要查询某一方面的信息数据，不必知道有哪些数据供应商或数据生产者，只需通过信息网格提供的元数据信息库进行最简单的查询，即可找到用户需要的信息数据。数据的查询检索对用户是透明的，只要查询请求的格式符合系统要求，经过网格计算，就可以从基于网格的信息平台上轻松获取所需要的数据。就是采用网格技术解决信息共享问题，由此构造一个基于语义的信息网格平台。

在解决语义异构的方面，本体概念的引入也是最普遍，最重要的一种方式。目前业界比较认同的定义是1995由Gruber[T.Gruber, 1995]提出的：本体（Ontology）是共享概念模型的明确的形式化规范说明。本体的目标是捕获相关的领域的知识，提供对该领域知识的共同理解，确定该领域内共同认可的词汇，并从不同层次的形式化模式上给出这些词汇(术语)和词汇之间相互关系的明确定义。我们将本体的概念引入主要是为了实现信息服务的基于语义自动发现、定位、合成和替换，并进行有效的解决语义异构。

本文首先在对网格技术和本体概念研究的基础上，以劳动力市场为背景提出了一个基于OGSA的语义网格系统——LF-Grid，来有效地解决上述问题，实现分布式资源的有效利用和整合。该语义网格模型就是在利用现有的网格基础设施、在网格的协议规范的基础上，引入描述语义的本体系统，为用户提供基于语义的一体化应用服务的虚拟信息平台。在这个平台上，信息处理是基于语义和分布式协作的，用户可以在语义网格的语义转换层透明地对所有集成到语义网格的信息资源实现基于语义的访问服务。语义网格追求的最终目标是把Internet上的信息服务站点在语义转换层连接起来，实现基于语义的信息共享和分布式资源的整合。

1.2. 研究的课题及意义

服务发现机制是一项关系到广域分布式环境中资源共享和协同工作效率的关键技术，同样，它在网格环境中也是具有非常重要的地位。本文研究的课题是网

格环境下服务发现机制。服务发现是把服务和服务请求者联系起来的重要环节。随着信息量的不断增多，网格中的服务种类繁多，功能各异，网格应该为用户提供一种功能，能够根据用户的请求从网格服务中找到满足用户请求的资源。服务发现将网格中不被用户所知道的服务和请求使用服务的用户联系起来。服务发现功能的强弱，直接决定了网格的使用效率和友好程度。

随着服务种类的不断增多，服务规模的不断扩大，用户能够在这么繁多的服务中找到适合自己需要的服务成为一个新的挑战。如何让用户能既快又准地找到自己所需服务需要进一步研究。

目前在网格服务的表达和检索方面，仍然存在着许多技术缺陷，制约着Web服务的准确、高效的发现。主要体现在以下一些方面：

(1)以语法性语言表达的网格服务，主要是面向用户直接阅读的，不利于计算机直接阅读和处理；

(2)不同团体对同一领域事物的认识和表示往往不同，使得来自服务提供者与服务请求者关于同一网格服务的描述存在着冲突，这种认识上的差异所产生的描述差异可被称作语义异构，具体表现在:①不同的服务描述使用多种术语(词汇)表示同一概念；②同一概念在不同的服务描述中表达不同的含义；③各服务描述使用不同的结构来表示相同(或相似)的信息；

(3)以关键字匹配的方式为主的检索，根据广告服务描述中是否包含请求查询中的关键词来返回结果，由于许多不相关的服务也会在它的描述中包含查询关键词，检索的结果往往会出现很多不相关的网格服务，随着服务数量的增大，检索的准确率就越低。同时这种关键字匹配的方法，查询关键词与广告服务描述中的关键词可能是语义相同但是非语法相同的，遗漏了大量与检索概念同义或相关的内容信息，因此检索在查全率方面不高，难以达到期望效果；

(4)服务的检索只是对服务功能描述的关键词匹配，无法充分反映服务所提供服务的功能信息，造成服务检索结果不理想；

本文在了解服务发现的重要性和分析制约服务发现的原因的基础上，提出一种在网格环境下简单、实用、可实现服务准确匹配的语义发现机制，解决上述服务发现的缺陷，帮助用户既快速又准确地找到自己所需服务。

本文的网格服务语义发现机制借鉴了Web服务[2]的经验。在试图解决Web服务发

现面临的问题过程中，业界呈现出了许多有关服务发现的研究。最早为发现 Web 服务而增加语义信息的是DAML-S[3]，DAML-S有一个上层本体，这个本体描述Web服务时使用三种类型知识——ServiceProfile，ServiceModel和ServiceGrounding。ServiceProfile描述Web服务做什么，ServiceModel描述Web服务怎么工作，ServiceGrounding用来定义如何访问Web服务。OWL-S[4][5]是在DAML-S的基础上发展起来的。DAML-S采用DAML+OIL描述Web服务，提供了足够的表示Web服务能力和特性的语义信息，目的是实现自动的Web服务发现、调用、合成和执行监控。

目前，许多研究都是基于DAML-S展开的，如语义表示的研究[6]，服务绑定的研究[7]，基于本体的服务匹配[8]的研究，DAML-S和UDDI相结合的研究[9,10]等。其中将DAML-S与UDDI相结合，补充UDDI的Web服务表示语义，增强服务发现能力，是服务最佳发现的一种趋势。在当前存在的研究中，最主要的形式是:实现DAML-S本体描述信息到UDDI的映射，补充UDDI的Web服务语义描述能力[9,10]； DAML-S服务概要信息全面地描述了服务，是服务发现的依据。

在[9,10]中，通过将DAML-S概要信息映射到UDDI注册仓中，来补充Web服务描述语义。具体实现方法是:

(1)DAML-S概要信息中可以直接映射到UDDI描述信息的属性，采用直接映射的方法实现，例如:服务提供者的信息如名称(name)、电话(phone)、地址(phsicalAddress)、电子邮件(e-mail)等；服务信息如服务名(serviceName)。描述(description)等。

(2)其它属性定义为相应的tModel；注册到UDDI的businessService中的CategoryBag中，如将serviceType定义为serviceTypeTModel，serviceCategory定义为serviceCategoryTModel，qualityRating 定义为qualityRatingTModel，input定义为inputTModel等。在服务发现中，采用服务匹配算法(输入、输出匹配算法和输入、输出匹配规则算法)，匹配发布的服务描述和请求的服务描述。

[11]描述了一个匹配引擎来匹配请求服务和广告服务。它提供了一个语义算法来对请求服务的输入和输出参数和广告服务的相应参数进行匹配。它在UDDI上增加了一个映射层，使用DAML-S作为服务描述语言，与基于关键字搜索相比，提供更好的服务发现。但是，它没有使用当今Web服务的工业标准，而是建立一种新的设施，增加了复杂性。[11]只是使用输入、输出参数搜索所需服务，没有对操

作进行功能描述。本文采用本体描述各个参数，并且增加了其它细节来确保服务匹配准确的需求。

近来有大量的研究集中在服务的有效发现上，服务发现是 Web 服务框架的关键必需的性能。改进基于关键字发现的发现机制范围很广，从服务与分类和领域无关的特性[12]到开发服务语义描述的技术。[13]分析了匹配的问题，着重强调了需要元数据描述来获取更好的结果，建议高级匹配必需具备更大的灵活度和表达能力；表示半结构数据的能力；对类型和包含的支持，对数据取值范围限制的能力。同时它发现 UDDI 没有较强的表达能力和灵活性。

本文在分析制约服务发现的原因和了解web服务发现现状的基础上，提出了一种在网格环境下简单、实用、可实现服务准确匹配的语义发现机制，解决上述服务发现的缺陷，帮助用户既快速又准确地找到自己所需服务。其中语义描述方面，本文采用将OWL本体扩展到GWSDL模式的方法补充网格服务的语义描述信息，其扩展的信息类似于DAML-S中的ServiceGrounding描述，强调的是从网格服务描述的多个操作中发现恰当的操作。

1.3. 论文解决的主要问题

随着网格技术的迅速发展，网格服务的潜在提供者是整个网格上的所有网格服务提供者，可供选择的候选服务数量巨大，同时，它具有高度的自治性，而且其应用环境也是异类的。这些特性导致了一系列问题，如网格服务的描述、网格服务的匹配等等。为了有效地实现网格服务的应用价值需要解决如下关键问题:如何发现满足需要的服务；如何选择最佳服务等。以上问题的解决，都将为最终实现网格服务的潜能提供先进的、便利的和可靠的技术基础。本文融合了已有服务发现策略的优势，提出了一种简单、实用、可实现服务准确匹配的服务发现策略。基本思想:(1)所有服务基于网格服务本体定义，实现对服务的语义描述，保证发布的服务满足相同的描述规范；(2) 基于UDDI规范扩展服务描述语义信息，将扩展的语义信息以tModel的形式存储在UDDI中心；(3)在服务发现过程中，基于服务本体描述，应用本体匹配度算法获得满足服务请求的网格服务；为此，本文针对如下几个关键问题展开研究:

(1) 网格服务语义描述的研究

主要研究如何定义和表示网格服务。服务描述旨在为服务提供者和服务请求者提供标准的方式来描述，是服务发现的基础。基于服务发现的需求，本文在分析了各服务描述语言的基础上，参考Web服务本体论OWL-S，根据WSDL 2.0版本的可扩展性，提出了在服务描述GWSDL文档，扩展服务功能性信息的语义描述，为服务发现奠定良好的基础。

(2)UDDI语义扩展的研究

为了实现扩展后的服务描述的有效发布，本文利用UDDI的数据实体tModel的特点，通过把服务描述的语义信息以tModel的形式注册到UDDI中心，实现扩展服务描述在UDDI中心的使用。为更好地实现高效、准确的服务发现提供了基础。

(3)网格服务语义发现的研究

根据智能化和高效性的要求，研究怎样基于语义描述和本体论对广告服务描述与请求服务描述进行服务发现。服务的匹配是服务发现的一个关键问题。目前，为了提高服务发现过程中服务匹配的能力，许多方法都考虑有效利用本体论技术，对服务进行语义匹配。本文在扩展服务描述的基础上，提出了语义匹配原理及LF-MCH算法，对其做了详细的阐述。LF-MCH该匹配方法从本体概念的语言特征和背景特征两方面进行匹配，计算语义相似度。接着基于这个匹配算法，提出了LF-Grid 网格服务发现三阶段算法，充分利用了网格服务中存在的潜在的语义，通过对服务功能的操作、输出和输入参数、前提条件和效果递进地进行语义相似匹配，很大程度上提高了服务检索的查准率和查全率。当存在大量功能相似、输入和输出相同的服务时，通过前提条件和效果的语义匹配筛选，实现最佳服务的发现。

1.4. 论文的组织结构

本文其余内容的结构和组织如下:

第二章首先介绍了网格的相关知识，接着对LF-Grid网格平台的总体框架及其组件进行了详述。

第三章主要研究网格服务描述问题，服务描述旨在为服务提供者和服务请求者提供标准的方式来描述，是服务发现的基础。针对目前服务描述在内容和语义信息两方面存在的不足，参考Web服务本体论OWL-S，根据WSDL2.0版本的可扩展性，

提出了使用OWL本体对服务描述GWSDL文档进行语义扩展，实现对服务描述的语义清晰化，为服务语义发现奠定良好的基础。

第四章在分析了现有网格服务的注册系统存在弊端的基础上提出在网格环境中引入UDDI注册数据模型。基于UDDI规范对服务描述语义信息进行扩展，将扩展的语义信息以tModel的形式存储在UDDI中心。

第五章主要介绍网格服务语义发现的问题。服务匹配是服务发现的关键问题，首先提出了语义匹配原理及LF-MCH算法；基于这个匹配算法，接着详述了LF-Grid网格服务发现三阶段算法。

第六章对本文进行总结，并指出进一步的工作。

第二章 LF-Grid网格平台的总体框架

随着劳动就业、保障业务的发展和信息技术的不断进步，国内的劳动力市场信息化工程早已起步，但是，劳动力市场信息系统中信息的组织和利用基本上是在相对孤立的系统内进行的。不同的劳动力市场信息系统由于数据结构、概念模型和软硬件环境的差别形成异构分布的信息孤岛，极大地阻碍了信息共享和应用。大多数地区社会保险管理、劳动就业管理、劳动保障综合管理等部门之间的业务关联错综复杂，但由于部门之间的信息不能共享，造成这些部门之间的业务配合和支持也相应存在着很大的障碍。在劳动力市场领域，业务的服务主体为单位和个人，一个服务主体所涉及业务的处理随服务主体的意愿、状态属性、特定条件等因素的制约可能仅涉及一个数据资源，也可能涉及到多个数据资源，而目前的信息系统还不能对资源做有效的整合，不能完全满足业务的需要。

本文提出了一个基于OGSA的语义网格架构——LF-Grid，来有效地解决上述问题，实现分布式资源的有效利用和整合。LF-Grid是一个面向服务的语义网格[14]的实现，它按照OGSA的简单运行环境构建了整个网格体系，补充了许多网格运行所需的组件，并且从语义角度对整个框架进行了扩展。

2.1. 网格的基本体系结构

2.1.1. 网格的系统组成和层次结构

2.1.1.1. 网格的系统组成

网格系统一般可以分为三个基本层次:底层资源层、中间件层和应用层。

网格资源层是包含劳动力市场上可以通过Internet访问到的所有分布式资源:计算机、贵重仪器、可视化设备、现有应用软件、工作站、机群系统、存储设备、数据库等这些逻辑上孤立的单个资源或资源群。基本功能控制局部资源和向上提供访问这些资源的接口。

网格中间件层是来完成广域计算资源(网格资源层所涉及到的各种资源)的有效共享，它是指一系列的工具和协议软件，其功能是屏蔽网格资源层中计算资源的分布、异构特性，向网格应用层提供透明、一致的使用接口。它是网格系统中的操作系统，同时提供用户编程接口和相应的环境，以支持网格应用的开发。网格中间件是网格技术的核心部分，包含了网格技术的关键技术和大部分内容。网格中间件在不同的环境和应用中被分为若干部分，例如，Globus体系中分为连接层、资源层、汇聚层，有时也被分为核心中间件、用户中间件。

应用层是在虚拟组织环境中存在的，是网格应用的入口，提供Web使能的应用服务，即用户可以通过Web接口实现相关任务到远程资源的提交及结果的收集，网格用户可以使用其提供的工具或环境开发各种应用系统。它是用户需求的具体体现。

2.1.1.2. 网格的层次结构

对于软件开发人员来说，用如图2-1所示的层次结构来理解网格更加容易它是从网格开发的角度体现网格的层次结构。

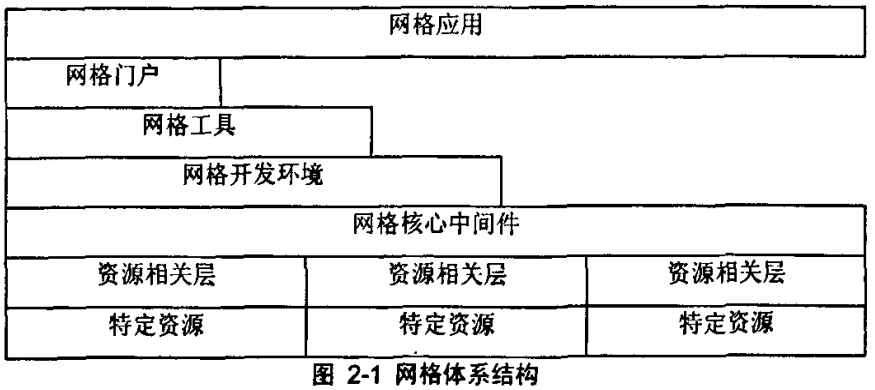

图 2-1 网格体系结构

这一结构中，最底层就是各种可以享用的网格资源，它可以包括极其广泛的内容，从超大型的仪器设备到一个简单的探测器都可以成为共享的目标。实现 各种 资源广泛共享，需要提供一层与特定的资源相关的功能，它既可以实现对该资源的有效控制，又能对上面的需求提供统一的接口，便于对资源的访问。这就是资源相关层。

网格中最关键的一层是网格核心中间件[15]，这一层软件设施能够对分布的各种资源进行有效的管理，为整个网格应用提供高效、安全、可靠的服务。它是网

格系统中连接上层应用和下层资源的纽带。网格开发环境是为了方便使用网格的各功能而提出的一种集成的、高效的网格应用技术人员工作环境，使用这一环境，可以更容易的实现各种网格功能。

网格工具层是将经常使用网格功能进一步以网格工具的形式固定下来，即使是非专业人员也可以通过网格工具对网格进行基本管理。

网格门户是为最终的网格用户提供的，是用来访问网格服务与资源的可定制、个性化Web接口。网格用户可以针对不同网格用户的特点，给他们提供熟悉的容易理解和方便操作的界面。

最上面一层是各种各样的网格应用，也就是网格最终服务的对象，它使网格真正成为一种重要的基础设施。

2.1.2. 开放网格服务体系结构

Globus小组和IBM 于2002年初提出了一个新的网格结构，该结构是要将当时网格领域最热门的两个技术——计算网格和Web服务结合起来，把原来按照两条路线进行的研究活动归纳到一条主线上来。开放网格服务体系结构OGSA (Open Grid Service Architecture)是一个面向服务的系统结构，主要突出从网格用户的角度看上去的网格系统是什么样子。

2.1.2.1. OGSA

OGSA [16]的主要思想是定义网格系统的框架、体系结构和功能性。OGSA以服务为中心，把一切都抽象为服务，服务既包括计算机设备、应用程序、数据，也包括仪器、设备等。将一切都抽象为服务有利于通过统一的标准接口来管理和共享网格上功能各异的资源。在前面的介绍中可以看到，Web服务提供的是一种永久性的无状态服务。OGSA在原来Web服务的基础上，提出了网格服务的概念，用于解决服务发现、动态服务创建、服务生命周期管理等与临时服务有关的问题。

OGSA定义了如图2-2所示的服务[17]。

接口	操作	描述
GridService	FindServiceData	查询网格服务实例的各种信息

	SetTerminationTime	设置并得到网格服务实例的终止时间
	Destroy	终止网格服务实例
NotificationSource	SubscribeToNotificationTopic	向通知发送者进行登记
	UnSubscribeToNotificationTopic	取消登记
NotificationSink	DeliverNotification	异步发送消息
Registry	RegisterService	网格服务句柄的软状态注册
	UnRegisterService	取消注册的网格服务句柄
Factory	CreateService	创建新的网格服务实例
PrimaryKey	FindByPrimaryKey	返回根据特定键值创建的网格服务句柄
	DestroyByPrimaryKey	撤销特定键值创建的网格服务实例
HandleMap	FindByHandle	返回与网格服务句柄相联系的网格服务实例

图 2-2 OGSA 定义的接口

网格服务的框架如图2-3所示。工厂服务将自己发布到注册中心后，网格用户就可以查询到自己需要的工厂服务。用户向工厂服务发送服务创建请求，工厂服务创建服务实例，并将所创建的服务实例的网格服务句柄(Grid Service Handle. GSH)返回给用户。用户获得GSH后，并不清楚服务的调用方式，需要从句柄影射服务中得到服务的网格服务引用(Grid Service Reference，GSR) 并根据GSR生成服务器调用程序。

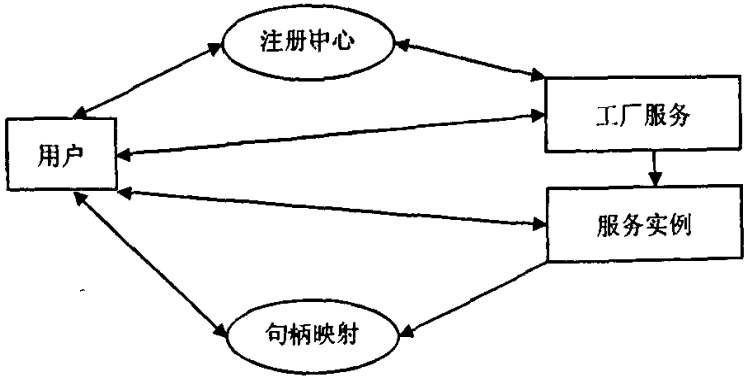

图 2-3 网格服务框架

OGSA 涉及到以下方面:

1. 互操作

OGSA 框架使用面向服务的体系结构，通过使用WSDL将按口定义和协议绑定分离开来实现服务的无缝集成。统一的服务接口定义(服务语义)包含了一个抽象层。隐藏了特定于平台的实现。这增强了服务虚拟化的概念并增加了虚拟组织的灵活性。服务质量、认证、授权和信任代理由协议绑定的属性处理。这种实现允许不同的协议绑定到接口上，同时，网格服务接口的抽象性也允许不同的实现相互集成。

2. 资源的发现和访问

资源的发现和访问包含三个方面:

(1) 服务数据的标准表示法。这种服务数据用XML来描述，包含了有关网格服务实例的信息。

(2) 标准操作——FindServiceData， 用于从单个的网格服务实例中获取服务数据。

(3) 标准的接口——注册器，用于注册网格服务实例的信息。

3. 资源的临时生命周期管理

在动态环境中，服务在需要时被创建，在不再需要时被销毁。OGSA使用软状态来实现生命周期管理。网格服务在创建时都赋予了一个特定的初始生存期，并可以被客户服务或别的服务发送的“Keep alive”消息来显式的扩展。当客户不再需要该服务时，就停止发送“Keep alive”消息。或者当消息传送路径上的某个环节崩溃时，服务也收不到“Keep alive”消息。在所设定的生存期到期后，服务提供者的运行环境或者服务本身可以决定销毁服务以释放资源。

网格服务接口有一个操作SetTerminationTime来设置初始时间。

另外，客户也可以通过一些操作在创建服务时来协商服务的生存时间，这些操作允许客户指定一个最大和最小的可接受时间。如果服务提供者同意客户的请求，并能在所指定的时间内提供服务，就会为客户创建服务。

4.服务状态

网格服务的状态在生命周期中会发生改变，这就需要一些过程来对状态进行管理。OGSA定义服务数据元素(SDE)来存储和维护这些服务状态。这些服务状态可

以通过OGSA 定义的服务接口来访问。状态的改变可以通过异步的状态变化通知来通知相关的服务。客户服务向提供者服务请求其所关注的状态变化。同时，当服务状态变化时，提供者服务负责通知所有的客户。

在网格体系结构中，服务的定义和实际服务的实例是分离的。一个单一的接口可以同时拥有多个服务实例，服务于多个客户服务。OGSA的注册器维护服务定义到服务实例之间的映射。

5. 工厂

OGSA 定义了一个称为工厂的接口来创建新的网格实例。一个工厂从客户服务处接收请求，并在成功创建服务之后返回该实例的网格服务句柄GSH和初始的网格服务引用GSR。在OGSA中，工厂服务是分级的，高级的工厂服务将其工作委托给一个或多个低层的工厂。

6. 从永久句柄中动态解析临时引用

一个服务实例被创建时都被分配一个唯一的服务实例句柄(GSH)，这个句柄在一段时间内是不变的和唯一的。服务的其他信息保存在GSR 中。与GSH 不同的是，GSR在服务的生存期内可能变化，如版本信息和协议绑定信息。

OGSA 定义了获取更新后的GSR的机制。使用一个过期的GSR的结果未定义。接口HandleMap用于从GSH中解析出GSR。这个接口维护最新的GSH和GSR的映射，不会返回一个已经过时的引用。

2.1.2.2. OGSI

开放网格服务基础设施OGSI(Open Grid Service Infrastructure)是构建OGSA的基础设施，它建立在Web服务的基础上，并对Web服务规范进行了一些扩展，定义了网格服务的约定和规范，他们的协议和接口的行为、性质和属性。有了OGSI，已经完全可以实现网格应用。OGSA，OGSI和Web服务的关系如图2-4所示[18]。在Foster和Kesselman的The Grid 2新书中，他们做了这样的比喻:OGSI之于网格就像TCP/IP协议之于Internet。 OGSI定义网格服务时也采用了WSDL，但是由于WSDL所定义的元素不足以描述网格服务的特征，因此OGSI对WSDL进行了扩展，对WSDL中的portType定义了新的模式，关联新的域名gwsdl。

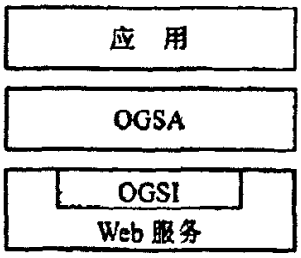

图 2-4 OGSA、OGSI 和 Web 服务的关系

OGSI对于Web服务的一个重要的扩展是服务数据(Service Data)。通过向请求者暴露服务实例的服务数据元素(Service Data Element, SDE)，使网格成为一个有状态的系统。用户可以通过接口访问一个服务实例的数据，用户可以读、更新、订阅服务数据，一个服务实例在运行过程中，可以动态增加和删除服务数据。

此外，OGSI对WSDL的另一个较大的扩展是支持portType的继承模型，继承属性会影响继承链上所有的服务数据的定义。

OGSI定义了网格服务的接口，根据这些接口的功能，可以把这些接口分为三组，如图2-5所示。

组	接口	操作	服务数据
1	GridService	findServiceData setServiceData requestTerminationTimeAfter requestTerminationTimeBefore Destory	Interfaces ServiceDataName factoryLocator GridServiceHandle GridServiceReference findServiceDataExtensibility setServiceDataExtensibility TerminationTime
	Factory	createService	CreateServiceExtensibility
	HandleResolver	findByHandle	HandleResolveScheme
2	NotificationSource	Subscribe	NotifiableServiceDataName SubscribeExtensibility
	NotificationSink	deliverNotification	
	NotificationSubscription		subscriptionExpression sinkLocator
3	ServiceGroup		MembershipContentRule entry
	ServiceGroupRegistration	Add remove	addExtensibility removeExtensibility
	ServiceGroupEntry		memberServiceLocator content

图 2-5 OGSI 定义的网格服务接口

第1组是支持网格服务行为、服务数据元素和静态服务数据值的接口(portType)。

目前有三个操作和多个服务数据。GridService是任何一个服务都必须有的接口，用于实现服务数据的操作:Factory用来创建网格服务实例；HandleResolver把一个GSH解析为一个GSR。

第2组是关于通知框架的接口。NotificationSource是一个可选择实现的接口类型，实现该接口可以使客户订阅根据服务数据值的改变发出的通知。NotificationSink是一个可选择实现的接口类型，用于接收通知。

第3组接口提供了网格服务成组的概念。网格服务可被按照特定的分类模式利用简单的聚合机制分组。ServiceGroup是一个可选择实现的接口类型，代表0个或多个服务的组的抽象接口。ServiceGroupRegistration继承了ServiceGroup接口，提供管理一个ServiceGroup的操作，包括向一个组中增加服务和删除服务。ServiceGroupEntry是一个ServiceGroup单个条目的代表，是由ServiceGroupRegistration的add操作创建的，每个条目包含一个成员服务的服务定位符和关于成员服务的信息。成员服务是通过服务组成员规则定义的。

2.1.3. 网格中间件

2.1.3.1. 中间件的概念

中间件[19]一般是指运行在客户机或服务器系统上的一种独立的系统软件或服务程序，是一种新型的软件设计模式[20]。在实际应用中，它可以实现多种功能，比如提供远程进程管理、空间信息资源分配、信息存储与访问、系统安全登录和认证、系统安全或服务质量监测等。中间件应被理解为是一类软件，而非某一种软件；在网格环境中，中间件不仅仅可以实现各种应用程序间的简单互连，而且它也可以实现它们之间各种更复杂的互操作。目前，在基于分布式环境的各种应用中，中间件的引入主要是为了解决网络通信方面的功能问题；其中，中间件的位置一般处于应用层和网络层之间，它通过对属于相应层次的功能实现并进行透明的封装，使得相应的应用层软件可以独立于低层实现机制(如计算机硬件和操作系统平台)单独进行开发，并实现不同平台间相同层次应用的跨平台的操作[21]。在已有的实际应用中，很多大型的企业级分布式应用标准的平台的建立都利用了中间件技术，通过各种中间件将大型企业分散的现有子系统进行组合，从而增强这

个系统集成的简单性以及健壮性。

中间件的基本概念如图2-6所示。

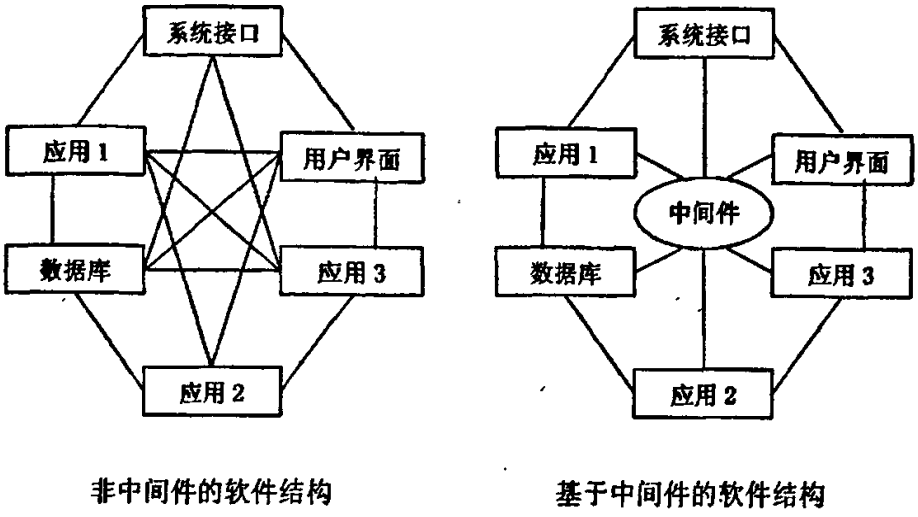

图 2-6 中间件的基本概念

如果将不同层次间的各种应用程序间的协同工作或互操作理解为一种客户/服务器的工作模式的话，引入中间件技术扩展了这种传统的客户服务器结构，形成了一种新的包括客户、中间件和服务器在内的三层或多层结构，这种结构为开发可靠的、可扩展的、复杂的事务密集型应用提供了有力的支持。在基于分布式的网格环境中，中间件可以被分为4种类型[22]：

(1)基于远程过程调用RPC(Remote Procedure Calls)的中间件。RPC是一种对传统程序设计语言过程调用的扩展，被调用的对象可以存在于分布式系统的任何物理平台上。远程过程调用是一种广泛使用的分布式应用程序处理方法。一个应用程序使用RPC来“远程”执行一个位于不同地址空间里的过程，并且从效果上看和执行本地调用相同。

(2)面向消息的中间件(MOM)，支持基于消息传递的进程间通讯方式。这类中间件既适用于客户/服务器模型，也适用于对等网模型，一般比基于RPC形式的中间件会具有更高的运行效率。MOM是利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。通过提供消息传递和消息排队模型，它可在分布环境下扩展进程间的通信，并支持多通讯协议、语言、应用程序、硬件和软件平台。目前流行的mom中间件产品有IBM的MQSeries, BEA的MessageQ等。

(3)基于对象请求代理(ORB, Object Request Brokers)的中间件。此类中间件是面向对象应用程序的首选。消息可通过ORB进行路由选择，ORB同时处理集成和安全方面有关的问题。随着对象技术与分布式计算技术的发展，两者相互结合形成了分布对象计算，并发展为当今软件技术的主流方向。1990年年底，对象管理集团OMG首次推出对象管理结构OMA(Object Management Architecture)，对象请求代理(Object Request Broker)是这个模型的核心组件。它的作用在于提供一个通信框架，透明地在异构的分布计算环境中传递对象请求。

(4)事务处理监控(TPM)。事务处理监控(Transaction Processing Monitions)最早出现在大型机上，为其提供支持大规模事务处理的可靠运行环境。随着分布计算技术的发展，分布应用系统对大规模的事务处理提出了需求，比如商业活动中大量的关键事务处理。事务处理监控界于client和server之间，进行事务管理与协调、负载平衡、失败恢复等，以提高系统的整体性能。它可以被看做是事务处理应用程序的“操作系统”。

中间件具有如下特点:①满足大量应用的需要；②运行于多种硬件和OS平台；③支持分布计算，提供跨网络、硬件和OS平台的透明性的应用或服务的交互:④支持标准的协议；⑥支持标准的接口。

2.1.3.2. 中间件原理

一般来说，从客户/服务器模型的角度来看，中间件的基本工作原理可以理解为图2-7所示[23]。客户端上的应用程序需要从网格中某个节点处获取一定的数

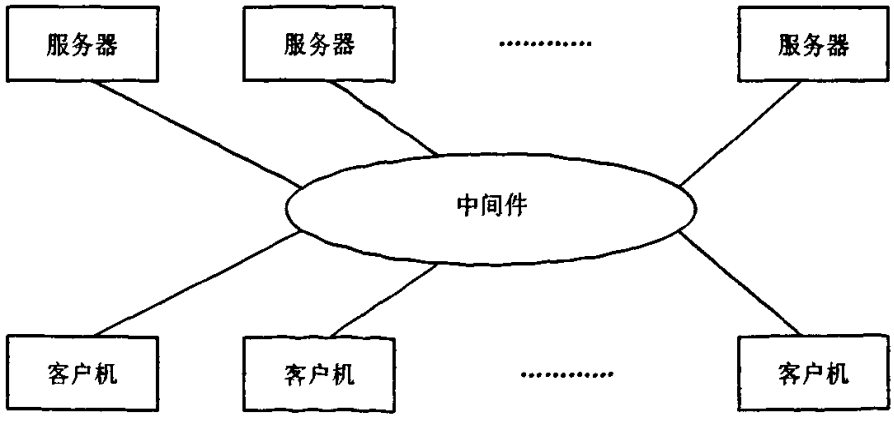

图 2-7 中间件的基本原理

据或者服务，而且这些数据和服务可能处于一个运行着和客户端不同的操作系统的服务器上，这种情况下，各种应用程序只需要访问中间件系统，中间件系统会自动完成到网络中查找目标数据源或者服务的任务，向目标提交客户请求，并将结果重组为答复信息，送回给应用程序。空间信息网格中间件技术的主要实现技术有CORBA技术、Microsoft的.Net技术以及SUN的J2EE技术。

应当指出的是，由于网格技术的初期发展主要是在高性能计算领域中，因此现有的网格中间件主要解决的问题是对计算资源的管理与调度，对其他资源的管理必须在现有的网格中间件基础上开发必要的软件包。

2.1.4. 网格中间件的实现标准—Globus Toolkit

网格中间件是组建一个网格的关键，目前只有 Globus、Legion、SNIPE 等，而以 Globus 为代表。Globus 是由 Argonne 国家实验室、南加州大学信息科学院、芝加哥大学联合开发的一个项目，Globus ToolKit[24]是 Globus 最重要的成果。Globus ToolKit 源码开放，任何人都可以从其网站上下载源代码。Globus ToolKit 提供了构建网格应用所需要的基本服务，如安全、资源发现、资源管理、数据访问、通讯以及错误探测等[25]。Globus 主要的研究领域包括资源管理、数据管理和访问、应用开发环境、信息服务以及安全等领域。Globus 网格技术的典型代表和事实上的规范，目前已在 NASA 网格（NASA IPG）、欧洲数据网格（European Data Grid）、美国国家技术网格（NTG）等 8 个项目中得到应用，包括 Entropia、IBM、Microsoft、Compaq、Cray、SGI、Sun、Veridian、Fujitsu、Hitachi、NEC 在内的 12 家计算机和软件厂商已宣布将采用 Globus ToolKit 作为网格技术的开发架构和开放标准。

下面介绍一下 GT3 的体系结构[26]。如图 2-8 所示，GT3 体系结构图中，灰色的部分是 GT3 Core 提供的。它们是建立网格服务的基础。OGSI 参考实现（RI）实现了 OGSI 规范 1.0 所定义的接口，以 API 和工具的形式提供给用户以方便开发和 OGSI 兼容的网格服务。安全基础设施（SI）提供 SOAP 安全，传输层安全，相互认证，单点登录服务认证等功能。GT3 核心还提供了系统级服务用来作为运行时和其他服务相关联的基础，它们建立在 OGSI-RI 和 OGSI-SI 之上。GT3 同时还提供了一些基础服务如程序执行，数据管理和信息服务等（见 2-8）。这些服务都是建立在 OGSI 和 GSI 组件之上的。用户定义服务是指由用户开放的高层服务，

它不由 GT3 提供，可以建立在任何 GT3 的组件之上，包括基础服务。所有这些服务是和抽象的 OGSI 运行时环境相交互的，我们称为网格服务容器。网格服务容器把应用和服务运行时细节分离开来，同时还控制服务的生命周期、把消息分发给对应的服务实例等。容器的前端封装了一个标准的 Web 服务引擎接口，用来实现 XML 消息映射。

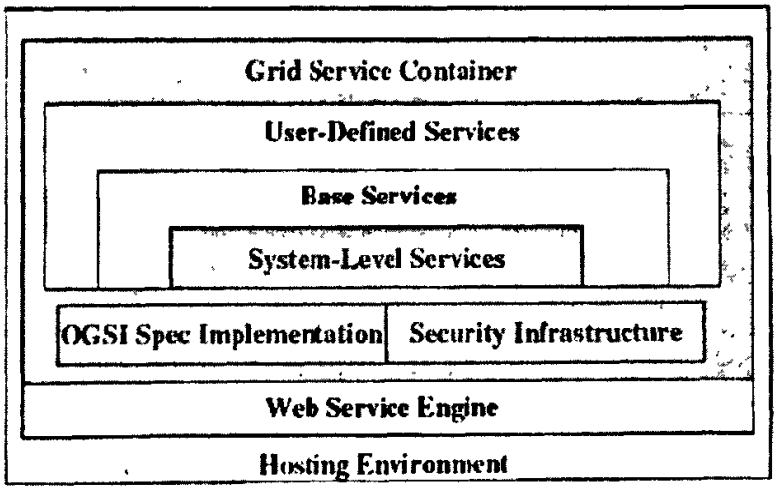

图 2-8 GT3 体系结构

最后，Web 服务引擎和网格服务容器宿主到一个主机环境中。

目前 GT3 支持 3 种 Java 主机环境：

1. Embedded：以库文件的形式将 OGSI 主机环境嵌入已有的 J2SE 应用中。
2. Standalone：一个可以部署网格服务的轻量级的 J2SE 服务器。
3. J2EE Web Container：将 OGSI 主机环境嵌入 Web 服务器中，该 Web 服务器具有 Java Servlet 引擎，如 Jakarta Tomcat。

GT3 的软件结构模型[27]包含两部分框架。一个是服务器端框架，另一个是客户端框架。服务器端框架如图 2-9 所示。

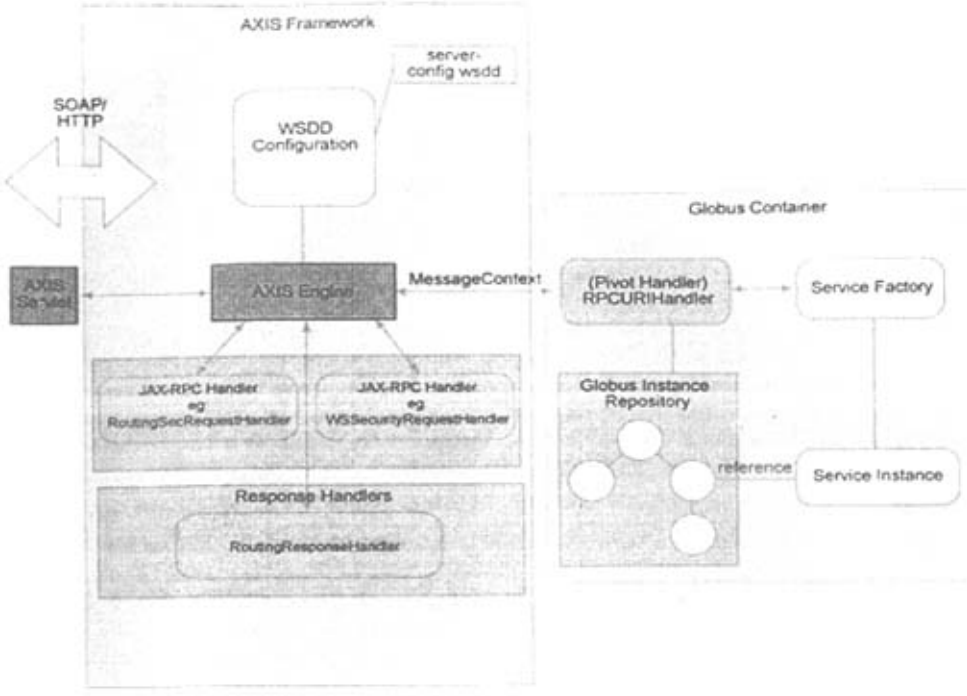

图 2-9 GT3 软件框架：服务段体系结构组件

如上图所示，GT3 服务端框架主要包括以下组件：

Web 服务引擎：这个引擎是由 Apache AXIS 提供，用来处理普通的 Web 服务行为，SOAP 消息处理，JAX-RPC 句柄处理和 Web 服务配置。

GT3 服务容器：GT3 提供一个容器来管理网格服务。它包括状态管理，生命期管理和“软状态”管理等。

目前 GT3 使用 Apache AXIS 作为它的 Web 服务引擎，它运行在 J2EEWeb 容器内，并提供一个 SOAP 消息监听者（AXIS Servlet）。它负责 SOAP 请求/响应进行序列化和反序列化，JAX-RPC 句柄调用和网格服务配置。GT3 容器提供了一个主句柄（pivot handler）来把从框架中的消息传递到 GT3 容器中[28]。

GT3 容器用来管理网格服务的状态和生命期。只要服务工厂产生一个网格服务实例，框架就会为这个实例生成一个唯一的句柄（GSH），这个服务实例注册到服务容器资源库中。服务容器资源库保存着所有的状态服务实例，它可以和其他框架组件和句柄联系以完成如下服务：

鉴别服务和调用方法

设置/得到服务属性（例如 GSH 和 GSR）

激活/钝化服务

把 GSH 解析为 GSR 和保持服务持久性

客户端的框架如图 2-10 所示。

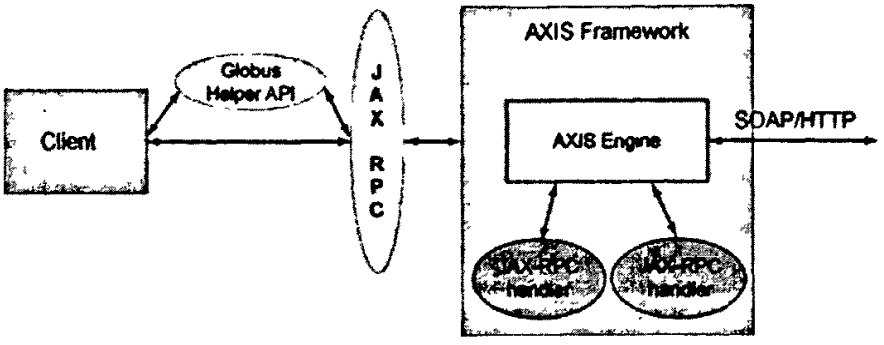

图 2-10 客户端的框架

如图所示，GT3 使用 JAX-RPC 客户端编程模式和 AXIS 客户端框架来编写网格服务客户端[29]，除此之外，它还提供了一个帮助来隐藏 OGSI 客户端编程模式的细节。

2.2. LF-Grid 的总体框架

LF-Grid是一个面向服务的语义网格的实现，它按照OGSA的简单运行环境构建了整个网格体系，补充了许多网格运行所需的组件，并且从语义角度对整个框架进行了扩展。

我们把 LF-Grid 的基本结构划分为 3 部分：客户端、LF-Grid 核心组件以及 LF-Grid 资源。如图 2-11 所示。

下面就对这三部分组件进行详细的介绍。

(1) LF-Grid 主要通过客户端展现给用户，用户可以通过 Grid Browser（网格浏览器）像使用 IE 浏览因特网一样方便地使用 LF-Grid。在 Grid Browser（以下简称 GB）中用户可以要求网格为自己提供某种服务，GB 会把用户的要求发送到网格中，网格执行一系列相应操作，最后把产生的结果在 GB 中显示给用户。在客户端还可以有其他的应用程序运行，为此 LF-Grid 提供了 Grid Application Interface（网格应用程序接口）。另外，网格管理员还可以通过 Monitor and Management Tool（监控和管理工具）来维护 LF-Grid 的正常运行。

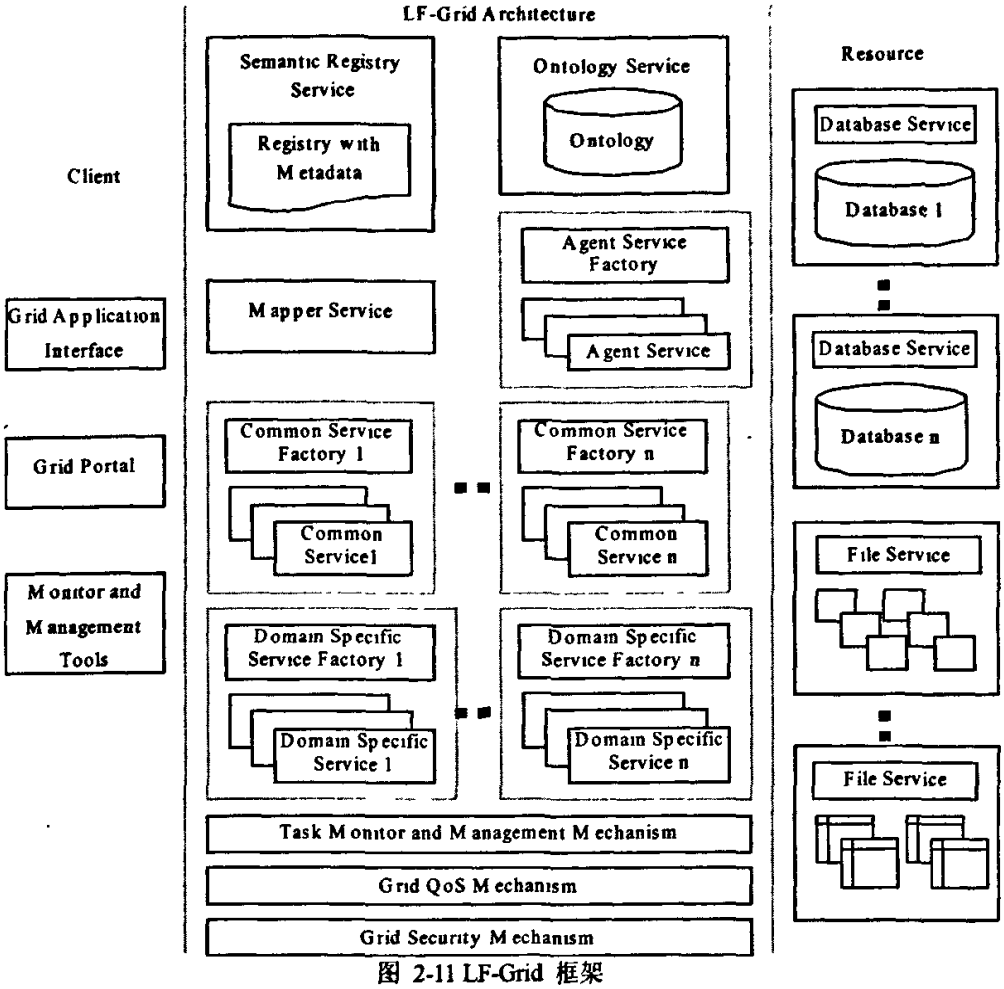

图 2-11 LF-Grid 框架

(2) LF-Grid 核心组件是整个网格运行的基础和关键部分。核心组件的设计主要用于支撑基于服务的网格应用的开发、部署、运行和调试，集中解决各类资源的共享和协同问题。LF-Grid 核心组件包括：

①Registry with Metadata(带有元数据的注册中心，以下简称 RM)主要用于服务的注册和发现，参考 Web Service 的注册中心我们加入了用来对服务进行描述的元数据（metadata），服务元数据以 XML 的形式保存在注册中心。采用 XML 的形式主要考虑利用它的跨平台性和它对语义的支持。当 LF-Grid 使用者准备创建一个服务的时候，他需要用本体（Ontology）中的概念对自己的服务进行描述，这样在查询某个服务时，我们就可以利用语义匹配进行准确的查找。在 Register with Metadata 的基础上，我们提供了一个 Registry Service 服务，用于实现注册、注销和查询功能。关于 LF-Grid 中的服务，我们将在第三部分详细介绍。

②为了更好的支持语义我们为LF-Grid引入了本体，一个本体是一个共享概念

化模型的形式化和显式的说明，也就是说本体准确地描述了一个领域内的概念体系。Ontology主要用于从语义角度描述服务，帮助提供更精确的服务匹配，使得用户对劳动就业和劳动保障领域有一个更清晰的认识，从而更好地整合网络上已有的服务和资源。Ontology由劳动力市场方面的专家建立，保存在采用OWL（Web Ontology Language）[30]描述的概念图文件中。Ontology对外提供了Ontology Service服务，以支持网格用户和其它服务对Ontology进行维护和查询。

③Mapper Service（映射服务）是LF-Grid中的核心服务之一，该服务是OGSA框架中必需的服务，主要为其它服务提供从GSH（Grid Service Handle，网格服务句柄）到GSR（Grid Service Reference，网格服务引用）的映射。GSH是网格服务实例在网格中全局唯一的名字，但是GSH还不能直接用于服务的调用，只有知道GSR（包含接口和消息格式）才能调用一个网格服务。当网格用户在注册中心找到自己所需服务的GSH之后，需要调用Mapper Service来得到对应的GSR，然后才能调用这个网格服务。

④Agent Service（代理服务）是专门用来帮助用户使用网格资源和服务而设计的一个服务， Agent Service Factory（代理服务工厂）为每一个用户生成一个Agent Service服务实例，该实例负责与GB交互并且在网格中代表用户的身份享受服务，我们以此来实现用户权限的管理。Agent Service位于网格基础框架之中，这使它的运行具有无可比拟的优势。

⑤Common Service（通用服务）是一系列为整个虚拟组织提供公共能力的网格服务，主要实现网格的信息协议和管理协议，提供资源的状态监控、系统容错、服务质量保证等机制。

⑥Domain Specific Service（领域服务）是对领域相关的常用功能进行抽象而得到的一组服务，一个领域服务可以通过调用一系列网格中的其它服务例如资源服务来完成一个相对完整的功能。领域服务的创建主要取决于网格用户的需要，我们倾向于把常用的，并且需要调用多个其它服务的功能封装为领域服务。Domain Specific Service可以由网格设计者提供，也可以在LF-Grid的运行过程中由用户创建。我们考虑在创建LF-Grid时同时创建就业管理、失业管理、再就业优惠待遇管理、失业保险、职业介绍等主要功能的领域服务。例如失业保险统计服务等等。

⑦Task Monitor and Management（任务监控和管理）模块负责监控、调度和

管理LF-Grid中运行的程序，使得网格系统可以正确地、高效地运行。

⑧Grid QoS Mechanism（网格服务质量机制）负责跟踪网格服务的运行，最终记录服务质量。LF-Grid的后续版本将支持用户或第三方机构对所调用的服务给出评价，该评价会作为服务质量元数据保存在RM中，为基于QoS的服务查找提供方便。

⑨Grid Security Mechanism（网格安全机制）用来保障LF-Grid不受外界恶意地侵害。

(3) LF-Grid资源包括网格中的高性能计算机、关系数据库以及某些仪器和设备等物理资源还包括网络带宽、应用程序等逻辑资源。由于这些资源可能隶属于不同的组织具有独立的访问策略，因此LF-Grid应用环境的异构性、分布性和自治性就凸显出来。为了解决这个问题，我们提出了Resource Service（资源服务，包括Grid Data Service、Computation Service、Experiment Service等），这是一组用于封装各类异构资源的网格服务，通过Resource Service在网格看来，原来各种各样的资源变为了具有统一的接口服务，更易于管理和使用。Resource Service运行在具体的资源节点上，具有更好的性能。同Common Service和Domain Specific Service一样， Resource Service也需要把自己的元数据注册在RM。

在LF-Grid中，劳动就业、劳动保障、社会保障以及其它组织和部门的数据库是一类最重要的资源，由于LF-Grid不可能对它们拥有所有的权限，因此直接访问这些数据库是不现实的，而在被允许的范围内提供可以控制接口的服务是一个可行的办法。我们为每一个分布式关系数据库提供一个网格数据服务（Grid Data Service）。

按照OGSA的观点，网格服务是网格体系的中心概念，LF-Grid继承了OGSA的这一思想，在LF-Grid中，用户可以用到的功能都被包装成网格服务。网格服务是一种提供一系列特定接口的web服务，OGSI定义了网格服务的标准接口，它们提供服务发现、动态服务创建、生命期管理、通知等功能。除此之外，为了支持服务元数据查询以及本体的查询与维护本文扩充了OGSI的接口，详见图2-12。其中，FindService接口由语义注册服务实现，本体服务则实现了OntologyService接口。

接口	操作	描述
FindService	FindByMetadata	按照服务元数据查询服务

OntologyService	Query	查询和识别本体中的词汇，包括词汇间的关系
	OntologyMaintain	在本体中增加、删除、修改词汇

图 2-12 扩展的 OGSI 接口

代理服务、领域服务、通用服务和资源服务是 LF-Grid 中主要用于实现业务逻辑的服务，它们由所对应的服务工厂创建。服务工厂是一种持久的网格服务，它们在接收了所需消息之后可以创建临时的服务实例，然后由服务实例来为用户提供服务。需要说明的是，在 LF-Grid 中，仅有一个代理服务工厂，而有多少种业务就有多少个相应的领域服务工厂，通用服务工厂与之类似。

在 LF-Grid 中，代理服务可以代表用户调用其它功能服务包括领域服务、通用服务和资源服务，其中领域服务为了完成一个完整的业务功能也可能调用其它服务并且这是一个递归的过程，由此形成的服务的调用关系如图 2-13 所示。

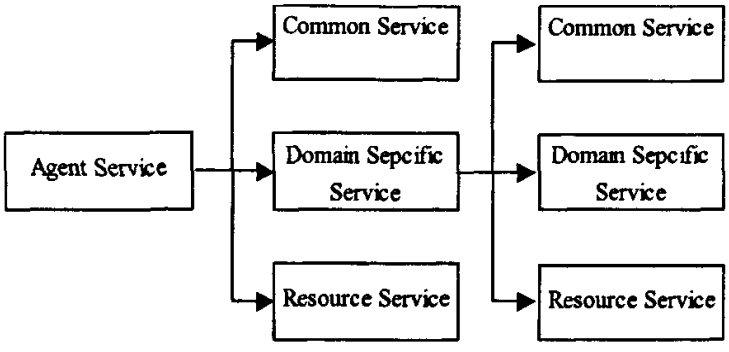

图 2-13 LF-Grid 中的服务调用关系

第三章 LF-Grid服务描述的语义扩展

在网格环境下，为了使用户发现最准确最相关的服务，需要采用标准的方式来描述服务的各种信息。如何进行服务的描述是网格服务发现的关键问题之一。服务的描述包括形式和内容（语义）两个方面。在形式方面，需要采用XML，WSDL等相关标准。在服务的内容描述方面，如服务的分类、功能等的描述，需要制定一套描述语义的标签以及标签之间的关联、规则。为了增加对服务的语义描述信息，LF-Grid引入OWL[30]本体（Ontology）。本体（Ontology）是对领域知识概念的抽象和描述。其主要原因在于本体使人机的交流建立在对所交流领域共识的基础上。而在服务的动态发现中，不可避免地要牵涉到领域概念，而这正是LF-Grid引入本体的目的。领域本体是领域专家建立的，它提供了该领域的概念定义和概念之间的关系，提供该领域中发生的活动以及该领域的主要理论和基本原理等，在该特定领域中可重用。利用本体来指导服务注册、管理与发现过程，以帮助服务提供者更加准确地对服务进行描述，用户能够基于本体实现服务的动态查找、绑定与调整。

本文使用WSDL规格2.0支持的元素和属性的扩展特性，对LF-Grid服务描述GWSDL文档进行了扩展修改。LF-Grid服务描述GWSDL文档在保持原来输入输出参数的基础上，增加了前提条件和效果两个标签，这样GWSDL[31]文档就能更准确地描述服务。引入OWL本体（ontology）语义模型表示服务的语义；为了将GWSDL文档中的服务信息与OWL本体概念关联起来，我们把LF-Grid服务描述GWSDL文档中的元素映射到相应的OWL本体概念上并且把这种映射关系连同OWL本体、GWSDL一起注册到RM，解决了服务缺乏语义描述的问题。

3.1. 服务描述 GWSDL 文档扩展前提条件和效果元素

OGSA采用了万维网服务的WSDL规范，因此LF-Grid服务使用WSDL规范描述，提供了服务信息。LF-Grid服务描述GWSDL文档通过指定输入和输出消息的结构以及服务运行时间绑定来定义服务的语法。GWSDL使用XML Schema描述消息组件的结构。

本文使用WSDL规格2.0支持的元素和属性的扩展特性，新添加了两个元素，一个是前提条件，一个是效果。每一个操作有许多前提条件和效果。

定义1 前提条件 服务执行时必须具备的一些逻辑条件，执行此服务操作时必须是TRUE。

定义2 效果　　执行服务后系统状态的变化。

例如，劳动力市场中，用户进行失业登记（unemployedRegistry）前提条件是用户必须是已经失业（unemployed）；进行失业登记操作后的效果是个人信息登记到相应数据库中（registed）。我们建议把前提和效果作为操作元素的子元素。使用这种扩展用户能查找到和需求更相关的服务。代码1描述了一个失业登记服务的WSDL文档，该服务只有一个操作是unemployedRegistry，其中前提条件Unemployed、效果Registed（下划线标出）是作为unemployedRegistry操作的子元素；此外，unemployedRegistry操作还有一个输入参数personalDetails，一个输出参数confirmation。

```
<?xml version="1.0" encoding="UTF-8"?>
<gwsdl:definitions targetNamespace="http://localhost:8080/lfgrid/services/
LFGridRegirstryService"
xmlns=http://schemas.xmlsoap.org/wsdl/
xmlns:SGExt=http://localhost:8866/lfgrid/GWSDLExtension
xmlns:SGOnt=http://localhost:8866/lfgrid/registontology.owl
.....
<schema targetNamespace=http://LFGrid
xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
<complexType name="PersonalDetails">
<sequence>
<element name="Name" type="string"/>
<element name="Age" type="int"/>
<element name="Sex " type="bool "/>
<element name="degree" type="string"/>
<element name="speciality " type="string"/>
<element name="unemployedDate " type="date"/>
.....
</sequence>
</complexType>
</schema>
</gwsdl:types>
<gwsdl:message name="OperationRequest"
            SGExt:onto-concept= "SGOnt:PersonalInformation">
<gwsdl:part name="in0" type="tns1:PersonalDetails"/>
</gwsdl:message>
<gwsdl:message name="OperationResponse"
          SGExt:onto-concept =" SGOnt:ConfirmationMessage">
<gwsdl:part name="return" type="tns1:Confirmation" />
</gwsdl:message>
<gwsdl:portType name="LaborPower ">
<gwsdl:operation  name="unemployedRegistry "  parameterOrder="in0"
            SGExt:operation-concept = "SGOnt:unemploymentRegistry" >
<gwsdl:input message="intf:OperationRequest" name="registryRequest " />
<gwsdl:output message="intf:OperationResponse" name="registryResponse"/>
<SGExt:precondition name="Unemployed"   type ="bool"
SGExt:precondiction-concept=" SGOnt:Umemployment"/ >
<SGExt:effect name="Registed"  type =" bool"
SGExt:effect-concept=" SGOnt:Registed/>
</gwsdl:operation>
.....
```

代码1　失业登记服务的GWSDL文档

3.2. 将服务描述 GWSDL 文档中的元素映射到本体概念

为了将 GWSDL 文档中的服务信息与 OWL 本体概念关联起来，方便用户使用语

义查找所需服务，本文把 GWSDL 中服务信息映射到相应的 OWL 本体概念上，实现对 GWSDL 概念进行语义扩展。这些扩展是按照 OWL-S ServiceGrounding[30]的扩展设计的。映射包括三个方面的内容：操作映射到本体概念；将消息部分映射到本体概念；将前提条件和效果映射到本体概念上。下面对这三方面分别进行详述。

3.2.1. 将操作映射到本体概念

像 WSIF[32]这样的工具，如果知道 WSDL 文件的位置和操作的名字，就可以调用 web 服务。网格服务发现不仅涉及到 GWSDL 文件的定位，还包括相关操作的调用。每一个 GWSDL 描述含有大量的不同功能的操作。DAML-S[33]的匹配没有考虑操作功能性因素，影响了查找服务的准确性。与之相比，本文的服务描述更适合服务的语义发现。为了扩展服务的语义描述，找到相关的操作，这些操作应该映射到描述操作功能性的合适的 OWL 本体概念上。如代码 1，操作 unemployedRegistry 使用 WSDL2.0 的 operation-concept 扩展属性映射到 unemployedRegistry 本体概念上。

这就允许用户基于本体概念来查找操作。操作名称和本体概念之间的映射储存在 RM 中。

3.2.2. 将消息部分映射到本体概念

消息部分（操作的输入和输出参数），在 GWSDL 文件中使用 XML Schema 结构来定义的。因为服务提供者将这个 Schema 定义和服务描述作为内联元素，所以致使共享和重用很困难。而本体，有极强的表达力，支持共享定义，可以注解 WSDL 的消息部分。使用本体不仅把用户需求和服务发布带到一个通用概念空间，而且帮助推理机制找到更好的匹配。通过在 GWSDL 中使用 OWL 本体，这些结构的语义描述使原来只有网格服务提供者知道的信息为大家所知。如代码 1 所示，输入 PersonalDetails 和输出 Confirmaton 使用 WSDL2.0 的 onto-concept 扩展属性映射到本体概念 PersonalInformation 和 ConfirmationMessage。

3.2.3. 将前提条件和效果映射到本体概念

本文把前提条件和效果作为操作的子元素。为了加入语义描述，方便找到更相关的服务，它们应该映射到相应的 OWL 本体概念上。如代码 1 所示，前提条件 unemployed 和效果 registed 分别使用 WSDL2.0 扩展属性的 precondition-concept 和 effect-concept 扩展属性映射到本体概念 unemployment 和 registed。

笔者认为前提条件和效果对于服务发现的意义也是十分重大的。基于操作，输入和输出的服务匹配后再加入前提和效果的匹配会很大程度上帮助用户找到最相关的服务。有可能许多操作有相同的功能，还有相同的输入和输出，但是有不同的效果。举一个更明显的例子如有个操作“bookTicketAndSend” 和buyTicket 有相同的功能，输入和输出，但是分别有不同的效果“CardCharged－TicketBooked－sent”和“CardCharged－TicketBooked－PickUp”，一个是把票送到用户手里，另一个让用户自己来取票。用户对效果需求不同，相应操作也是不一样的。所以前提和效果因素对服务发现也是非常重要的。

第四章 LF-Grid服务的语义注册

完成对服务的描述后，为了能使网格服务被共享，需要把服务在网上进行发布。网格服务封装后，必需到注册服务中注册，才能真正成为网格服务，供用户使用。注册服务作为网格的中间件，在整个OGSA架构中占有很重要的地位，所有网格服务都必须到上面注册，并分配一个全球唯一的标识符，即网格服务句柄GSH(Grid Service Handle)，从而实现服务在网格平台中的定位。

网格服务是有状态的，并且可以动态地创建和撤消，这样就需要通过一种方式来将一个动态创建的服务与另一个服务区别开来，分配一个全球唯一的标识符，网格服务句柄GSH就是这个目的。

支持服务发现的网格服务叫做注册。一个注册服务根据两个东西来定义:一个是注册接口，它提供GSH的注册操作；二是相关的服务数据元素，它包括注册的GSH的信息。

常用的网格服务注册系统是OGSA TP5[34]，该系统的基本注册单位是网格服务，以及由这些网格服务衍生的运行态网格服务实例。网格服务作为Web服务的扩展，不仅包含由Web服务描述语言定义的各种服务元素，还定义了网格服务扩展元素:服务数据。不过本文服务发现机制用到的是有关Web服务描述语言定义的各种服务元素的知识然而在OGSA TP5注册系统中，网格服务存储中心需要在容器启动后加载到内存，存储容量受内存限制，对于信息网格这种大型，海量数据注册、激活和实例创建是不利的，网格服务被描述、部署和调用的过程由网格容器屏蔽，要调用远端服务必须获取相应的网格服务句柄GSH(由网络协议、远端服务器的IP地址、端口号和网格服务的相对路径组成)。OGSA TP5中虽然提供了远端注册功能模块，但那只是对等网格节点之间互相传递GSH。这种远端注册并不为全局可见，网格服务之间的协调能力有限。

为使网格服务之间有效协调，网格中需要设立全局可见的独立注册中心，专著于管理网格服务的发布和查询。网格容器中可供调用的用户级网格服务局限于本地容器，网格容器之间不存在网格服务的交互和协作，网格服务无法做到高效重复利用。为解决这些问题，网格的注册系统必须引入Web服务架构中的注册中心:

UDDI(Universal Description Discovery and integration统一描述、发现和集成)[35]。UDDI作为Web服务架构中服务发布者和服务请求者之间的第三方，是可以存储大量服务描述的注册中心。服务提供者将WSDL描述的服务向UDDI发布；服务请求者可以向UDDI查询所需的服务，在取得相应的服务描述信息后向服务的提供方发出调用请求。UDDI拥有对应WSDL的数据结构和各种有关服务查询的消息格式，制定了大量的标准API。

4.1. UDDI 注册数据模型

UDDI(Universal Description, Discovery and Integration) [35]是一套基于Web的、分布式的、为Web服务提供信息注册和查询的实现标准规范。UDDI注册中心为Web服务提供了一个良好的服务发布、维护和管理环境，是实现Web服务构架不可缺少的组成部分，受到了业界的强大支持。通过UDDI，所有的Web服务应用系统都能够通过一个共通的标准相互沟通。它定义的数据结构为描述业务和服务的基本信息提供了一个框架，并构架了通过任意的描述语言来提供更多服务细节的可扩展机制。在特定的行业领域以及不同的协议层次中存在着很多这样的描述语言，例如，受到业界广泛支持的WSDL语言，它在技术层面上描述如何使用服务。

UDDI注册使用的核心信息模型由XML Schema[36]定义。UDDI XML Schema定义了四种主要的信息类型，它们是技术人员在需要使用合作伙伴所提供的Web服务时必须了解的技术信息。它们是:商业实体信息(businessEntity结构)、服务信息(businessService结构)、绑定信息(bindingTemplate结构)和技术规范信息(tModel结构)。下图表示了UDDI信息模型[37]以及各部分之间的关系:

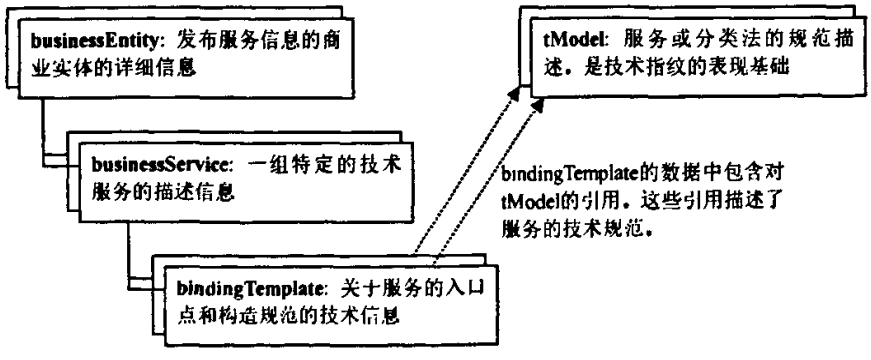

图 4-0-1 UDDI 信息模型

下面简单介绍一下UDDI中的两个重要数据结构tModel和businessServicee这两个结构在将Web服务语义描述应用于UDDI目录时非常重要。

tModel结构用来描述与某一规范、概念或共享设计的兼容性。tModel的主要角色是作为描述一种技术规范的机制存在的。一个tModel是描述UDDI注册中心中各种商业、服务和模板结构的一种方式[38]。任何抽象的概念都可以在UDDI注册中心中以一个tModel的形式进行登记。例如，如果定义了一个新的WSDL端口类型，就可以在UDDI中定义一个代表那个端口类型的tModel。然后就能够通过将tModel和一个商业服务绑定模板来指定实现了端口类型的商业服务。此处主要是用它来代表一个抽象服务。当一个抽象服务以tModel的形式在UDDI中进行注册时，它将被分配一个唯一的键值，对应于tModel结构的tModelKey属性。当服务提供者把一个服务实现发布在UDDI中时，可以引用这个链值从而说明其实现了相应的抽象服务。tModel的语法结构见图4-2。

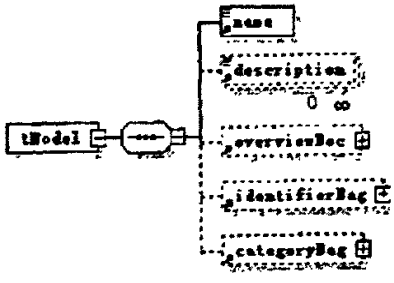

图 4-0-2 tModel 结构图

business Service结构对应于发布在UDDI中的一个服务，如何使用和访问这个服务等细节信息则由其包含的一个或多个bindingTemplate提供。bindingTemplate包含有关服务的详细信息。它指定了一个网络访问点(对应于其accessPoint元素，定义了对应bindingTemplate所描述的服务调用入口的访问地址，用户可以通过该地址访问具体的服务)，并包含对一系列描述服务的tModel入口的引用。businessService的总体结构见图4-1。

bindingTemplate主要通过tModelInstanceInfo元素来引用已注册的tModel。在这里被引用的tModels的组合构成指纹技术去提供服务。tModelInstanceInfo有属性tModelKey，其语法结构见图4-3，其所包含的instanceDetails的语法结构见图4-4。

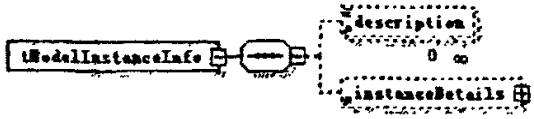

图 4-3 tModelInstanceInfo 结构

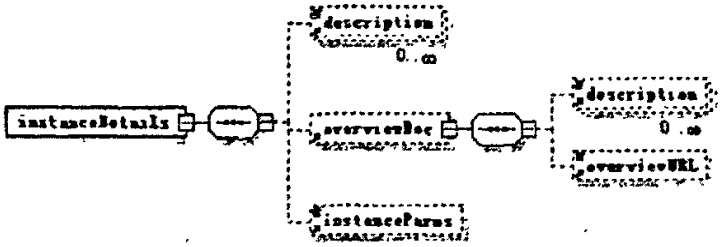

图 4-4 instanceDetails 结构

4.2. 在 UDDI 中加入语义

UDDI 在服务的语义发现中要做以下两个任务：(1) 在 UDDI 的现有结构中存储服务的语义描述；(2) 提供一个接口实现基于语义的查询。本文的设计思路类似于把 DAML－S 映射到 UDDI 结构上；但不同于它，而是把 WSDL 映射到 UDDI 结构。

目前 UDDI 使用 tModels 描述商业和服务，只能支持有限形式的语义。在网格服务发布时，首先使用本体概念描述网格服务 WSDL 中的操作、和它们的消息部分、前提、效果。这些本体概念使用 UDDI 结构存储。Tmodels 在 UDDI 数据结构中是元数据结构，它提供描述和规格书、概念、共享理解的一致性。它们在 UDDI 注册中有许多用处。普遍公认的规格书和分类法都可以在 UDDI 中注册成 tmodels。它们还可以用来将实体和分类中的个体节点联系起来。当 tModel 在 UDDI 注册了，分配给它一个唯一的 key，实体通过这个 key 找到它。将 UDDI 中的实体分类，tModels 经常和 categoryBags 联系在一起使用。Categorybags 可以使实体根据一或多个 tModels 分类。使用 UDDI 版本 3 新的分组结构 keyedReferenceGroups ，可以将 tmodels 分类再分组。本文使用 keyedrefencegroup 和 tmodels 一起将 operations 分组。

为了描述 UDDI 中的语义信息，本文在 UDDI 中，创建了 6 个 tModels。第一个 tModel 描述相关领域中操作的功能，第二和三个 tModel 分别描述输入和输出的本体概念，第四和第五个 tModel 分别描述前提和效果的本体概念，最后一个 tModel 描述每个操作按照输入、输出、前提和效果的分组情况。这些 tModels 各自的本

体通过 tModels 的“overviewURL”标签联系。所有的 tModels 可以链接成一个综合的本体。如代码 2，创建了一个 keyedReferenceGroups 描绘代码 1 WSDL 文件中的操作 unemployedRegistry 及其输入输出参数、前提和效果。每一个 keyed reference 都有一个 keyvalue 描述本体概念，一个 tModelKey 描述本体本身。举例：tModel“ OPERATION_CONCEPTS”用来存储 WSDL 操作和本体概念之间的映射。其包含操作的名字作为 keyName，操作映射到的本体概念作为 keyValue。相似的，输入和输出的映射分别使用 INPUT_CONCEPTS 和 OUTPUTY_CONCEPTS tModels 存储；前提和效果分别使用 PRECONDICTION-CONCEPTS 和 EFFECT-CONCEPTS 进行本体映射并存储。每一个操作和它的输入、输出、前提和效果使用 MAPPINGGROUP tModel 组合到 keyedReferenceGroup 中。

```
<businessService businessKey="uddi:LFGRID_REGIST.example"  serviceKey="…" >
<categoryBag>
  <keyedReferenceGroup
    tModelKey=  "uddi:ubr.uddi.org:categorizationGroup:MAPPINGGROUP" >
    <keyedReference
    tModelKey="uddi:ubr.uddi.org:categorization:OPERATION_CONCEPTS"
    keyName="unemployedRegistry"  keyValue="unemploymentRegistry" />
    <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:INPUT_CONCEPTS"
    keyName="Input"  keyValue="registryRequest" />
    <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:OUTPUT_CONCEPTS"
    keyName="Output"  keyValue="registryResponse" />
    <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:
     PRECONDICTION _CONCEPTS"
    keyName="precondition"  keyValue="Unemployment" />
    <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:EFFECT_CONCEPTS"
    keyName="effect"  keyValue="Registed" />
  </keyedReferenceGroup>
</categoryBag>
</businessService>
```

代码 2 操作、输入、输出在 UDDI 中的表示

4.3. LF-Grid服务的发布过程

现在讨论如何在UDDI中发布扩展定义的服务，这主要包含三个步骤。

第一步，使用GWSDL语义扩展创建服务语义描述。

第二步，将扩展的WSDL服务描述以tModel的形式在UDDI中注册，其overviewDoc元素分别将指向发布在网格上的相应的WSDL文档。

第三步，发布一个引用上述GWSDL描述的tModel的businessService描述。用GWSDL描述的该服务的基本信息和其他属性，在UDDI中进行注册，即在UDDI中创建一个相应的businessService条目。该businessService条目将为每一个服务访问点创建一个bindingTemplate元素，其网络地址在bindingTemplate的accessPoint元素中指定。对于该服务bindingTemplate将创建一个tModelInstanceInfo元素，其属性tModelKey设置为扩展的WSDL服务描述对应的键值。

下面是一个发布的例子。

```
<businessService businessKey="uddi:LFGRID_REGIST.example" serviceKey="…" >
<categoryBag>
 <keyedReferenceGroup
   tModelKey= "uddi:ubr.uddi.org:categorizationGroup:MAPPINGGROUP" >
   <keyedReference
   tModelKey="uddi:ubr.uddi.org:categorization:OPERATION_CONCEPTS"
   keyName="unemployedRegistry" keyValue="unemploymentRegistry" />
   <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:INPUT_CONCEPTS"
   keyName="Input" keyValue="registryRequest" />
   <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:OUTPUT_CONCEPTS"
   keyName="Output" keyValue="registryResponse" />
   <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:
    PRECONDICTION _CONCEPTS"
   keyName="precondition" keyValue="Unemployment" />
   <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:EFFECT_CONCEPTS"
   keyName="effect" keyValue="Registed" />
 </keyedReferenceGroup>
</categoryBag>
</businessService>
```

代码3 扩展的GWSDL服务描述发布例子

其中 tModel键值“TTTTTT”引用了第二步中发布的一个扩展的WSDL服务描述对应的tModel。

第五章 LF-Grid服务的语义发现

由于目前的网格服务发现是在基于语法性的服务描述基础上，通过关键词(如服务名称，服务描述等)的匹配来实现的。因此，缺乏对服务内容的理解，使得网格服务发现的结果无论在精确度还是返回率上都难以满足用户日益增长的需求。为了得到更好的效果，需要更加智能的网格服务发现技术。

扩展LF-Grid服务的语义描述和在UDDI中加入语义的目的是改善服务动态发现过程，支持对服务内容的理解，提高查全率和查准率。本节在前面对语义描述和语义注册知识介绍的基础上，对服务发现机制进行研究。 在服务发现中，语义匹配是一个关键问题。下面就详细讲述语义匹配问题。

5.1. 语义匹配

5.1.1. 匹配原理

本体ontology[39][40]是由概念(concept)、属性(property)、语义联系(semantic relation) 组成。所以在本体匹配时应该考虑这几方面的因素。

本体匹配需要满足下面两个需求。

第一：须考虑本体概念描述的语言特征。本体概念的意义取决于它们定义时所起的名字和它与其他本体概念之间的语义联系。

第二：须考虑本体概念描述的背景特征，要区分属性和概念。在本体概念定义时属性的作用在不同本体中有不同的影响度。

语言特征 本体概念定义时所起名字的含义。查找关于术语和术语联系的字典如WordNet可以获得其语言特征。术语联系定义为是synonymy（同义）， similar（近义词），norelation（没联系）。

背景特征 本体概念的上下文定义为它的属性和邻接的并集。某个概念的邻接是指与其有语义联系的概念的集合。如下图本体概念Hotel的属性为Rooms和Class，而它的邻接是Accommodation。

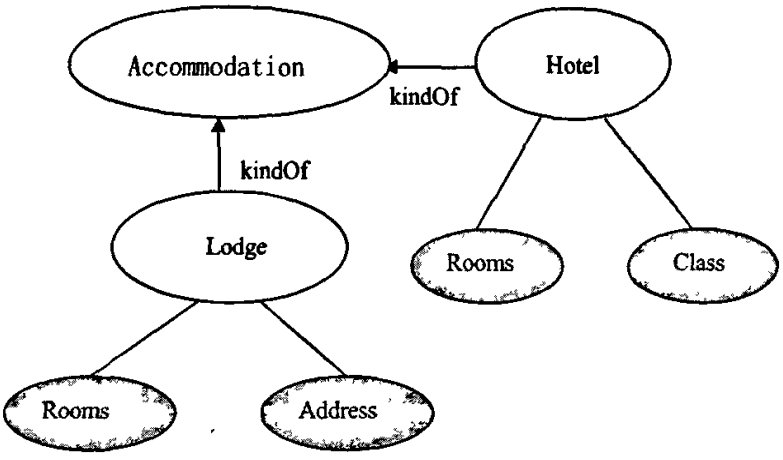

图 5-1 本体概念图形表示

所以计算本体概念之间的语义相似度，既要计算其语言相似度，也要计算背景相似度。另外，语义相似度还考虑了字典中的术语联系和背景中的语义联系以及属性的影响程度。根据语义影响程度，赋予这些联系不同的权重。

术语联系权重比较：$W_{synonymy} \geq W_{similar} \geq W_{norelation}$

语义联系权重比较：$W_{same-as} \geq W_{kind-of} \geq W_{part-of}$

本体的属性也分为两种：一种是强属性（sp），它是本体概念不可缺少的特征；另一种是弱属性（wp），是可选的。属性的权重比较如下

属性权重比较： $W_{sp} \geq W_{wp}$

其中，术语联系权重用来执行语言相似度计算的，而属性权重和语义联系权重是用来执行背景相似度计算的。

5.1.2. 语义相似度的计算

计算本体概念之间的语义相似度，既要计算其语言相似度，也要计算背景相似度。

语言相似度 语言相似度函数LS(c, c')用来计算c和c'的语言相似度，通过查找字典，计算c和c'的名字的语义匹配程度。函数A(t, t')用来计算两个术语t和t'之间的相似度。

$$A(t,t') = \begin{cases} Wtt_1 \bullet Wt_1t_2 \bullet \cdots\cdots \bullet Wt_nt' & \text{iff } \exists\ t \Rightarrow t' \\ 0 & \text{否则} \end{cases} \quad (1)$$

其中，Wt_it_j表示t_i和t_j的术语联系权重；

公式（1）表示如果从术语t到t'至少有一条路径，即它们之间通过其它本体概念相互联系，具有术语联系，那么A(t, t')等于最佳路径上权重的乘积，否则，相似度为0。

本体概念c和c'的语言相似度LS(c, c') 就等于它们名字A(nc, nc')的语言相似度 。公式如下：

$$LS(c,c') = A(nc,nc') \tag{2}$$

背景相似度 背景相似度函数CS(c, c')用来计算本体概念c和c'的背景相似度。计算背景相似度，一方面，需要计算这两个概念它们的背景元素集合的语言相似度，另一方面，需要计算上下文的紧密度。紧密度函数CL(e, e')计算两个概念上下文中元素之间的紧密程度（如两个属性、两个语义联系，或者一个语义联系和一个属性）。公式如下：

$$CL(e,e') = \begin{cases} T(dte,dte')\bullet(1-|We-We'|) & \text{iff } e\in P(c), e'\in P(c') \\ 1-|We-We'| & \text{否则} \end{cases} \tag{3}$$

其中dte和dte'是e和e'预定义的数据类型，We和We'表示e和e'各自的权重。权重的差的绝对值越大，紧密度越小。P(c)和P(c')分别表示本体概念c和c'的属性集合。当比较的元素是属性，还要考虑它们的数据类型是否相容。如果dte和dte'相容，则$T(dte, dte')$等于1，否则等于0。

紧密度是指两个背景元素之间的紧密程度。两个背景特征元素的匹配度不仅考虑的紧密度，还需要计算二者的语言相似度。匹配度函数$m(e, e')$计算的是两个背景元素之间的匹配度。函数m(e,e')等于背景元素e和e'的语言相似度（公式1给出）和紧密度（公式3给出）的平均数。表示如下：

$$m(e,e') = \frac{A(ne,ne') + CL(e,e')}{2} \tag{4}$$

其中e，e'分别是本体概念c和c'的背景元素，ne和ne'分别是这两个元素的名字。假定元素e是本体概念c背景中的某一个元素，$\overline{m(e)}$表示e和本体概念c'的背景中任何一个元素e'匹配度中最大值。

$$\overline{m(e)} = \max(\{m(e,e') \mid e' \text{是本体概念} c' \text{背景中的任一个元素}\}) \tag{5}$$

背景相似度函数CS(c,c')等于本体概念c背景中所有的元素匹配度最大值的平

均值。$CS(c,c')=\frac{\sum_{i=1}^{n}\overline{m(e_i)}}{n}$ （6）

其中 n 等于本体概念 c'的背景中元素的个数。

语义相似度　语义相似度函数SS(c,c')，结合这两个本体概念 c 和 c'的语言相似度和背景相似度，给出了它们相似度的综合值。本体概念 c 和 c' 的语义相似度 SS(c,c') 等于它们的语言相似度（由公式（2）给出）和背景相似度（由公式（6）给出）的平均值。函数SS(c,c') 表示如下：

$$SS(c,c')=\frac{LS(c,c')+CS(c,c')}{2} \quad (7)$$

自此，本体概念 c 和 c'的语义相似度就计算出来了。

5.1.3.　LF-MCH 匹配算法

在前面匹配原理的基础上，本文提出了 LF-MCH 匹配算法。算法如下：

LF-MCH 算法	说明
输入：本体概念 c 和 c'	
输出：SS	；语义相似度
Begin	
抽取 nc 和 nc'	；nc 和 nc'是本体概念 c 和 c'的名字
抽取本体概念 c 的所有属性和邻接，组成 c 的背景特征集合	
抽取本体概念 c'的所有属性和邻接，组成 c'的背景特征集合	
$LS=LS(c,c')$ $CS=CS(c,c')$ $SS=\frac{LS+CS}{2}$	；计算各种相似度
End.	

LF-MCH 匹配算法不仅仅考虑了本体概念名字的意义，还考虑到了本体概念的背景特征。LF-MCH 匹配算法提高了查询的精确度，避免了冗余信息的出现，提高了查询的效率。

5.2. 服务的语义发现算法

在前面讲述的语义匹配算法基础上，本文提出了基于服务模板的本体使能发现算法。用户需要按照模板填写服务请求。

LF-Grid 服务的语义发现三阶段算法如下：

输入：用户按照利用本体概念构造的服务模板输入所需服务的操作(operation)、以及对其所期望的相似度 OS([0,1]区间某个数)；输入参数(input)和输出参数(output) 以及对其所期望的相似度 IOS；前提条件(precondition)和效果(effect) 以及对其所期望的相似度 PES；

输出：符合用户服务请求的服务 (TargetService)；

Begin

Step1：首先根据操作本体概念按照 LF-MCH 算法匹配服务，如果语义相似度大于 OS，则将符合要求的服务存储在服务集 ServiceResults1；

否则 显示 “没有符合功能要求的服务”，退出程序；

Step2：将用户填写模板中的输入、输出参数分别和 ServiceResults1 中每个服务的输入、输出按照 LF-MCH 匹配算法进行语义匹配，如果输入、输出参数的语义相似度都大于 IOS，将符合语义匹配要求的服务存储在服务集 ServiceResults2；

否则 显示 “没有符合输入和输出参数要求的服务”，退出程序；

Step3：对于 ServiceResults2，再按照第二阶段同样的方式对其前提条件和效果按照 LF-MCH 匹配算法分别进行语义匹配，如果前提条件和效果的语义相似度都大于 PES，则对符合语义匹配要求的服务按照语义匹配程度由高到低进行排序，将语义匹配程度最高的服务存储在 TargetService；

否则，显示 “没有符合前提条件和效果要求的服务”；

End

现有大部分的服务语义发现算法只是考虑了服务的输入和输出参数的语义匹配；因为有可能许多操作有相同的功能，还有相同的输入和输出，但是有不同的前提条件和效果。所以 LF-Grid 服务语义发现三阶段算法在此基础上增加了 LF-Grid 服务的前提条件和效果这两个条件的语义匹配，会很大程度上帮助用户找到与服务请求更相关的 LF-Grid 服务，提高了服务查询的准确性。

5.3. 示例

图 16 表示了映射 LF-Grid 服务描述 GWSDL 结构映射到劳动力市场领域中具体本体节点的概念化过程。这个映射在发布过程中存储在 RM 中。在图 5-2 中，操作 unemployedRegistry 映射到本体节点 unemployedRegistry，LF-Grid 服务描述 GWSDL 文档中的输入概念 PersonalDetails 和输出概念 Confirmation 分别映射到 RegistryServices 本体中 PersonalInformation 节点和 ConfirmationMessage 节点。

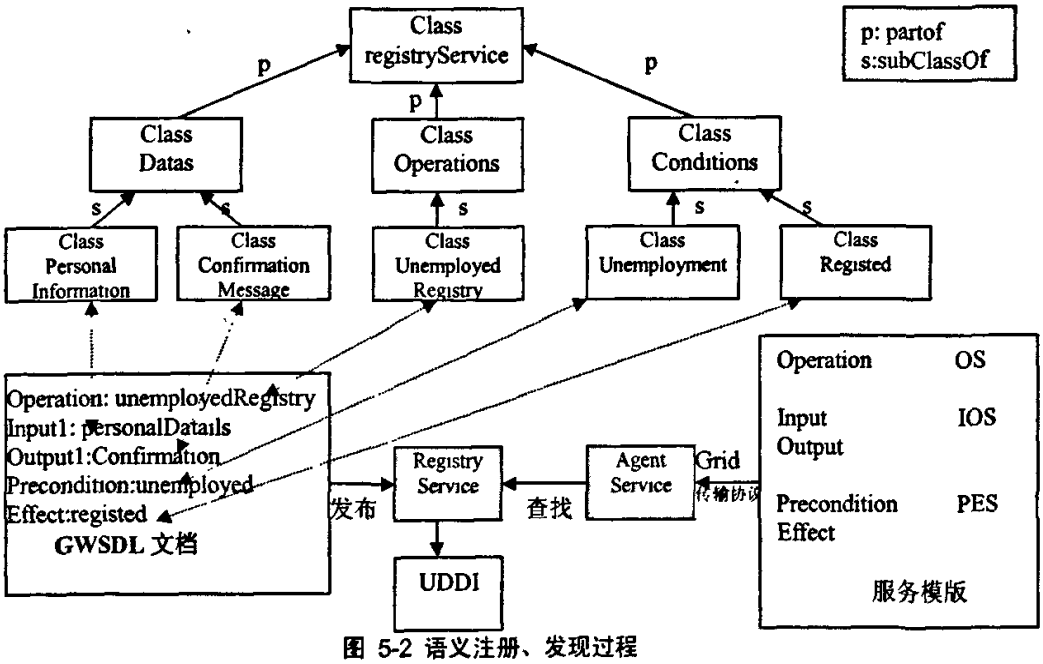

图 5-2 语义注册、发现过程

例如，某用户查询失业登记服务，首先在客户端按照服务模板填写相应的操作概念 unemployedRegistry，输入概念 PersonalInformation，输出概念 ConfirmationMessage；前提条件 Unemployment 和效果 Registed。客户端提交后通过 Grid 传输协议把用户需求传输给 AgentService。该 AgentService 按照 LF-Grid 服务语义发现三阶段算法进行本体概念的语义匹配，找到与服务请求更相关的 LF-Grid 服务，然后根据映射关系查找 RM，找到相应服务的 WSDL 文档，绑定服务的 SOAP 消息，调用相应服务。

第六章 结束语

随着网格技术的迅速发展，网格提供的服务种类的不断增多，服务规模的不断扩大，让用户能够在这么繁多的服务中找到适合自己需要的服务成为一个新的挑战。这正是网格服务发现技术的任务。网格服务发现的研究目标是服务发现的高效率，在服务发现技术中，利用语义描述和服务本体论是达到该目标的有效途径：而如何用OWL描述服务，如何将本体论应用到网格服务的匹配中已成为研究界和业界共同关注的热点。在本论文中，较为详细地探讨了网格环境下基于本体的网格服务发现机制的研究，针对一些问题，本文抓住网格服务描述和服务匹配这两个相互关联的重要问题，提出了一些解决方法，取得了一定的成果。并对进一步的工作进行了探讨。

本文融合了已有的服务发现策略的优势，提出了一种基于本体描述的简单、实用、可实现服务准确匹配的服务发现策略。基本思路如下：对服务描述WSDL文档使用本体扩展语义，实现对服务的语义描述；基于UDDI规范扩展服务描述语义信息，将扩展的语义信息以tModel的形式存储在UDDI中心；在上述基础之上，服务发现过程中使用三阶段语义发现算法获得满足服务请求的网格服务，其中LF-MCH匹配算法是服务发现的关键技术。

6.1. 主要工作总结

(1) 网格服务语义描述的研究

主要研究如何定义和表示网格服务。服务描述旨在为服务提供者和服务请求者提供标准的方式来描述，是服务发现的基础。基于服务发现的需求，本文在分析了各服务描述语言的基础上，参考Web服务本体论OWL-S，根据WSDL 2.0版本的可扩展性，提出了在服务描述GWSDL文档，扩展服务功能性信息的语义描述，为服务发现奠定良好的基础。

(2)UDDI语义扩展的研究

为了实现扩展后的服务描述的有效发布，本文利用UDDI的数据实体tModel的

特点，通过把服务描述的语义信息以tModel的形式注册到UDDI中心，实现扩展服务描述在UDDI中心的使用。为更好地实现高效、准确的服务发现提供了基础。

(3)服务语义发现的研究

根据智能化和高效性的要求，研究怎样基于语义描述和本体论对广告服务描述与请求服务描述进行服务发现。服务的匹配是服务发现的一个关键问题。目前，为了提高服务发现过程中服务匹配的能力，许多方法都考虑有效利用本体论技术，对服务进行语义匹配。本文在扩展服务描述的基础上，提出了语义匹配原理及LF-MCH算法，对其做了详细的阐述。LF-MCH该匹配方法从本体概念的语言特征和背景特征两方面进行匹配，计算语义相似度。接着基于这个匹配算法，提出了LF-Grid 网格服务发现三阶段算法，充分利用了网格服务中存在的潜在的语义，除了对服务功能的操作、输出和输入参数进行语义相似匹配外，又增加了对前提条件和效果的语义匹配，很大程度上提高了服务检索的查准率和查全率。

6.2. 进一步工作

由于时间和其他因素还存在一些遗留问题和不尽人意的地方，待进一步改进，主要体现在以下几个方面:

(1) 服务发现只涉及单一服务，在以后的工作中需要将服务组合考虑进去;

(2) LF-MCH语义匹配算法从语言特征和背景特征分析计算语义匹配度，将它们同等看待，但在现实中，这两个特征对语义匹配度的影响程度是不同的，下一步工作要权衡两个特征的影响程度，赋予不同的权重;

(3)下一步的研究中，应该从更具体的网格技术入手，深入研究本体、网格、语义网、语义网格的理论和技术，在信息资源的共享与协同方面做更多的努力，并对GT4做进一步研究，对现有模块进行功能上的实现，真正将网格技术运用到实际中。

参考文献

[1]http://www.grid.org/home.htm

[2]Web Service. Reference:http://www.w3.org/2002/ws/

[3]Burstein M,Hobbs J and Lassila O.DAML-S:semantic markup for Web services[C]. In Proc.of the International Semantic Web Workshop,2001.

[4]Deborah L, Harmelen F. OWL Web Ontology Language Overview [EB/OL]. http://www.w3.org/TR/owl-features,2003.12-15

[5]The OWL Services Coalition. OWL-S: Semantic Markup for Web Services (EB/OL]. http://www.daml.org/services/owl-s/1.0/owl-s.html,2003.

[6]Anupriya A, Hucb F. Concurrent Semantics for the Web Services Specification Language DAML-S[C].In Proceedings of the Fifth International Conference on Coordination Models and Languages,York,UK,2002.

[7]Ankolekar A,Burstein M. DAML-S : Web Service Description for the Semantic Web[C].
In The First International Semantic Web Conference(ISWC),2002,06 :348-363.

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

[9]Paolucci M ,Kawamura T. Importing the Semantic Web in UDDI. Proceedings Web Services[C].E-Business and Semantic Web Workshop,CAiSE 2002,225-236.

[10]Paolucci M, Kawamura T. Semantic Matching of Web Services Capabilities[C]. In Proc. of the International Semantic Web Conference (ISWC2002),Sardinia, Italy,2002.

[11]Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. "Semantic Matching of Web Services Capabilities." Forthcoming in Proceedings of the 1st International Semantic Web Conference, 2002.

[12]Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D. and Hofstede, A. Towards a Semantic Framework for Service description. In Proc. of the 9th Int. Conf. on Database Semantics, Hong-Kong, April 2001. Kluwer Academic Publishers.

[13] Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. 2001. A Semantic Web Approach to Service Description for Matchmaking of Services, Proceedings of the International Semantic Web Working Symposium (SWWS)

[14]Mario Cannataro, Domenico Talia. Semantics and Knowledge Grids: Building the Next-Generation Grid. IEEE INTELLIGENT SYSTEMS, 2004, 19 (1): 56~63.

[15]宋丽华，王海涛 中间件技术与网格计算.信息技术与标准化，2004, Il: 50-53

[16]I .Foster, C.Kesselman, J.Nick, S.Tuecke. The Physiology of the Grid:An Open Grid Services Architecture for Distributed Systems Integration. http: //www.globus.org/research/papers/ogsa.pdf

[17]徐志伟,冯百明,李伟. 网格计算技术.电子工业出版社:北京，2004.5

[18]Borja Sotomayor. OGSA/OGSI,WSRF&Globus. http://people.cs.uchicago.edu/~borja/lectures/towards_service_oriented_grid.pdf

[19]http://tech.ccidnet.com/pub/series/s64.html

[20]Geoff Coulson. What is Reflective Middleware[EB/OL]. http:/ /dsonline.computer.org/middleware/Rmarticlel.htm, 200105

[21]李琪林，刘强，周明天.论中间件技术及其分类.四川师范大学学报，2001,24(6): 657-660

[22]秦颇，高文，储方杰.中间件技术研究.计算机应用研究，2003 (8):34-38.

[23]骆剑承，周成虎，蔡少华等.基于中间件技术的网格 GIS 体系结构.地球信息科学，2002,9 (3) :17-25.

[24]http://www.globus.org/toolkit/

[25]Stelling P , Foster I, Kesselman C , Lee C , Laszewski von. A Fault Detection Service for Wide Area Distributed Computation. Proc.7th IEEE Symp. on High Performance Distributed Computing,1998,268-278.

[26]Joshy Joseph,Globus ToolKit3.0 and OGSI architecture-overview. http://www-106.ibm.com/developerworks/grid/library/gr-gt3/index.html

[27]Foster I. Information Service in the Globus Toolkit3.0 Release. Http:// www.globus.org/mds,2003-10

[28]The Globus Toolkit3 Programmer's Tutorial. http://www.casa-sotomayor.net/gt3-tutorial-working/

[29]Foster I, Roy A, Sander V. A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation.8th International Workshop on

Quality of Service,2000.

[30]W3C. OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features, 2004-2-10

[31]http://gdp.globus.org/gt3-tutorial/multiplehtml/apas01.html

[32]Duftler, M.J., Mukhi, N.K., Slominski, A. and Weerawarana, S. Web Services Invocation Framework, 2001.

[33]http://www.daml.org/services/daml-s/0.7/

[34]庞丽平，董兴昌，邹德清，金海.基于 UDDI 的两级网格服务注册系统.计算机工程与科学 .2004 ,2 6(11):80-86

[35]http://www.uddi.org/specification.html

[36]http://www.w3.org/XML/Schema

[37]http://www-128.ibm.com/developerworks/cn/xml/soap/index8.html

[38]柴晓路，tModel 的用途及结构详解，http：//www-900.ibm.com/developerWorks/cn/webserviceslws-tmodel /part1/index.shtml

[39]http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[40]http://www.geneontology.org/

致谢

衷心感谢我的导师李庆忠教授，在我攻读研究生期间，李老师从各方面给予我热情的帮助和支持。李老师的确具有非常精深的专业知识和严格认真的研究精神，尤其是他对待研究的态度更令我佩服。尽管已经拥有了显著的成就和荣誉，李老师依然坚持不懈的学习专业知识，了解专业领域内的最新研究进展。从他身上我学到了严谨踏实的工作作风，学会了在科研中分析问题、解决问题的能力。在论文撰写期间，李老师给予我无微不至的关怀和细心的指导，为我提出了很多好的建议，使我得以顺利完成论文。

感谢孟祥旭院长等院领导，感谢他们为我提供了一个良好的学习环境，使我最终顺利完成论文。

感谢潘鹏老师，感谢他在撰写论文过程中给我以无私的帮助；感谢同学王栋及师弟师妹们的帮助，正是在共同的努力下才使论文得以顺利完成。

感谢我的父母，感谢他们始终如一的支持我，给予我生活上无微不至的关怀和思想上的巨大鼓励，使我在人生道路上不断成长。感谢我的爱人，感谢他对我的巨大鼓励和始终如一的支持。

感谢所有关心和帮助过我的朋友。

攻读学位期间发表的学术论文目录

1. 李霞，李庆忠，LF-Grid 服务描述的语义扩展，web 信息系统与技术，清华大学出版社，160-166。