网络制造环境中服务组合的研究

学　科：计算机系统结构

研究生签字：郭永社

指导教师签字：

摘　要

随着网络和制造业的迅猛发展，企业内的系统集成已不能满足密集型业务的需求；Web 服务是一种新型的松耦合的，跨平台的分布式计算技术，采用 Web 服务技术进行网络化制造和生产，可以实现网络化制造环境下跨企业协同制造过程中资源、数据、应用、能力等的有效集成。但传统的 Web 服务描述由于缺乏语义信息而不能被计算机理解，从大量服务中匹配所需服务并其进行组合的实用性差，为解决制造服务的匹配与组合问题，本文引入了语义网，通过为制造服务添加语义标记，实现基于语义面向服务的网络化制造，服务的匹配与组合可由计算机程序自动完成。

文章以网络制造环境为应用背景，对基于语义的 Web 服务组合进行深入的研究，首先，提出了一种语义 Web 服务组合的模型。Web 服务是能实现某种功能软硬件的封装，SOA 可用 Web 服务实现，它是一种粗粒度、松耦合的软件架构，其服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。本体是描述概念及概念之间关系的概念模型 ，通过概念之间的关系来描述概念的语义。该模型通过在 SOA 下的 Web 服务的体系结构中引入本体，使得 Web 服务组合具有语义功能。

然后，对语义 Web 服务的一种匹配算法进行了改进。在对语义 Web 服务的描述、发现、组合和执行进行深入分析和研究的基础上，将原有的一级匹配算法改进为两级匹配算法，使匹配过程更加精确化。

最后，以网络化制造为项目背景，对该模型进行实证。建立了网络制造环境下的网络制造资源领域本体，在本体上进行了推理功能的验证，以网络制造流程对原子服务进行组合，用实验来验证该模型的有效性和合理性。

关键词：语义网；语义 Web 服务；本体；OWL-S；服务匹配；　服务组合

Research on Web Services Composition in Network Manufacturing Environment

Discipline: Computer System Architecture

Student Signature: Guo Yongshe

Supervisor Signature: Wang Changyuan

Abstract

With the rapid development of network and manufacturing, system integration inside the business enterprise already cannot meet intensive business needs. Web services is a new type of loose coupling, cross-platform distributed computing technology. Effective integration of resources, data, application and ability of different enterprises under network manufacture environment can be realized by using Web services technology for networked manufacturing and production. But lack of semantic information the description of traditional Web Services cannot be understanded by computer, matching and compositing services from a large number of services is difficult to finish. In order to solve the matching and composition problems of manufacturing services, this article introduces the semantic web, and realizes network manufacturing based on semantic service by adding semantic markup for manufacturing services. By using the technologies, services of matching and compisiton can be finished automatically.

Under network manufacturing environment, the paper studies the semantic Web service compositon. Firstly, a semantic service compostion model is put forward.Web service is a functional encapsulation of hardware and software package. SOA(Service Oriented Architecture) can be realized by usable Web service, and it is a kind of coarse granularity, loose coupling software architecture.The different services communicate by simple, accurate definition interface do not involve the underlying programming interface and communication model. Ontology is a conceptual model to describe properties and relations between concepts. Semantic is described through the relation among the concept. The model make Web services composition contain semantic function by adding ontology in SOA architecture.

Then, a kind of exist matching algorithm of semantic Web service was well

improved. Analysis and research on the semantic Web service description, discovery, composition and executive has in-depth been done, and the original matching algorithm was improved to more level and made matching process more precise.

Finally, this model in the background of network manufacturing environment was confirmed, and a network manufacturing resource domain ontology of network manufacturing environment was established, and a kind of reason function on the resource domain ontology was verified. These atomic services was composited to complex service on the base of business process of network manufacturing, and the validity and rationality of the model was verified by doing a lot of experiments.

Key Words: Semantic Web; Semantic Web Services; Ontology; OWL-S; Service matching; Service composition

目 录

1 绪 论

在实际应用中，网络上发布的服务大多数都存在结构简单、功能单一的缺陷，无法满足企业复杂应用的需求。如何有效地组合分布于网络中的各种服务，实现服务之间的组合集成，形成功能强大的企业级服务流程以完成企业的商业目标，是在 Web 服务发展趋势中的一个重要步骤，这也是 SOC 与 SOA 能否成功应用和实施的关键。服务组合为网络制造环境提供了企业间协同支持，在大范围的组合需要提供基础协议、模型库管理、公共服务、使能工具和系统管理等功能，从而为企业提供透明、一致的信息访问手段，方便了不同企业间的软件和制造资源的集成，进而建立具有特定功能的网络制造系统。当前，Web 服务组合成为网络化制造的关键技术和主要研究方向。

Web 服务技术作为面向服务计算(SOC)和面向服务架构(SOA)的主要实现技术，已经得到广泛的应用。然而目前传统 Web 应用技术解决的问题是如何让人来使用 Web 应用所提供的服务，而 Web Services 则要解决如何让计算机系统来使用 Web 应用所提供的服务。而在 Web Services 中加入语义的支持，使得计算机之间能够理解互相通信的内容，从而实现自动化和智能化。

本文针对网络制造环境中系统集成的现状及存在的问题，在充分借鉴与吸取当前广泛应用的先进制造理念，结合 SOA 及 Web Services、本体及语义网、语义 Web 服务等技术，对网络制造环境中 Web 服务组合的框架和语义 Web 服务组合的关键技术进行了深入的研究，如以网络制造环境为背景基于语义的 Web 服务组合框架的研究、语义 Web 服务的发现算法和基于图论的 Web 服务组合算法的深入研究。

1.1 课题的研究背景及意义

1.1.1 研究背景

网络制造环境的研究是实施先进制造核心的使能技术；同时，网络制造环境又是先进制造技术与网络技术结合的产物，是先进制造模式在因特网环境下的发展和应用。在这种环境中，结合语义 Web 技术和 Web 服务技术，使得网络制造环境中的不仅具有语法上的组合功能，而且也具有语义上的组合功能。

目前的Web信息的搜索过程主要是依赖于关键词，且机器不能在语义上“解释”服务请求者发出的请求内容，所以通过搜索引擎得到的结果往往就不是服务请求者想要的内容，也无法通过对语义上相近的检索结果返回类似的结果[1]。对于这种情况的解决方案就是：使用让计算机能够解释的形式把 Web 上的信息

表示出来，在结合运用本体知识、描述逻辑推理等，使得 Web 上的信息具有明确的语义，而这种语义功能达到计算机与人或其他计算机进行更好的理解，达到真正语义检索的作用和目的。

Web 技术的最新发展是 Web 服务，它可以在不同的协议、不同的系统平台之间具有互操作性，使用 XML 的形式进行消息的传递，实现与其他应用程序之间的数据交互，进而在 Internet 上实现不同应用程序之间的远程过程调用。Web 服务支持基于整个互联网的协议， 使用到的协议如 HTTP、XML、SOAP、WSDL、UDDI 等都是国际标准化的协议，这些协议它们与用到编程语言、对象模型和操作系统的选择无关。Web 服务是 SOA 的一种实现方式，它不涉及具体平台和语言的软件架构，具有强大的生命力和广泛的应用前景，当前的研究工作的热点主要集中于 Web 服务的描述、发现、组合和执行等技术[2]。

语义 Web 服务由语义 Web 和 Web 服务两种技术相互取其优势结合而产生的。语义 Web 是以本体为基础进行研究，加入语义功能，作为 Web 来理解语义功能的基础，也具有了知识间的推理功能；而 Web 服务利用其可以跨平台的系统集成使得语义 Web 服务具有了自动化和智能化。

1.1.2 研究意义

在网络制造环境中，对语义 Web 服务组合的研究意义深远而重大，Web 服务需要有对语义功能的需求，避免原有的只基于关键词的单一搜索要求。第一步，对已有服务检索和发现不能仅仅依赖于关键词，应是从服务所提供的功能上进行检索，这样才能找到确实真正需要的服务，因为服务的功能不是单纯的依靠若干关键词来完整表达；第二步，对于服务的使用者，对服务进行调用和使用已有的服务进行更大功能的组合，需要在基于语义的互操作性之上，和服务之间必须达到能够理解和交互的信息；第三步，即使一个 Web 服务执行后的输出参数与另一个 Web 服务执行前的输入参数的名称和类型都相同，也不能将它们直接进行串行连接，因为无法保证两个参数的在含义上所表达的信息是相同的。已有的 Web 服务所采用的协议仅仅在语法层面上限定了服务描述，对服务在语义层面上的描述能力具有局限性，对于进行 Web 服务的组合就没有灵活性和自动性[3]。

语义网是采用本体的技术来描述 Web 上的各种资源，网络中的信息采用一种知识化、结构化、语义化的明确的共享方式来进行表示，本体的技术使得 Web 上的信息在计算机上具有了共同理解的语义基础，为异构分布式信息有效访问和搜索提供了更大的灵活性。本体是基于 XML、RDF/RDFS 和 OWL 语言来进行构建的，再运用描述逻辑的推理规则，进而实现了在语义层面上的知识表示和推理功能，从而达到了使计算机在语义层面上的理解和处理要求[4]。

语义Web服务是使用具有语义功能的描述服务语言来表达中的服务中的语义，进而Web服务成为了计算机可以理解化的实体，在功能上就可以支持服务的自动发现、调用和组合。目前语义Web服务主要方法是利用本体来描述Web服务，然后通过这些带有语义信息来描述Web服务，达到了服务的自动发现、调用和组合。

语义Web服务相比于传统的Web服务，它的优点主要为：服务是计算机可理解的；语义上无歧义，增加了服务发现的准确性；为自动发现，调用、组合和互操作提供了便利；环境适用能力强和适用的范围广等。结合Web服务的松散耦合性和语义Web服务提供的语义信息，可以使服务自动地匹配、发现、调用和执行[5] 。

实现语义Web服务的关键技术是对Web服务进行语义化得描述。目前应用最广的描述Web服务的本体语言是OWL-S，在语义的描述上使得服务提供者与服务请求者都适用统一的本体化共享公共语义的形式化表达，实现机器之间的互操作，为信息的自动化和智能化的处理提供了基础，使得服务的选择、匹配具有一定的智能性，对在实现语义方面的检索具有现实意义。

1.2 目前国内外的研究现状

目前国内外学术界和企业界对语义Web服务的技术都展开了详细而深入的研究。语义Web服务的描述使用OWL-S语言使得其应用不仅具有Web服务的松散耦合和高度集成能力等特点，而且具有语义Web提供的语义信息，在进行系统服务集成时能够自动的发现、匹配、组合和执行Web服务。在语义Web服务的发展过程中，W3C的语义Web小组成员和世界各地的研究机构和科研人员也都在进行积极深入的研究工作，他们在标准规范的制定和理论及实践方面的探讨都在做着不懈的努力[6]。

在理论方面的研究上，语义Web服务研究工作主要集中在语义的匹配算法研究和服务组合策略研究。S.Narayanan等利用Petri网理论和情景演算的方法在语义Web服务描述本体语言的操作语义方面进行了深入研究；B.Benatallah等在基于请求重写的Web服务发现问题的方面上，研究Web服务自动发现的推理功能，并且提出了一个灵活的服务匹配算法来提高服务的自动组合；D.wu等在自动的OWL-S Web服务组合问题方面进行了研究，结合SHOP2(Simple Hierarchical Ordered Planner2)来实现自动服务组合问题，并且通过SHOP2规划来进行语义Web服务的服务组合实现[7]。

在项目实现方面上，研究的主要方向是语义Web服务的自动发现和自动组合。在美国DARPA组织和欧洲联合会共同联合资助的项目SWSI(Semantic Web

Services Initiative)，就是为了建立一个支持语义 Web 服务功能的集成框架，为服务的检索、发现、调度和监控提供集成框架，满足自动化和智能化的系统集成要求。还有欧洲语义 Web 高级服务开发组，他们的提供 SWAD-Europe(Semantic Web Advanced Development of Europe)项目也是在积极推动语义 Web 服务成为网络发展的主流[8]。

在开发工具方面上，语义 Web 服务的开发环境上主要是针对本体编辑器和本体的知识推理方面，在服务组合方面主要是服务匹配的自动化进行研究。卡耐基.梅隆大学的软件智能实验室研究出用于开发、发布和执行语义 Web 服务的一系列工具，主要的研究成果是有语义 Web 服务发现的 Web 服务匹配器和自动化 Web 服务调用的 OWL-S 虚拟机[9]。马里兰大学研究了用于 WSDL 与 OWL-S 的转换的软件包 OWL-S API(最新为 owl-s-1.1.0-beta) 、马耳他大学研究了的 OWL-S 本体编辑工具 OWL-S Editor、可以作为插件集成在 Protégé 中；斯坦福大学在本体的编辑工具方面开发的本体编辑平台 protégé（最新为 Protégé_4.0）[10]。

虽然目前对语义 Web 服务的研究进一步详细而深入的研究着，但各研究团队对语义 Web 服务的研究工作和重心主要体现在语义 Web 服务的描述和组合流程及算法上，较少地从语义 Web 的整个流程的角度来考虑 Web 服务的语义发现匹配和组合问题，这正是本论文所要重点研究的工作。本论文借助于最新的 Web 本体服务描述语言 OWL-S，对 Web 服务的语义描述框架、语义 Web 服务的检索和发现匹配、语义 Web 服务的组合进行研究。

1.3 课题的主要研究内容和重点

1.3.1 研究主要内容

研究的主要内容分为以下几个部分：

1）对语义 Web 服务体系结构的研究

语义Web服务组合的层次OWL-S框架由五层组成：传输层，XML消息层，服务描述层，实现层、服务发布查找和组合层。在Web服务组合的层次结构中，传输层、XML消息层、服务描述层与传统的Web服务协议一致；而服务发布查找、组合层则采用OWL-S框架。研究的重点在上三层中，从服务描述层如何描述语义Web服务，如何查找语义Web服务，如何匹配语义Web服务，如何组合语义Web服务等。

2）语义 Web 服务组合的动态组合过程研究

语义 Web 服务的动态组合过程主要涉及以下几个方面：服务注册：将所开发的语义 Web 服务在 UDDI 注册中心注册；服务建模：Web 服务本身采用 WSDL

文件来描述的，不同服务商对开发的 Web 服务中的 WSDL 元素可能采用不同的表示方式，所以，在组合前，需要用领域本体进行统一建模，消除语义差别，满足后面组合的需要；服务组合：用户根据需要对统一建模的服务进行自动或手动组合，形成 Web 服务组合方案，然后把形成的组合服务转换成可执行代码；组合服务的查找与执行：组合引擎接受用户的查询并返回所需的 Web 服务。

3）对本体及语义 Web 服务的研究

语义 Web 服务的建模，就是基于语义对 Web 服务的属性、功能和结构进行描述，使用户可以自动的发现、选择、使用和组合服务。借鉴动作理论对服务的控制流和数据流等逻辑层次的信息进行语义建模，简单服务用带前提和结果原子动作描述，控制流用复杂动作的组合服务描述，数据流用服务间的依赖关系描述，最后提出了基本语义 Web 服务组合模型。

4）结合图论知识对服务组合算法研究

服务组合的方法及算法：按照语义功能将服务组合归结为一个图论的问题，按照图论的方法进行服务组合，图的深度和广度算法，及最少生成树、最短路径及拓扑排序等。

1.3.2 研究重点

本文主要在以下几个方面进行重点的研究：

1）语义 Web 服务组合模型的建立

在面向服务架构（SOA）下，在充分学习和掌握传统的 Web 服务的体系结构和 Web 服务的运作模式，全面理解本体的技术和语义 Web 等的相关理论知识，查找和参考各种文献资料后，在归纳整理的基础上，借鉴已有框架的优点，避免它们的缺点，尽可能试图建立一种基于语义 Web 服务的组合模型。

在这个研究过程中，主要是对已有 Web 服务体系结构中的 Web 服务加入语义功能，是 Web 服务本身内部带有语义功能，而且在查找的时候带有语义功能查找。进而，在语义检索出来的结果上进行匹配和组合，匹配不断进行，直到达到满足组合服务提供的需求，模型中对组合出的多种方案进行最优匹配，选择最优的组合方案。这些都有研究的重点和研究的难点。

2）服务匹配算法的研究

已有的服务匹配算法主要是通过 UDDI 服务注册中心来实现的，UDDI 注册中心负责服务描述文件的注册、管理和发现服务，这种机制的缺点就是对服务的描述单一，服务匹配主要采用关键词匹配、难以满足服务匹配的要求。在 Web 服务的描述中增加语义信息，利用本体对服务的相关信息进行概念、定义上的统一认识，明确服务之间的关联，提高服务匹配质量。这是本文的研究重点之一。

3）模型验证及实验设计

模型的建立是否正确，匹配算法的实施和组合模型的具体应用是否值得可行，只在理论研究上的研究是不够的。本文是在网络制造环境的背景下，以网络制造流程进行服务组合研究来验证模型的有效性和可行性。

1.4 本文的组织结构

本论文在章节上分为六章，组织结构和各章的主要内容介绍如下：

第一章为绪论，介绍了网络制造环境下研究语义 Web 服务组合的研究意义，在详细分析了国内外的基本研究现状之后，指出本论文的主要研究主要内容和研究重点和难点，最后给出课题研究的整体组织架构和研究思路。

第二章为语义 Web 服务的基础关键技术的研究，介绍了语义 Web 服务定义及其组合的依据理论，并详细阐述了语义网及其体系结构，本体的概念、本体的建模、本体的形式化表达、本体的建模实例，基于 OWL-S 的服务组合框架等，深入分析 OWL-S 服务组合框架中服务组合的方法和思路，最后，具体介绍了 OWL-S 与 WSDL，UDDI 的关系。

第三章介绍了语义服务组合的思想和方法，详细分析了服务发现及匹配的方法，具体给出了实现的代码，在服务发现的基础上描述了服务组合的算法，并给出具体的实例进行分析验证。

第四章在前面研究分析的基础上，提出了的基于语义 Web 服务组合模型，提出了模型的整体框架，在分析各个模块功能的基础上，给出了整体的运作流程。

第五章为设计和实现部分，针对前面提出的基于语义 Web 服务组合模型和语义 Web 服务匹配算法进行了实验的验证，在得到实验结果的基础上，来分析所建了模型的适用性。

第六章为总结部分和进一步展望部分，总结了网路环境下服务组合的研究工作，并对下一步的进行深入研究的工作进行了展望。

1.5 本章小结

本章第一步首先分析了网络制造环境下服务组合的研究的背景和研究的意义，指出了传统的 Web 服务具有语法上的功能，在加入语义上的功能后来进行服务组合，增强了服务的发现和匹配功能。语义服务组已经成为 Web 服务网络中亟待解决的问题，第二步详细的分析了目前国内外在语义 Web 服务匹配和组合方面的研究现状，了解了要进行服务组合的关键核心技术。第三步，在前面分析的基础上，提出了本课题的主要研究方向、研究内容、研究的重点及研究

的难点，并搭建起整篇论文的组织结构，为以后的各章节的内容展开提供了坚实的基础。

2 语义 Web 服务及服务组合理论

2.1 Web 服务的技术架构

2.1.1 Web 服务的体系结构

传统的 Web 服务是建立在一系列开发的万维网标准之上的一种 Web 应用程序，其基本的协议规范包含 XML、SOAP、WSDL 以及 UDDI。Web 服务是由一套协议栈构成的层次化体系结构，如图 2.1 所示，上一层需要下一层的支持。在安全、管理和服务质量上需要在各个层次上都有体现[11]。通过这样一个层次分明的架构，Web 服务希望达到一个目标：实现动态的跨平台、跨协议的应用系统平台的集成，将网络应用推向一个更广范围内的实用阶段。

Web 服务主要利用 HTTP 和 SOAP 协议使服务数据在 Web 上传输，SOAP 通过 HTTP 调用服务对象执行远程功能调用，Web 用户能够使用 SOAP 和 HTTP 通过 Web 调用的方法来调用远程对象[12]。如图 2.2 所示，具体地说就是客户根据自己的要求，使用浏览或直接获取两种途径查找通过 UDDI 发布在网上满足自己需要的 Web 服务，获得这些 Web 服务的 WSDL，生成客户端代理。使用该客户端代理就像使用本地组件，而通过 SOAP 协议去激活远程方法，实现数据访问。

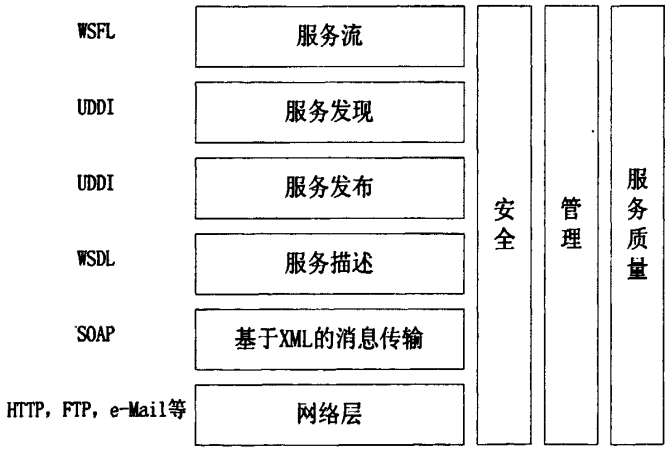

图 2.1 Web 服务协议栈

2.1.2 Web 服务的运作模式

Web 服务是 SOA 架构的一种具体应用，Web Services 是独立的，模块化的

应用，能够通过 Internet 来描述，发布，定位和调用。Web Services 的体系结构中包括三个角色，服务提供者，服务请求者和服务注册器，角色之间有三个操作：发布、查找和绑定。

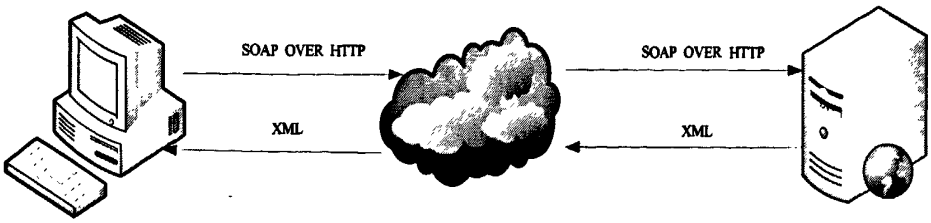

图 2.2 客户调用 Web 服务的方式

典型的 Web 服务的运行模式如图 2.3 所示[13]，服务提供者开发一个通过网络可以被访问到的服务，然后将服务的描述注册到服务注册器或发送给服务请求者；服务请求者通过查找动作在本地或服务注册器中检索服务描述，找到结果后，通过服务描述中的绑定就可以使用该项服务。总的来说，在 Web 服务架构中，使用 WSDL 来描述服务，UDDI 来发布、查找服务，SOAP 则来执行服务调用。

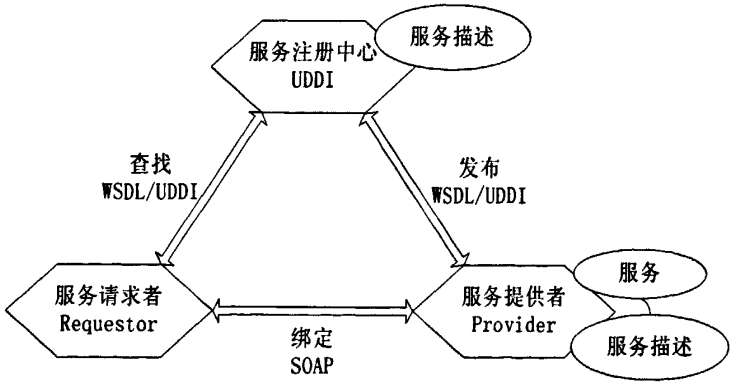

图 2.3 基于 SOA 的 Web 服务体系结构

Web 服务的工作过程是一种分布式计算的体系结构。通过设计和指定 Web 服务来促使跨平台的程序通信。

2.2 语义网及其关键技术

2.2.1 语义网及其体系结构

语义 Web 的目标是为了解决在不同应用、企业和社区之间的互操作性问题，

这种互操作性是通过语义来保证的，而互操作的环境是在不同的应用环境、进行动态开发的，得到全球化语义 Web；是为因特网上的信息提供具有计算机可以理解的含义，从而满足智能的软件代理（Agent）对因特网上异构、分布的信息的有效检索和访问，实现网上信息资源在语义层面上的全方位互联，并在此基础上实现更高层次、基于知识的智能应用。简单的说，语义 Web 项目就是对因特网中的信息的一种新型的组织方式。

在 2000 年的时候， 语义网的创始人 Tim Bermers-Lee 在世界 XML 的大会上，正式的提出了语义 Web 蓝图和体系结构[14]。其体系结构如图 2.4 所示，整个语义 Web 体系结构分成七层：

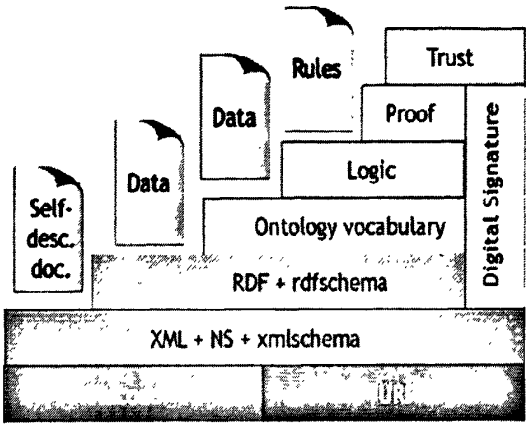

图 2.4 语义网的体系结构

1）第一层为 Unicode 和 URI 层

Unicode 是为了使计算机应用国际化应用而产生的，它是一种在计算机上使用的字符编码，为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转化和处理的要求。Unicode 编码具有的特性不但支持所用通用语言的组合，更能进行统一的查询和检索。而 URI 的作用是用来唯一标识网络上的特定的概念或资源。这一层为整个语义网的扩展的建立坚实基础，其中 Unicode 解决了处理统一所有资源的编码问题，URI 解决了因特网上资源的唯一性标识问题。

2）第二层是 XML、 NS 和 XML Schema 层

第二层的主要作用是利用 XML 来提供了结构化的数据表示。XML 本身结构化的特点为异构平台之间的数据交互的提供了有利的保证。NS 即为 Namespaces 的缩写，是来解决命名的名称空间的冲突，保证访问资源的唯一性。 XML Schema 用来描述 XML 文档的结构原型，给出了 XML 定义规范。

3）第三层是 RDF 和 RDF Schema 层

RDF(Resource Description Framework)是一种数据结构模型，如同关系数据库中的实体-关系模型，主要是用来编写关于网络对象的简单陈述句。RDF 的数据模型是基于 XML 语法规范的，但不依赖于 XML 的词法。RDF Schema 提供将对象组织成层次结构的建模原语，其中，关键原语是类和属性、子类和子属性关系、定义域和值域限定等。

在不同元数据标准下建立一种能充分利用各种元数据的框架，利用该框架能进行基于 Web 的数据交换和利用。XML->RDF->RDF Schema 是一个不断发展的过程，XML 是一种标准化的元数据语法规范，RDF 将语义加入其中，以三元组的形式表示出来，达到的目的就是提供了词汇融入的语义方法和方式，利用该方法，多种词汇可以有机地组合来系统的描述 Web 上资源。

4）第四层是 Ontology 层

建立在 RDF/RDFS 基础之上，增加了属性的局部辖域、类的不相交、类的布尔组合、基数约束和属性的特殊性质如自反性、传递性、对称性等的本体层，这些都是为解决 RDF/RDFS 的描述能力不足而扩展的。该层使用 OWL 网络本体语言用于描述知识的语义以便提供给机器来处理，本体层的 OWL 是以 RDF 和 RDFS 为基础，借助 OWL 到逻辑的映射提供了形式语义和推理功能，为此使用了谓词逻辑和描述逻辑，这一层是语义网体系结构中的核心层。

5）第五层是 Logic 层

前几层主要实现对 Web 资源进行描述和表示，Logic 层可以根据特定的规则对语义描述的 Web 资源进行推理。其中的有描述逻辑的推理算法包括结构包含算法和 Tableau 算法，其核心是针对不同的描述逻辑构子，定义相应的算法来展开规则进行推理。

6）第六层 Proof 层和第七层 Trust 层

Proof 层主要是实际的演绎过程、执行逻辑层产生的规则，来验证和判别是否能得到该层的证明。Trust 层是随着数字签名和其他种类知识的使用而出现，信任将按照与万维网自身同样的分布式加以组织，它处于分层结构的最顶端，信任是一个高层而且至关重要的概念。

2.2.2 本体的概念

Ontology 一词，从哲学角度上理解为是对世界上存在的客观物系统的描述；是对现实的抽象本质[15]；也是一种系统化的描述或说明。本体和元数据不同，本体解释资源概念间的相互关系，元数据解决资源的语义描述问题。本体包含概念化、明确、形式化和共享等几个方面的内容。

概念化：是通过对客观世界的采取抽象的方法得到一些相关概念模型，这

些概念模型是一组概念、定义和关系；是对事实结构初步得出的约束规则。

明确：无异议的定义概念和其上的约束。

形式化：以能被计算识别的方式表示信息。

共享：本体中包含的知识能大家在共同理解的基础上达到共识和认可的。

本体主要包括：类、关系、函数、公理和实例等这5类元素。

和面向对象的技术的方式一样，类是对象的抽象，采用框架(Frame)结构描述，包括概念的名称、关系和用自然语言对其进行描述[16]。

关系表示类与类之间的关联，这种关系具有开放性，随着本体类的增加而可以随时加入，定义的表达形式为R：A1×A2×...×An。表示类A1，A2，...，An之间存在n元关系R。

函数是一种特殊的映射关系来表达类之间的关联程度，可以通过前n-1个元素来确定第n个元素。形式定义为F：A1×A2×...×An→An。

公理表示的是事实规则，在逻辑上表达为永真断言，用来进一步限定和规范“概念”和“属性”，相当于数据库技术中的完整性约束[17]。

实例是等同于面向对象技术中的对象，是对属于某个类的对象具体化。

2.2.3 本体的建模

网络制造环境是将先进制造技术和信息技术相结合，快速形成虚拟网络企业联盟，充分利用社会资源，协同开展产品开发和设计、制造、销售、采购、管理等产品生命周期业务活动，以及时、快速地响应和适应市场和用户需求变化，提高企业群体竞争优势[18]。

网络制造资源管理是网络制造环境中的关键部分。它通常是指完成产品生命周期的所有生产活动的软硬件元素即人、财、物的总和。互联网中的Web制造资源作为一个庞大的信息资源联合体，没有一个统一的规范，企业之间对资源的描述还存在着差别，给互联网上制造资源的发现和查询造成了困难。为解决这个问题，引入语义领域本体对资源进行描述，使得资源之间在具有语法性的基础上，还具有语义性，构造制造资源领域本体模型。

概念层和元数据构成了制造资源本体的语义元数据模型[19]，如图2.5所示，在概念层上主要是各种共享资源概念实体定义的有机联系组合的集合，主要由资源共享概念模型，资源间连接属性和资源服务等构成。制造资源本体模型中出制造资源的概念实体、资源的约束条件的关系集构成。元数据层是概念类的属性基础集，对概念进行整理归纳形成知识体系，以知识结构的表示和规范来约束这些可控词汇。

概念层对制造资源本体的类、关系、约束、公理、函数等进行描述，是一种网状结构。包括制造服务资源等，由语义关系和相关关系两种关系。资源间所具

有的基本关系用层次关系来描述，可以反映概念间的父子关系，同层概念间形成对等的兄弟关系。层次关系可以构成一个相对独立的概念族。

不同共享资源层次形成分类层次树，最高层为根节点，最底层为叶子节点，其关系用Is-a关系表达。例如学校里面的资源共享，可细分为教师、教授、副教授、讲师、计算机学院的教师，光电学院的教师等。相关关系存在于同层或不同概念族之间基本的横向联系。这两种关系可以将制造资源领域知识组成一个概念集，语义化作为领域初级本体。鉴于概念集中所包含的关系不能包括制造资源之间复杂的语义关系，实际中通常通过增删语义关系来实现领域本体的扩展或细化。约束给出了制造资源共享需要满足的条件，是描述制造资源和服务之间相关关系的一种机制。约束分为构建层和参数层两层。构建层表达两个资源类型的兼容性；参数层是通过参数之间的关系，来表达各类制造资源实例的存在性。

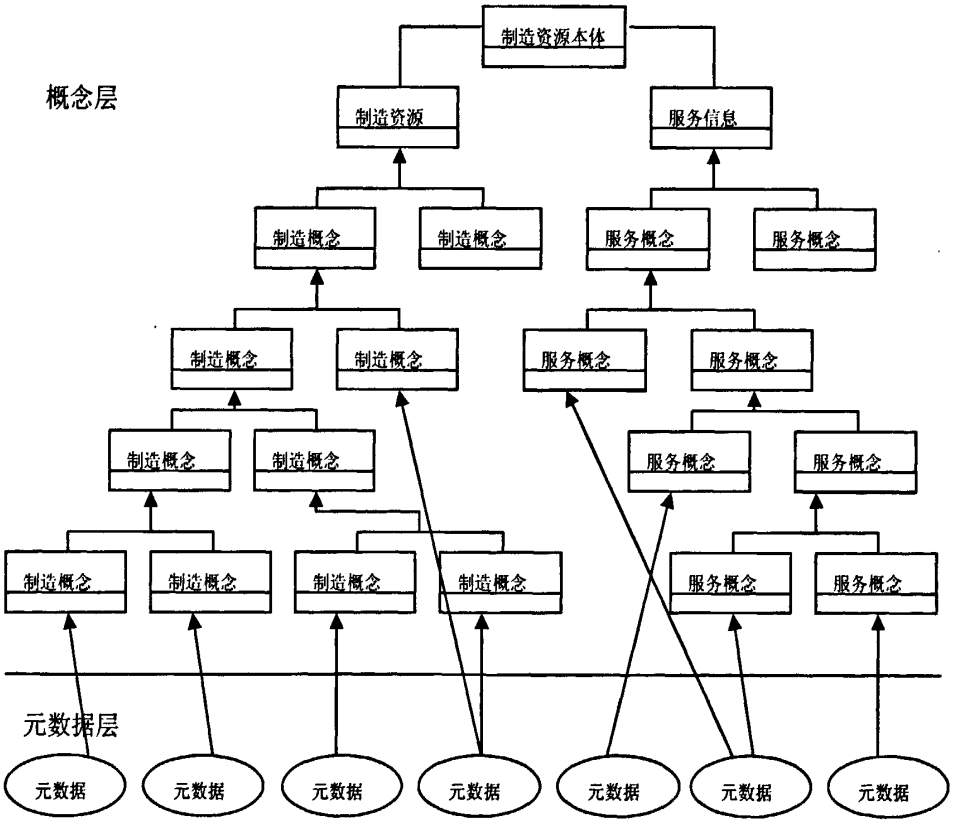

图2.5 制造资源本体的语义元数据模型

元数据层可以将特定应用的语义元数据集成到语义层次模型中。元数据是表达数据模型的数据，是数据的数据。它描述了数据模型中基本的概念、关系和约束的语义，还包括数据模型的建模过程、建模原则，为数据模型的建立和使用提

供了依据[20]。因此，在Web细粒度资源的语义应用中，引入了元数据，在下层形成概念层，这样实体的概念集合就可以直接应用到元数据层构成的元数据的属性集。概念层和元数据层的关系类似于面向对象中类和对象的关系，下层的元数据作为资源对象实例可以继承来自概念层类间的关系，通过这种方式就可以把离散的元数据单个概念可以系统化的带有表达丰富的语义关系。

2.2.4 本体的形式化表达

制造资源本体的形式化是为了使实际应用中的概念系统化以便让计算机处理起来更加规范，而进行形式化表达，在格式要求上提供了更加严格规范的表达，加强了机器的自动识别能力[21]。制造资源本体是有关共享网络制造领域资源数据知识表达上的主要概念、概念间的关系和概念知识之间的规则方法的综合体，与之对应的语义元数据也具有类似的关系，因为它是继承概念层的，对网制造资源本体的形式化表达形式如下[22]：

定义一. 网络制造资源本体结构的形式化表达采用四元组的形式来描述为，On_{ms} = (C，A，R，S)。C描述网络制造资源数据本体中的类概念，对应模型中的概念层。概念集合是制造资源中所有领域概念中的集合，形式化为：$C=\{C_{loc},C_{net},C_{ext}\}$，其中$C_{loc}$、$C_{net}$分别为为本地资源和远程资源概念的集合。A用来对制造资源本体中的概念集实例进行描述，指的是概念集的属性集合。属性集合 $A=\{a_i \mid i=1，2，\ldots，a_i \square DC\}$，其中DC为元数据的描述规范。R为描述概念间的逻辑关系集合，包括类与对象的关系、整体与部分关系，类之间继承的关系，属性联系之间的关系等。R定义了概念之间的关系集合，这种关系比面向对象的的关系更加松散，具有弱约束关系，便于以后扩展如新的概念的加入很容易通过这种关系而加入本体中。S为描述概念之间约束关系的规则集合，如互斥约束、自反约束、权限约束、关联约束等。

定义二. 网络制造资源元数据模型结构可表示为： MetaData = (On_{node},S,I,F_S^C,F_S^R,IC)。On_{node}是网络制造资源本体结点。S 是所有网络制造资源的对象集合，每个资源对象是由一个资源标识来唯一确定。I 是结点内所有的概念实例的集合。F_S^C是所有资源到对应概念实例映射函数的集合。F_S^R是所用资源对应概念关系实例映射函数集合。IC是资源对象实例特征约束集合。

2.2.5 本体的建模实例

网络制造资源需要建立以设备制造资源库、产品资源库、技术资源库和服务资源库的公共数据中心的基础设施上，为企业联盟提供有效资源集成与管理、企业之间可以灵活的调度和协同工作、支持网络化制造服务有机组合[23]。

为了有效的对本体进行存储和查询，需要把创建好的制造资源本体转化成

OWL描述，即为MR.owl文件，具体的解释如图2.6所示。在得到设备资源本体的OWL文件后，应用系统可以通过Jena来对OWL文档进行处理，Jena具有对OWL推理的多方面支持，它允许应用系统解析、创建和查询OWL模型[24]。在完成制造本体的语义构建工作后，获得一个集成的、相对比较完整的制造知识资源库，这个库是以语义的方式组织起来的，所以前面的工作是基于语义Web的制造知识的定义和集成管理。

企业信息和设备资源与服务方式的关系的关系式用属于或拥有的关系来表示，一个企业拥有一种设备资源，一个设备能属于一个企业，一种服务方式对应一种设备资源。企业信息和设备资源之间的关系是完全不相交继承关系；设备资源和机床之间存在继承关系[25]；制造资源、服务方式和信息服务时间存在类不相交关系，且它们同属于网络制造资源本体的类，是聚合关系。本体编辑工具使用 protégé, 它是基于 Java 语言，支持 XML/RDF,RDFS,OWL 等本体语言，提供了对本体的读入和编辑后的输出，其中对 OWL 文件的操作是借助了惠普实验室开发的 Jena[26]。

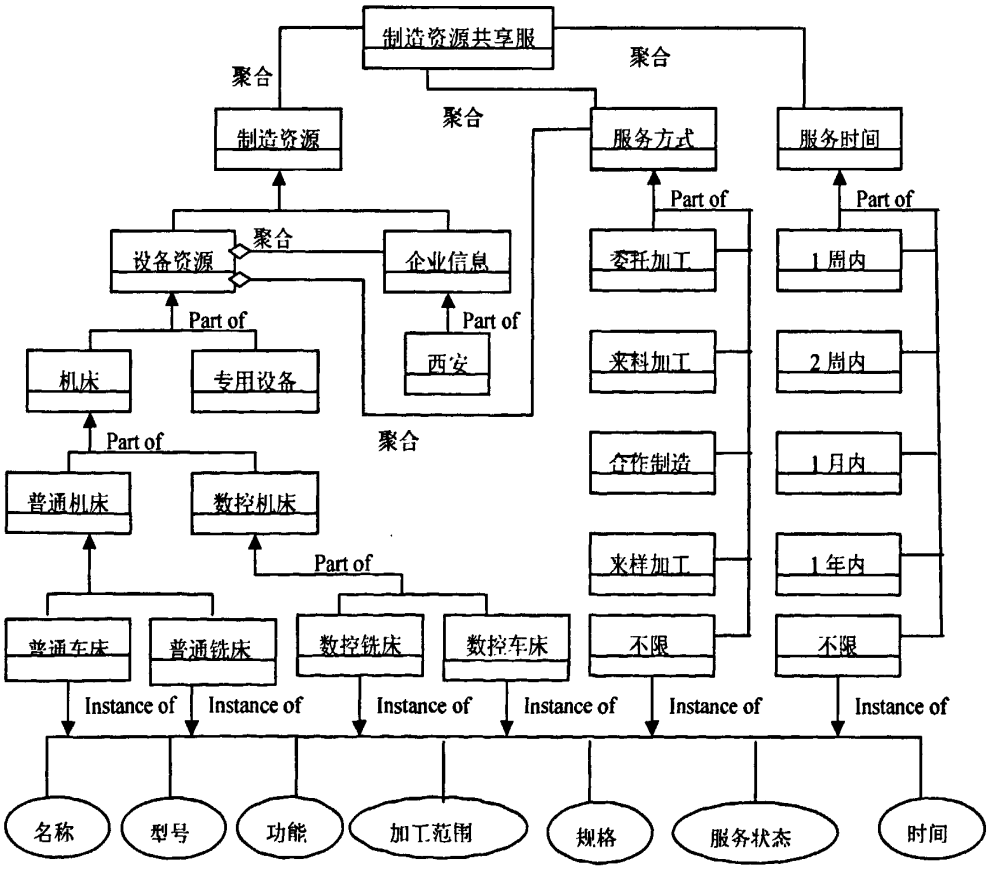

图 2.6 网络制造设备资源本体的语义模型

2.3 语义 Web 服务组合

2.3.1 语义 Web 服务

传统 Web 资源及 Web 服务的访问是通过基于关键字实现的，而语义 Web 通过本体的对现实世界的表达融入语义功能，从而实现机器对语义 Web 信息和数据的自动化处理。可以提供服务的资源是在所有的 Web 资源中最重要的，因为这里的服务既包括了网站能够提供的静态信息、而且还包括允许用户的请求发来执行某些操作或者变化[27]。语义 Web 服务是基于本体的语义网技术和具有动态调用 Web 服务的结合，语义 Web 服务研究的根本任务就是对现有的 Web 服务加入语义功能，使得 Web 服务可以被计算机所理解。

语义 Web 服务是语义 Web 和 Web 服务两者的有机结合，语义网有语义功能但没有 Web 服务的松散耦合，服务自治和服务独立等特性；而 Web 服务可以跨平台、跨协议和跨网络的功能来实现而没有语义功能。两者的结合，优势互补，使得语义 Web 服务成为 Web 发展的一个必然趋势，语义 Web 发展趋势如下图所示[28]。

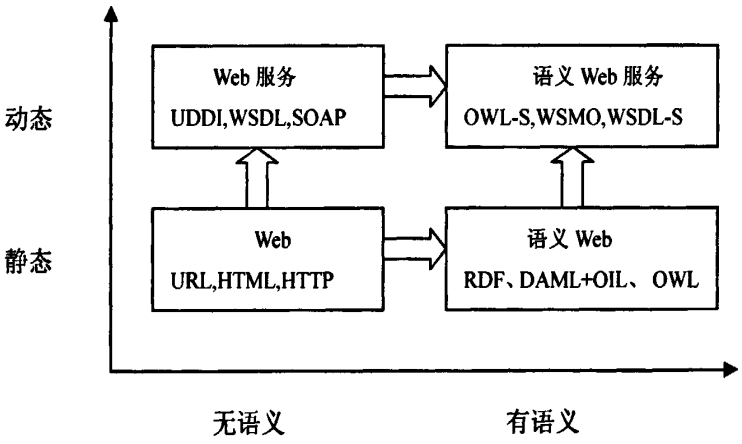

图 2.7 Web 服务发展的趋势

语义 Web 服务的重要的目标是:使得语义服务成为一种机器可以解释和理解的，避免传统的 Web 服务没有语义操作能力的局限性，为语义 Web 的发展提供更大的发展空间；支持在语义环境中下 Web 服务自动查找、发现、匹配、执行以及动态组合；通过领域本体和通用本体与推理技术，扩大整个应用范围。

2.3.2 基于 OWL-S 的服务组合框架

基于OWL-S的服务组合框架中语义Web服务组合的层次结构由五层组成:传

输层，XML消息层，服务描述层，实现层、服务发布查找和组合层[29]，如图2.8服务组合的框架所示。

在Web服务组合的层次结构中，每一层具有各自的特定功能，传输层负责进行网络传输，使用的一般使用的协议是HTTP、SMTP、FTP等；XML消息层服务是完成用户的请求和应答，是通过SOAP协议来进行描述的；服务描述层与传统的Web服务协议一致，WSDL语言描述的服务的基本功能：输入和输出，消息格式，服务的绑定等，并在原有的基础上利用WSDL到OWL-S的映射使的WSDL具有了语义功能；而服务发布查找、组合层则采用OWL-S框架，服务的发布是具有语义描述的Service Porfile向现有的UDDI中进行映射，使得服务查找具有语义功能；服务组合则是Service Model来根据服务之间的关系和业务之间的流程完成特定的要求来进行组合。我们研究的重点在上三层中，对服务描述层WSDL、服务实现层OWL-S Grounding和服务组合OWL-S Service Model及查找发布层OWL-S Service Profile等。

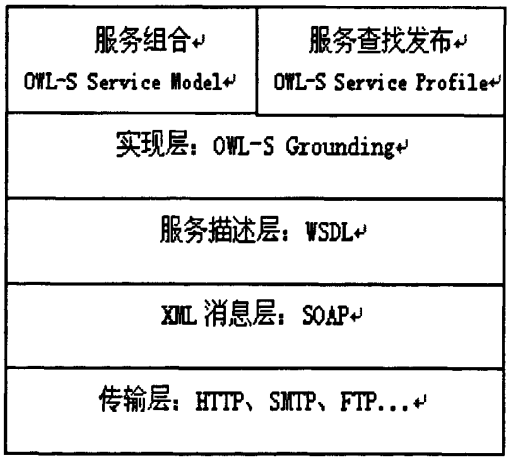

图 2.8 基于 OWL-S 的 Web 服务组合

2.3.3 OWL-S 顶层服务本体模型

面向语义的Web服务组合方法是通过在已有的Web服务中添加语义信息，如Web服务的中的调用是通过具有语义的本体来明确表达Web服务的数据、功能等，这样生成的Web服务使得机器就能够自动理解并进行相应的操作；在服务发布的时候，对服务的描述由原先基于关键字的提高为基于语义的描述，使得服务的发布和查找具有语义功能，也提高了服务进行组合的能力，按照这种方法进行Web服务生成过程。面向语义的Web服务组合中的最为主要结果是W3C推荐的OWL-S（Ontology Web Language for Services，Web服务的本体语言），OWL-S

是一个语义服务本体的描述，在这个描述的基础上，服务具有了强大的语义功能，可以使服务的查找发现、匹配、选择、组合、执行和监控等[30]。 如图2.9服务的顶层本体所示。

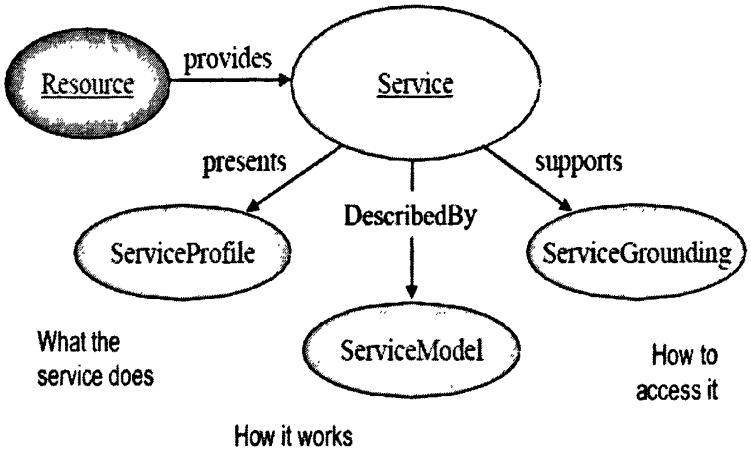

图 2.9 基于 OWL-S 的服务的顶层本体

OWL-S 是 W3C 推荐的新一代语义 Web 服务描述框架。OWL-S 框架结构如图所示。在 OWL-S 的顶层本体描述中，顶层的服务对应一个 Service，每个 Service 都由 ServiceProfile 本体、ServiceModel 本体和 ServiceGrounding 本体三个本体构成，分别担当不同的功能[31]，详细的描述如下：

1）ServiceProfile 本体的功能

ServiceProfile 本体是主要描述了服务的非功能属性、功能属性以及附加特征信息三个方面的特征。担当的作用是在服务发布时用于描述服务的性能，并支持服务查询和匹配及发现等功能。

a、非功能属性方面：

serviceName：指出了提供的服务的名称，通常是作为服务标识来使用。

textDescription：为服务提供了一个简短说明来描述服务的本身功能和运行所需的条件以及其它附加信息。

contactInformation：提供一种机制，用于说明对服务负责的单位或个人的联系信息。

b、功能属性方面：

Input：是服务的输入属性，指定了服务执行时作所需要的输入信息。

Output：是服务的输入属性，指定了服务执行后的产生的结果，相对应的是服务的功能的实现。

Precondition：前置条件说明了在服务被请求执行的前所应满足的逻辑约束

条件。这些条件可以影响到服务的不同结果，对于语义服务，在不同的前置条件下，会产生不同的结果，使得服务具有了更大的适用性和灵活性。

Effect：后置结果是在服务被成功执行后导致可能触发的的事件，即服务执行后对其他事件的影响，使得服务显得具有智能性。

c、服务附加特征方面：

ServiceCategory：服务的目录索引，主要包含服务分类的名称、按照不同的方式进行归类服务，在分类法中引入的叙词、该服务在分类法中对应概念空间、以及对应的关键字术语。

QualityRating：指定了服务的不同的级别，对应服务的不同质量指标。它包含在分类系统中服务的统一名称和服务的所处的级别与服务的优先级。

ServiceParameter：是一个可扩展的属性。可以包含任何信息，为以后加入新的功能而留有扩展空间。

2）ServiceModel 本体的功能

ServiceModel 本体的功能主要是描述了服务如何组织工作的。ServiceModel 和其子类 Process 使潜在的客户端能够理解一个服务如何运作，并且能够使其了解如何使用该服务[32]。ServiceModel 是设计用来处理简单服务和组合服务；具体如图 2.10 的 OWL-S 的语义服务组合原理图，其中包括如下的三个过程：

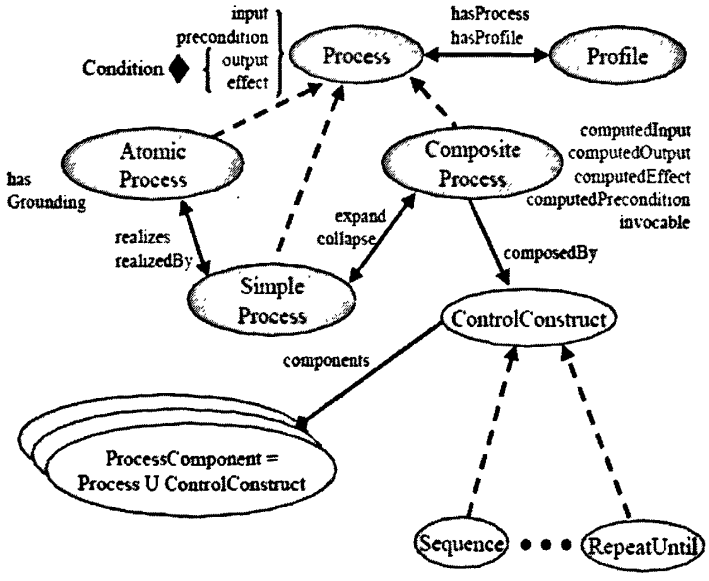

图 2.10 基于 OWL-S 的 Web 服务组合的原理图

原子过程：是一个不可再分的原子调用过程，对应的是原子服务，也有其对应的服务描述。

简单过程：是概念上的一个简化，是组合服务的抽象化，没有实用价值。

组合过程：构成是由原子服务或其他的组合服务，在组合构成中，包括了服务运行的流程控制和数据控制。

在这些基础之上，对服务的并行化和实时性进行控制。

3）ServiceGrounding 本体的功能

ServiceGrounding 本体用于描述如何访问服务的细节，详细描述了从抽象的 ServiceProfile 和 ServiceModel 类到具体实现的映射[33]。

2.3.4 OWL-S 与 WSDL 的关系

W3C推荐的OWL-S框架中的ServiceGrounding本体的主要作用是将OWL-S中的Process过程模型的描述和WSDL规范中的通信协议及消息描述关联起来[34]。ServiceGrounding本体的内容详细的描述了如何访问该服务和这个服务在什么位置，指明了服务执行者去该位置调用该服务，即服务的实际调用执行。服务的语义注册、查询和组合是由OWL-S描述中的ServiceProfile和ServiceModel来完成的。WSDL和OWL-S实际上在使用上覆盖了不同的概念空间。

WSDL规范是受XML Schema来规范约束的，而OWL-S规范是由OWL类来约束的。两者之间需要对应概念进行映射。方法是采用OWL中的类来表达WSDL中抽象类型的声明，而使用WSDL的绑定结构来表达消息的格式[32]。相对应的在服务注册中心，我们可以通过OWL-S中ServiceProfile来提供Capability Search来支持进行语义查找，而不是传统的基于关键词的查找。

OWL-S 规范进而 WSDL 规范之间的映射关系如下图 2.11 所示[35]。

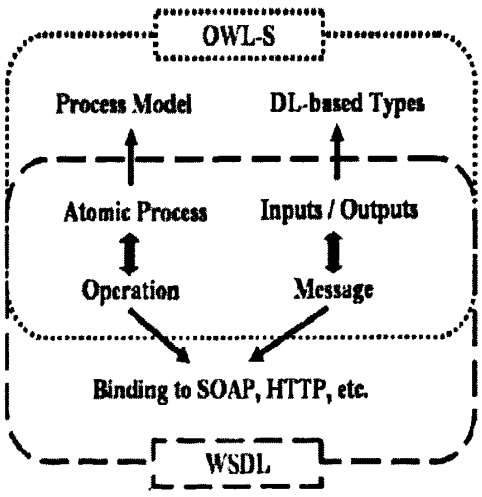

图 2.11　OWL-S 与 WSDL 之间的映射关系

具体的对应关系：WSDL 中的 operation 操作对应到 OWL-S 的 Atomic Process 原子过程；WSDL 中的 message 对应到 OWL-S 的 Atomic Process 的输入 Inputs 和输出 Outputs；WSDL 中的 abstract type(XML Schema 定义)对应到 OWL-S 中的 Inputs 和 Outputs 的类型（OWL Class 定义）；这样就建立了一一对应关系。

计算机在执行 WSDL 到 OWL-S 的映射方法的步骤[36]是：第一步，创建一个空的 OWL-S 本体实例给要进行转化的 WSDL 文件做好准备；第二步，解析 WSDL 文件从中得到数据；第三步，将解析读到的信息添加到 Process，Profile，Grounding 本体实例中的对应的关键项中。WSDL 到 OWL-S 的映射具体方法为：首先为该 WSDL 文档创建一个空的 OWL-S 本体实例；其次读取经过解析得 WSDL 数据，最后将这些信息添加到 Process，Profile，Grounding 本体实例中的对应项中。从具体的实现来看，创建一个空的 OWL-S 原子服务的本体实例主要是通过 OWL-S 规范提供的 API 函数来实现具体的功能。基本思想是首先利用工厂模式中的创建本体模型的方法来获得一个 OWL 本体模型；然后是对应的 service 名称，对应的 profile 调用策略，对应的 process 组合过程，对应的 grounding 绑定位置；最后将它们各个部分与本体实例的 service 建立起直接的连接。

2.3.5 OWL-S 与 UDDI 的关系

UDDI 是一套基于 Web 的、分布式、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 服务架构下不可缺少的组成部分，受到业界的强大支持。服务提供者注册服务的广告(advertisement)和提供所属类别的关键词。服务请求者基于关键词从注册机构检索所需的服务。UDDI 的检索机制仅依赖预先定义好的分类，通过关键词来匹配，不涉及广告的语义内容[37]。

由于 UDDI 不支持服务的语义描述，因此即使可以在 UDDI 注册中检索 WSDL 服务描述，也需要人工的参与来理解服务描述的语义。OWL-S 支持比 UDDI 更灵活的服务发现，ServiceProfile 对 UDDI 进行了补充和扩展，支持对 ServiceProfile 中概念及其属性的语义检索，而 UDDI 仅限于对 tModel_Keys 的关键词检索，如图 2.12 所示。

从服务描述方面，UDDI 和 OWL-S 是互相补充的。 ServiceProfile 是中立的注册模型(registry-model-neutral)，也就是说，ServiceProfile 支持各种各样的注册模型，最常用的注册模型是 UDDI 的基于服务注册中心的集中式解决方案。

将 OWL-S 和 UDDI 结合，既可以利用 UDDI 来扩大 Web 服务的发现范围，又利于基于语义描述的 Web 服务发布与检索，从而使基于服务功能的 Web 服务

组合发现成为可能[38]。当UDDI在发布或检索Web服务时，可以到对应的OWL-S文档所定义的本体中去检索元素所表达的语义。将 UDDI 广告植入 OWL-S 的 ServiceProfile 信息时，若 UDDI 中存在 serviceProfile 对应元素，采用一一映射，否则采用基于 tModel 的映射。

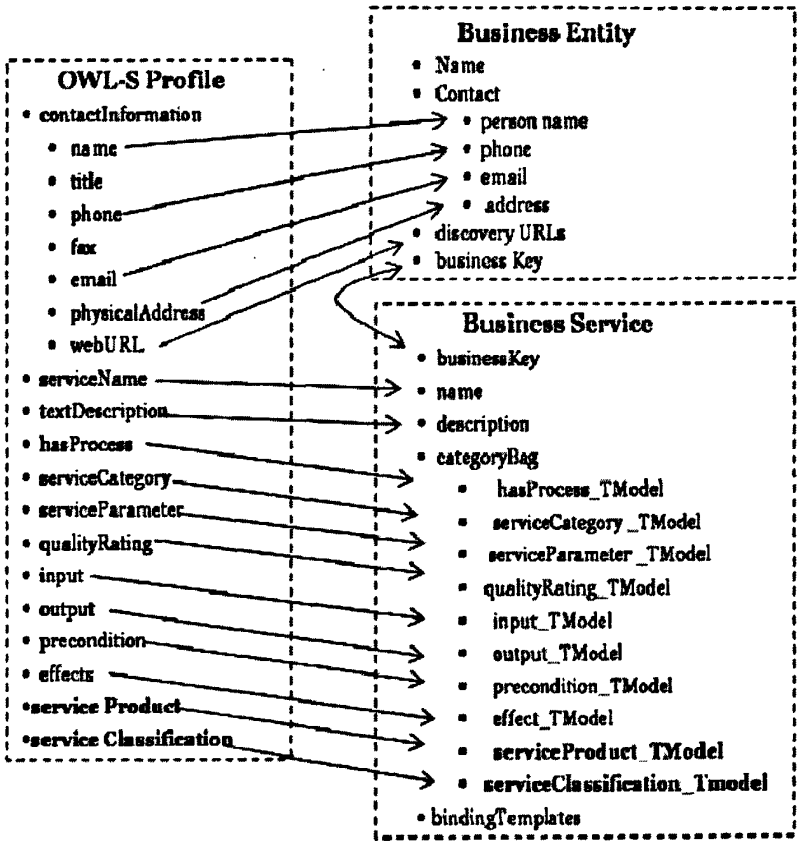

图 2.12　OWL-S 与 UDDI 之间的映射关系

将 OWL-S 嵌入到 UDDI 中以提供 UDDI 中所缺乏的语义描述能力。通过将 OWL-S Profile 映射到 UDDI 服务描述可以增加 UDDI 的语义功能。tMedel 方法使得对 UDDI 在进行语义的扩展非常的方便，并且与现行的标准的 UDDI 注册中心(无论是公共的还是私人的)的结合上没有任何的冲突，使得既有语义功能的 Web 服务直接注册保存在 UDDI 注册中心，通过找到特定的 tModel 方法就直接可以发现服务[39]。商业实体信息、商业服务信息、技术绑定信息和元技术信息(tModel 元素) 的语义信息都包含在 UDDI 的结构模型中，从图中可以看出 OWL-S 中的 ServiceProfile 中元素与商业实体信息元素和商业服务信息元素一一对应起来的。这种映射带来的好处是把 OWL-S 文档同 UDDI 进行了完全的结合，使得服务请求者对 UDDI 在语义性增强上的实施细节透明化。使得 UDDI 在语义方面得到了扩展，为我们进行基于语义的服务组合提供语义化得

基础。

2.4 本章小结

在介绍了首先介绍了 Web 服务的技术架构，然后，详细分析了语义网及其体系结构，包括本体的建模过程，本体的概念定义、建模原则、形式化表达到本体的实例；接着介绍了语义 Web 服务及其组合的框架 OWL-S 的顶层服务本体模型及模型提供给进行服务组合的基础内涵和外延；最后，介绍了 OWL-S 与 WSDL 和 UDDI 的关系，OWL-S 与 WSDL 之间的关系可以是一般的 Web 服务与同样可以转化成语义服务；OWL-S 与 UDDI 的关系可以是语义 Web 服务发布到 UDDI 上，并提供按照语义来查找服务，为下一章进行服务组合和模型建立奠定了基础。

3 语义 Web 服务组合算法

进行语义 Web 服务匹配、服务组合与执行的首要条件是实现语义 Web 服务发现。对于基于语义功能的服务发现的实现，首要的任务是解决语义的冲突问题，不同的领域本体之间对相同概念的抽象也有不同，存在着语义异构；然后是充分考虑对 Web 服务语义进行规范统一的标注，在这些基础上，为整个服务匹配和服务组合上提供了完整的基础设施。

3.1 语义 Web 服务发现算法

服务发现方法是指按照某种既定的匹配库中的匹配规则，由匹配规则在语义服务注册中心中查找满足匹配要求的服务的过程。然而传统的服务发现是单纯的基于语法功能上的输入关键字进行匹配查找，查找的范围空间只能给予关键字，查找的结果就远远不能满足服务的动态发现的所需的需求[40]。将语义概念引入到 Web 服务注册中心中，使得服务发现查找时按语义功能查找，语义是通过领域本体对服务进行标注，这样就能大大提高了服务的发现与查找精确程度。

3.1.1 语义 Web 服务的描述

在 W3C 推荐的语义 Web 服务描述语言 OWL-S 规范中，一个语义 Web 服务的顶层本体是由三部分组成，它们分别是服务轮廓 ServiceProfile 本体文件、服务过程 ServiceModel 本体和服务绑定 ServiceGrounding 本体文件等，其中服务轮廓本体文件描述服务了能力，是作为服务匹配的主要标准。在该文件中四个参数可用来进行语义匹配，它们分别为服务的输入参数(Input)、服务的输出结果(Output)、服务执行的前提条件(Precondition)和服务执行后的产生效果(Effeet)(简称为服务的 IOPE 四元素)，本文的所采用的匹配算法是基于语义的服务输入和服务输出参数的匹配。

作为描述语义 Web 服务的语义标注语言 OWL-S，主要的关键思想是考虑到了 Web 服务三个方面的语义功能，在实现方法是分别用类 ServiceProfile、ServiceModel 和 ServiceGrounding。对于服务的匹配过程，类 ServiceProfile 是服务请求者作为表达服务查询条件的语言，也是服务提供者作为广告发布在服务注册中心来进行注册的[41]。ServiceProfile 的信息被利用来进行服务匹配，匹配过程主要是在三个方面的信息进行匹配，它们是服务的基本信息、服务的功能信息和服务的其他特征信息；其中服务的基本信息是通过属性 serviceName、textDescription 和 contactInformation 来描述的，然而这三个属性所描述的信息是

能自动处理的、只提供人工阅读；服务的功能是体现在服务四个参数的 IOPE（Input/Output/Precondition/Effect），它们分别表示一个语义服务的输入参数、输出参数、执行服务的先决条件和执行服务后产生的结果，IOPE 可以定义多个还包括零个。在 ServiceProfile 定义的下列属性 hasInput、hasOutput、hasPrecondition 和 hasEffect；它们分别指向了 ServiceModel 中的 Process 类相应的 IOPE 描述，并利用 parameterType 来指定属性类型的分类，其分类可根据某本体论或领域的分类法；另外服务的其他特征参数是通过属性：serviceParameter、serviceCategory、qualityRating 来描述，其中 serviceCategory 用来指定服务的分类，是领域本体中的或服务分类法中的概念[42]。

3.1.2 语义 Web 服务的匹配算法

由于在 Profile 中的属性 serviceName、textDescription 和 contactInformation 所描述的基本信息不参与服务匹配算法进行匹配，然而 hasPrecondition 和 hasEffect 这两个属性的标准化描述程度达不到匹配的要求，很难在语义上的进行推理，所以在本章设计的匹配算法主要是针对 hasInput、hasOutput、serviceCategory 属性分别进行匹配，在目前流行的匹配方法中的匹配结果不进行分级匹配，结果要么匹配成功，要么直接匹配失败，这样对于要求有很多满足请求的匹配结果时，就很难确定所选的服务是否最优；或者是在没有服务进行匹配时，也在很大程度上减弱匹配条件来进行服务的匹配，所以在提出了对 hasInput、hasOutput 属性进行匹配的同时，再分别对它们属性类别进行分级的逐步匹配，因为属性所属的分级类别是在 OWL-S 中可描述的一些 OWL 本体论中的概念[43]。因此，在应用中实质是对属性之间的关系以及概念实体之间的关系进行匹配。

1）如果 M 与 N 是在给定的一组本体文件中的两个概念实体，可以通过下面的定义来说明它们之间的关系：

a、EQUIVALENT(M, N)：说明 M 与 N 在相同的概念。

b、SUBSUMES(M, N)：说明 M 包含 N。

c、SUBSUMES_INVERT(M, N)：说明 N 包含 M。

d、FAIL(M, N)：说明 M 与 N 彼此之间没有关系。

2）再者，若 R 与 S 为给定的一组本体论的两个属性，可以通过下面的定义来说明它们之间的关系：

a、EQUIVALENT(R, S)：说明 R 与 S 为相同的属性。

b、SUBPROPERTY(R, S)：说明 R 为 S 的子属性。

c、FAIL(R, S)：说明 R 与 S 彼此之间无关系。

d、UNCLASSFILED：匹配方法不支持这种情况。

根据以上的定义的这些关系，匹配过程就可以对请求与广告的服务描述中的 hasInput、hasOutput 属性以及所属的分类（某本体论或分类中的概念）进行匹配，它们的匹配结果如表 3.1 所示；而 serviceCategory 属性定义的是服务的分类，所以只对其分类上的概念进行匹配，结果按照匹配情况可分为 4 个等级，分别为 0：FAIL，1：UNCLASSIFIED，2：SUBSUMES，3：MATCH。

表 3.1 hasInput，hasOutput 分级匹配

等级	属性匹配等级	稳定性	速度
0	FAIL	FAIL	FAIL
1		SUBSUMES_INVERT	
2	UNCLASSIFIED	SUBSUMES	UNCLASSIFIED
3		EQUIVALENT	
4		SUBSUMES_INVERT	
5	SUBPROPERTY	SUBSUMES	SUBPROPERTY
6		EQUIVALENT	
7		SUBSUMES_INVERT	TYPE_INVERT
8	EQUIVALENT	SUBSUMES	TYPE_SUBSUMES
9		EQUIVALENT	MATCH

3.1.3 语义 Web 服务的匹配算法的实现

在 OWL-S 语言的基础上，用 Java 语言实现了匹配过程，主要构建了以下几个类：Reasoner, ProfileMatching, InputParameterMatching, OutputParamatching, MatchingAlgorithm[44]。

实现匹配过程的推理是使用 Reasoner 类，开发实现是使用了 OWLJessKB、Jena、Jess 等。其中 Jena 是由 HP 实验室开发的开放源代码，它能把 RDF 文件转化为 SPO(Subject-Predicate-Object)格式的三元组表示为 RDF 模型，查询模型使用 RDQL 语言。Jess 是一款以 Java 为核心技术的专家系统开发工具，是由事实库、规则库、推理机三大部分组成组合其核心部分，在推理方面使用高效地匹配算法和处理规则；在 Jena、Jess 的支持下，OWLJessKB 为用 OWL 语言编写的本体提供的一种描述逻辑推理器来加强它的逻辑推理功能[45]。具体的使用流程如下：装载用 OWL 语言表示的 Jess 规则库和事实公理；→用 Jena 程序装载 OWL 文件并创建 SPO 三元组模型，SPO 是语义信息存放在知识库中的形式；→使用 Jess Rete 进行基于规则集中的推理。因此定义的类 Reasoner 主要围绕着 OWLJessKB 对象，通过它访问知识库。在该类中存在好多方法，其中主要提供了方法如下所述[25]：

1）public void loadOwlFile(URL path):装载 OWL 文件，并把相关的信息加入到知识库。

2）public int propertyMatch(String propertyM, String propertyN)：通过对知识库的查询，来判断属性 M 与 N 之间的关系。

3）public int conceptMatch(String conceptM, String conceptN)：通过对知识库的查询，来判断概念 M 与 N 之间的关系。主要实现的代码如下：

```
public int concepMatch(String conceptM,String conceptN){
    if (conceptM.equals(conceptN)||sameClass(conceptM,conceptN)){
        return EQUIVALENT;
    }else if(subsumes(conceptM,conceptN)){
        return SUBSUMES;
    }else if(subsumes(conceptN,conceptM)){
        return SUBSUMES_INVERT;
    }else{
        return FAIL ;
    }
}
```

4）public int rankForParameters(Parameter param1,Parameter param2):根据表 1 决定匹配参数之间的匹配等级。主要实现的代码如下：

```
public int rankForParameters(Parameter param1,Parameter param2){
  int typeDegree=EQUIVALENT ;
  typeDegree=conceptMatch(param1.getRestrictedTo(),param2.getRestrictedTo());
  int      propertyDegree      =      propertyMatch(param2.getPropertyName(),
           param1.getPropertyName());
    if (typeDegree==FAIL||perpertyDegree=FAIL){
        return 0;
    }else if (propertyDegree==UNCLASSIFIED){
        if (typeDegree==SUBSUMES_INVERT) return 1 ;
        if (typeDegree==SUBSUMES) return 2 ;
        else return 3 ;
    }else if (propertyDegree==SUBPROPERTY){
        if (typeDegree==SUBSUMES_INVERT) return 4 ;
        if (typeDegree==SUBSUMES) return 5 ;
        else return 6 ;
```

```
    }else {
        if (typeDegree==SUBSUMES_INVERT)return 7;
        if (typeDegree==SUBSUMES) return 8;
        else return 9;
    }
}
```

类 ProfileMatching、InputParameterMatching、OutputParameterMatching 分别实现了对属性 serviceCategory、hasInput、hasOutput 进行分阶段匹配，最后得到各个属性的匹配程度。在类 MatchingAlgorithm 中根据 serviceCategory、hasInput、hasOutput 各自的匹配结果，然后与所期望的匹配程度进行比较，从而得到最终的匹配结果。

3.2 语义 Web 服务的组合算法

3.2.1 语义 Web 服务组合算法的描述

对 Web 服务组合的分类，可以从其组合方式上进行分类：静态组合和动态组合。在设计阶段就已经定义好组合方式的语义 Web 服务服务组合就是静态组合。Web 服务集合中的 Web 服务及其 Web 服务之间的依赖关系都是静态服务组合需要考虑的问题。依据满足特定的功能的 Web 服务而在服务调用执行和运行时来调用这些服务是属于动态服务组合范围的[46]。它的的特点是，按照既定的目标之后，临时性组合，在执行完之后，这种组合方式就不存在了[25]。

定义 1：一个语义 Web 服务的形式来描述如下：$SWS_i(IN_i, OUT_{ri}, P_i)$，其中 SWS_i 用来表示该服务的名称，IN_i 表示该语义服务所需要的输入集合，OUT_i 表示该服务的输出集合，P_i 表示该服务的属性集合。

定义 2：假设基于领域本体知识库的所有服务分别表示为 SWS_1 (IN_1，OUT_1，P_1)，$SWS_2(IN_2, OUT_2, P_2)$，……，$SWS_n(IN_n, OUT_n, P_n)$，所用输入集合为 IS={ IN_1，IN_2，……，IN_n }，所用的输出集合为 OS={ OUT_1，OUT_2，……，OUT_n}，服务属性集合 $PS= P_i(1 \leqslant i \leqslant n)$。

定义 3：一个语义 Web 服务请求可以用下面的形式来表达：$SWSR_r(IN_r, OUT_r, T_r)$，其中 $SWSR_r$ 是语义 Web 服务的请求标识；IN_r 是请求所需要的输入集合，OUT_r 是请求所需要的输出集合，T_r 的值是单个服务还是组合服务，T_r 是服务类型，取值范围为单个服务或组合服务，其初始值却为空值。

算法的基本思想：

第一步：在网络制造本体知识库支持下，把请求服务 $SWSR_r$ 中的输入参数映射到 IS 集合中，把请求服务 $SWSR_r$ 中的输出参数映射到 OS 集合中；从而转

化成请求 Web 服务全部可以识别的参数，而不会因为参数在语义上不同引起语义冲突，进而导致服务匹配的失败。如果映射结果成功，则转到第二步；否则返回服务“匹配失败“的信息；

第二步：搜索目标服务 $SWS_g(IN_g, OUT_g)$，如果$((IN_g \equiv IN_r) \cap (OUT_g \equiv OUT_r)) \cup ((IN_g \equiv IN_r) \cap (OUT_g \subseteq OUT_r))$，则匹配成功，调用目标服务 SWS_g，否则转到第三步；

第三步：要完成该服务请求的功能是组合已有的基本服务组合。

组合算法描述如下：

```
DDC(SWRr INr,SWSRr OUTr) //动态发现组合算法
{    //在共享语义本体库支持下，①INr 等价于 IS 中的输入参数；②OUTr 等价于 OS
     中的输出参数
   if(①&②)==false then return("匹配失败！");
   else { if ((INg≡INr)∩(OUTg≡OUTr))U((INg≡INr)∩(OUTg⊆OUTr)) then
       {   WSRr.Tr=Single; //表明所需的目标服务是单个服务
           SearchGoalServiceWSg(INg,OUTg);//搜索目标服务
           return SWSg(INg,OUTg);  //返回目标服务
        } else
         {   WSRr.Tr=Composition //表明所需的服务是组合服务
           while (Web Service 库中的 SWS.Used==0)
           {   ConstructDAG;   //创建 DAG 图
               计算每一组到当前服务的整个代价，包括∀SWSi 和 SWSj 之间的权
               重代价 WEIGHTij,以及单个服务的计算代价
           }
           SelectMinCost 组合；//选择最小代价的组合
         }
   }
}
```

创建 DAG 图的基本思想：

初始化过程：初始化∀SWS，在本论文中用到的属性有：Tail 表示后继；Used 表示是否被使用过，0 表示没有被使用，1 表示已经被使用过。

组合匹配过程：以 $SWSR_r(INT_r, OUT_r, T_r)(SWS_i(IN_i, OUT_i, P_i))$作为 DAG 的第一个顶点，其中 SWS_i 的输入参数满足 $IN_i \subseteq IN_r$，若存在$(SWS_j(IN_j, OUT_j, P_j)$，使得 $IN_j \equiv OUT_r$ 或 $IN_j \subseteq OUT_r$（$IN_j \equiv OUT_i$ 或 $IN_j \subseteq OUT_i$），则 $SWSR_r.Tail = S\ WS_j(SWS_i.Tail = SWS_j)$，$SWS_j.Used=1$；

3.2.2 语义 Web 服务组合算法的分析

对该算法分析如下：

在共享领域本体库支持下，服务的请求者通过请求服务 SWSRr(INr)将服务请求映射到领域本体库的 IS 的子集中，通过请求服务 SWSRr(OUTr)将服务请求的映射到领域本体库的 OS 的中子集。若有硬件能支持这样映射的话，可以大大提高整个过程的匹配效率。如果请求服务在 IS 和 OS 中的映射失败，即可以提前发现匹配失败，从而提前结束服务；这样提高了服务匹配的效率，而无需等到服务发现时才觉得服务匹配失败[47]。

从服务请求这来说，它并不知道搜寻的目标服务到底是单个服务还是组合服务。通用的解决之道是先搜寻单个服务，单个服务搜索时如果匹配失败，则在进行组合服务，这样既提高了服务匹配的效率，也使得一些失败的情况早些出现。

在构建服务组合的过程中同时计算所选合适的服务代价，当服务每一个组合构建完成后，可以得出每一个服务组合的服务代价，可以用来衡量服务组合的优劣性，这样避免先构造服务组合图，然后在反向回溯的计算每一组服务组合的服务代价。

算法在选择最优或者最佳服务组合时，不仅考虑到了服务组合出来的最短路径的代价，而且同时还考虑了单个服务的计算代价。在不同的服务组合方案中选择最优的作为最后的服务组合方案。

3.3 本章小结

本章在 Web 服务的基础上，对 Web 服务的发现匹配算法由一级匹配改进成二级匹配，并对服务组合算法进行了详细的描述和形式化，为后面进行服务组合模型的建立奠定了理论基础。

4 语义 Web 服务组合的模型

通过前述对语义 Web 服务技术的深入分析和研究，本章将建立一个全面、通用的语义 Web 服务组合模型，该模型对语义 Web 服务组合过程中涉及到的运行和执行流程、匹配算法、组合算法、监控执行等都有比较详细的设计，是进行语义 Web 服务组合研究的基础。

4.1 语义 Web 服务组合模型

在基于 SOA 的 Web 服务体系结构的基础上，结合了语义网及本体的知识和 OWL-S 框架的体系结构，提出了一种通用的语义 Web 服务组合模型。该模型如下图 4.1 所示，

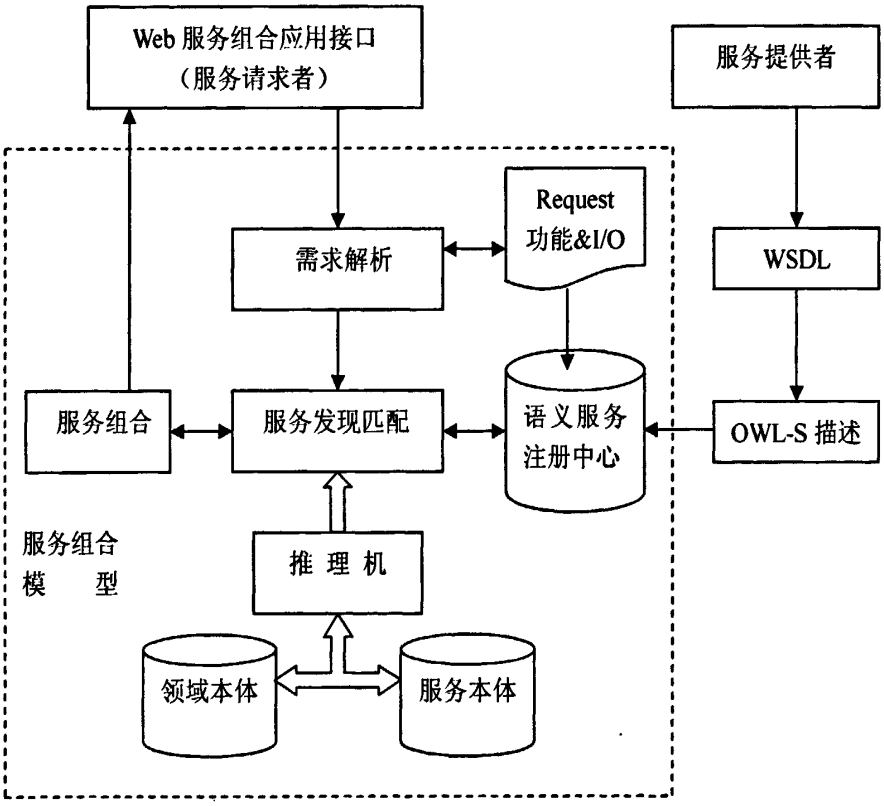

图 4.1　一种语义服务组合模型

该服务组合模型主要有服务发现匹配模块、服务组合模块、推理机模块、需求分析模块、语义服务注册中心、领域本体、功能本体等构成，在整个过程中加入了语义的功能，推理机需要有领域本体和功能本体的支持。

4.1.1 语义服务注册中心

语义服务注册中心用来存储企业的注册信息和服务本体文件。通过语义服务注册中心的接口，服务提供者的功能是可以将企业信息、个人信息和服务相关信息提交至服务注册中心，并且将描述该服务的OWL本体文件(或指定OWL文件的URL地址）上传给语义UDDI注册中心[48]。这样，一个服务的相关完整的信息才能够被全体服务请求者所看见，并且被作为服务请求者进行服务查询和匹配的依据。服务进行匹配时，服务匹配模块会遂一将服务注册中心的按预先存好的服务描述文件载入到 OWL 解析器中，分别对其进行解析，根据解析得结果再进行IO的匹配。一个原子服务也可是一个组合服务都可以在服务注册中心进行注册。

4.1.2 领域本体

对特定的应用领域环境中的一个层次结构的概念系统构成一个领域本体。它并且包含了领域内共同认可的共享知识库。将相关的领域知识和概念之间相互关系的明确分类定义是领域本体建立的目标。领域本体解决了不同的Web服务实体之间的语义理解和相互进行，从而避免了由于概念理解的歧义所造成的错误，为服务发现和组合提供了更大的方便，也为推理规则的顺利执行提供了便利，并且避免了关键字的查找技术所带来的不足。Web服务的本体通过导入领域本体的相关概念，使得服务的请求者和服务的提供者及服务发现匹配和服务组合引擎对语义的理解达成了共识，在机器层面上可以是极其智能化得自动处理。在本文中的领域本体以网络制造环境中的网络制造资源为参考标准，采用W3C标准的Web本体语言来描述领域本体，其中包含服务的概念以及服务间的关系等，为服务组合而提供了方便。

4.1.3 服务本体

服务本体中存放的是关于服务的语义描述。对原子服务按照语义的功能进行描述，为服务查找和组合提供了便利，服务本体直接可有Web服务的描述文件WSDL转化为OWL-S的本体文件[49]。

4.1.4 需求分析模块

需求分析模块负责与用户交互，可以将用户的输入信息分析后，传入给服务发现匹配模块，为服务匹配模块进行服务选择提供了依据。

4.1.5 服务发现匹配模块

实现服务匹配、服务组合与执行的首要前提条件是服务发现。领域本体是消解因缺乏共同语义解释而造成的语义冲突问题，也是要实现基于功能的服务

发现首要解决的问题。再者，对 Web 服务进行语义标注，这些信息使用户及时可以获得服务功能信息，因此用本体论建立领域本体库显得尤为重要。语义 Web 服务进行语义标注后，不仅对用户来说具有语义性，而且对于机器的查找也具有可“理解性”。

服务发现的实质是用户服务请求描述与语义注册中心的服务广告所描述之间的精确匹配程度，而传统的采用关键字匹配的模式缺乏语义，匹配的效率也非常低，匹配的结果只有成功或者失败，如从语义服务注册中心根据语义去匹配服务，即是查询的服务在注册中心不存在，匹配引擎也会把相关的服务发现查找出来，这样提供了服务匹配的准确率，也大大充分地提高了服务的利用率。

4.1.6 服务组合模块

服务组合模块的流程图如图 4.2 所示，当服务请求者发出服务请求（服务请求包含了服务功能参数和可以提供的输入参数以及期望得到的输出参数），系统将服务请求有需求分析模块转换成 Service Profile 文件，并将它直接交给匹配引擎来处理；匹配引擎是根据 OWL 推理功能对服务描述，进行基于 OWL-S 本体库和服务描述库的进行匹配，得到一组或几组满足条件的服务或服务集合[50]。这种匹配不仅仅是简单的关键字匹配，而且基于语义“理解”的匹配；使用匹配引擎可以逐个检查找期望的服务输入是否是请求者所提供的输入的子集或真子集，并且其输出是否请求者所期望的输出的子集或真子集；这样在所有匹配的服务中匹配率最高的服务就是发现的目标。

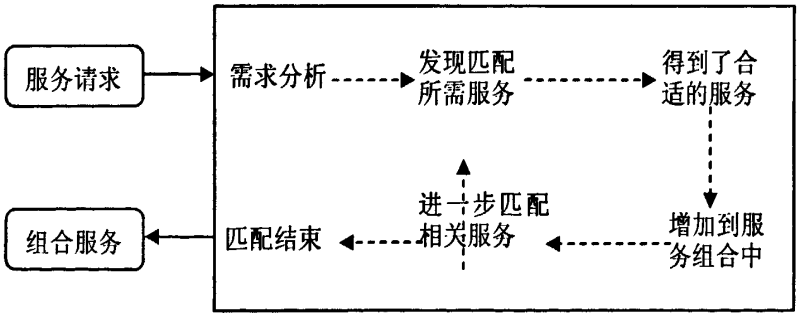

图 4.2 服务组合模块的组合流程

4.1.7 推理机模块

由功能本体来消解不同语义服务的服务输入和输出参数的语义冲突问题后，推理机根据推理规则与领域本体的相关概念信息相结合，可精确查找的相关的概念及关系。另外功能本体还确定服务请求和服务广告功能上的语义相似

性；领域本体确定输入和输出的相似性。根据预先设定的算法进行匹配，确定服务广告 POI 和请求 POI 的相似程度。算法是基于相似性，执行后返回和请求服务较为相关的服务。

4.2 领域本体的创建

领域本体是以网络制造为背景来创建网络制造设备资源本体，网络制造领域内的概念、关系、公理等非常庞大，按照一定的科学方法和步骤，并进行科学的组织和构建。运用本体建模工程的方法理念和步骤构建网络制造本体，用工程化的领域本体构造方法流程来构造本体。具体的步骤如下：确定本体的领域与范围；→领域信息的收集和分析；→重点概念和关系的确定；→建立本体框架；→进行形式化编码；→确认与评价。

企业信息和设备资源与服务方式的关系的关系式用属于或拥有的关系来表示，一个企业拥有一种设备资源，一个设备能属于一个企业，一种服务方式对应一种设备资源。企业信息和设备资源之间的关系是完全不相交继承关系；设备资源和机床之间存在继承关系；制造资源、服务方式和信息服务时间存在类不相交关系，且它们同属于网络制造资源本体的类，是聚合关系。本体编辑工具使用 protégé, 它是基于 Java 语言，支持 XML/RDF,RDFS,OWL 等本体语言，提供了对本体的读入和编辑后的输出，其中对 OWL 文件的操作是借助了惠普实验室开发的 Jena。领域本体的编辑工具创建了一个本体如图 4.3 所示[51]。

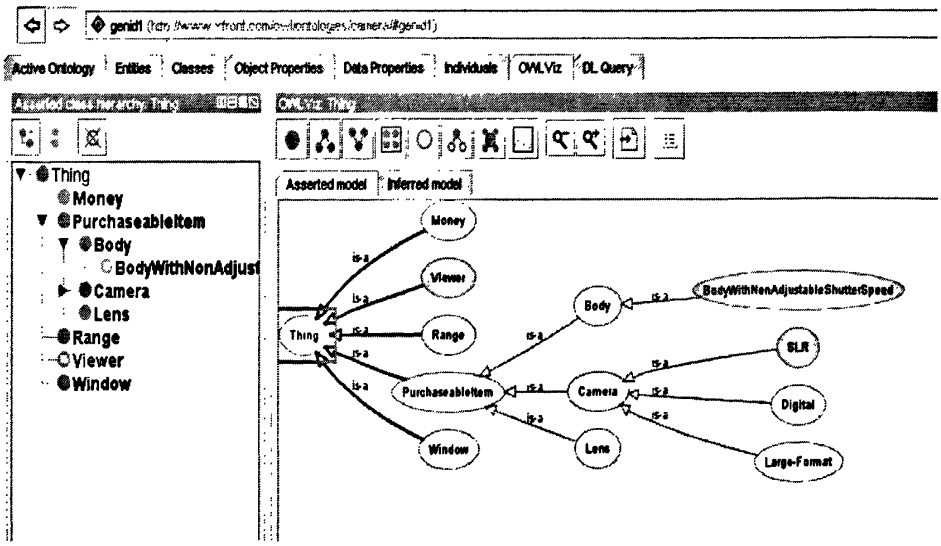

图 4.3 领域本体的创建实例

4.3 服务本体的创建

服务本体的创建采用 OWL-S Editor 来创建语义 Web 服务本体。OWL-S Editor 是语义 Web 中心组织开发本体是用到的编辑插件，专门用来编辑基于 OWL-S 描述的语义 Web 服务本体。OWL-S Editor 有两种方式可以生成 OWL-S 服务描述本体，一种是可以直接导入 WSDL 文件来生成，另一种是手工生成本体文件。直接生成的如图 4.4 所示。

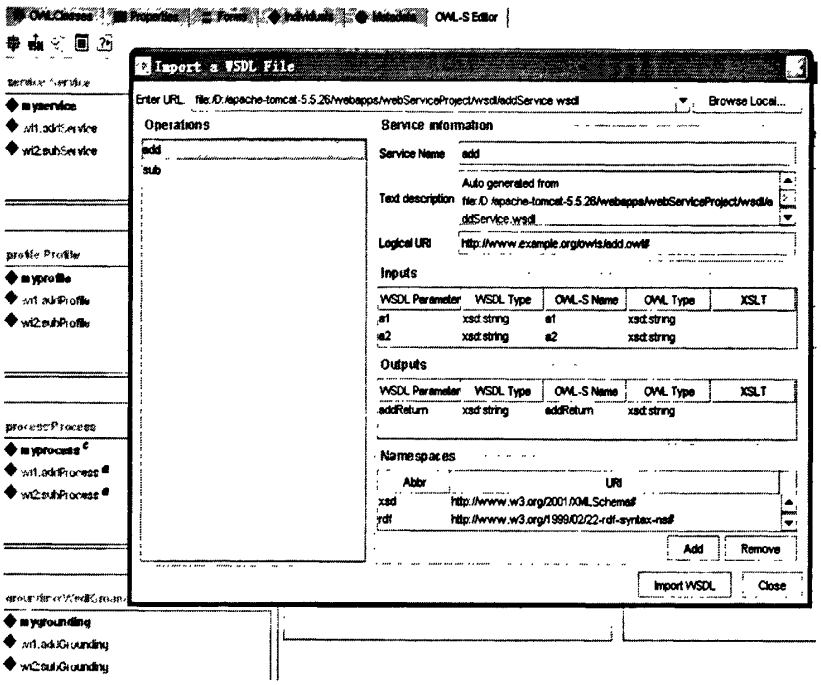

图 4.4 服务本体的创建实例

通过 OWL-S Editor 编辑器的 WSDL 文件生成服务本体文件，包括三个分别为：ServiceProfile、ServiceProcess、serviceGrounding 文件，文件的 profile 片段代码如下：

```
<?xml version="1.0" encoding="GBK"?>
<rdf:RDF>
    .....
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
    xml:base="http://www.example.org/owls/add.owl#">
    <profile:Profile rdf:about="#addProfile">
        <process:Input rdf:ID="a2">
   <process:parameterTyperdf:datatype="http://www.w3c.org/2001/XMLSchema#
```

```
anyURI">
        http://www.w3.org/2001/XMLSchema#double
</process:parameterTyperdf>
<rdfs:label>a2</rdfs:label>
 </process:Input>
 <process:Output rdf:ID="addReturn">
 <process:parameterType:datatype="http://www.w3.org/2001/XMLSchema#any
URI">http://www.w3.org/2001/XMLSchema#double
 </process:parameterType:datatype>
 </process:Output>
 <profile:serviceName>add<profile:serviceName>
 <profile:Profile>
</rdf:RDF>
```

4.4 本章小结

在语义 Web 服务的基础上，首先提出了语义 Web 组合模型，分析了组合模型中各个模块的作用和功能，其次，说明了领域本体和服务本体的创建的各自的作用，领域本体的作用是捕获相关的领域知识和概念之间相互关系的明确定义，为语义推理机的功能实现提供了语义基础，使得服务发现和组合更加方便。最后，介绍了服务本体的功能和作用，为单个服务进行组合而做好了准备。

5 语义 Web 服务组合的设计与实现

模型建立的正确性和可行性，只有概念和理论支持是远远不够的。在本章中，搭建了一系列的实验环境对第 3 章的语义 Web 服务组合的算法和第 4 章的语义 Web 服务组合模型进行实验，对其可行性及其合理性进行一定程度的验证。

5.1 背景简介

以网络制造环境为例，针对网络制造资源进行抽取，在软件层次上将服务组合分解为以下各层，如图 5.1 所示。

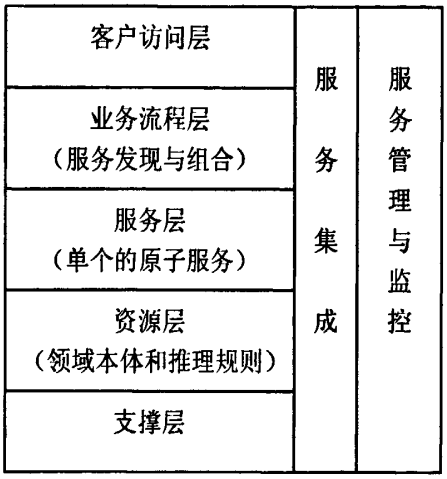

图 5.1 网络制造环境中的网络资源的抽取建模层次

在整体架构的支持下，每一层的功能明确，下面来具体描述各层的功能。

支撑层：包括基础的设施，即软硬件的支撑环境，网络设备环境和通信协议等。

资源层：包括领域本体和推理规则，这也是语义网的关键技术的实现点。由第二章的理论知识知道，概念层和元数据构成了制造资源本体的语义元数据模型，把网络制造资源进行概念、关系、实例、进行本体化建模，生成本体的实例，在本体实例的基础上，定义规则库，就可以按照网络制造领域的常识加入规则，可以获取本体中的知识。

服务层：服务层是用来提供原子服务，为上层的业务流程层提供服务发现与组合的基础。服务层是按照一定的功能把网络制造本体的知识封装成网络制造知识服务，把这些服务按照一定语义方法，发布到 UDDI 上，提供给按语义发现服务来查找。

业务流程层：即服务的发现和组合层，是我们研究的重点。按照语义发现服务，把业务流程定义为规则，按照一定的规则进行服务匹配，提高服务的发现和匹配，按照业务流程的规则也可以进行服务组合。

用户访问层：最终用户看到的界面，利用表示层来想用户提供用过访问接口。

服务集成：涉及到每一层，主要是管理所有的服务，服务的调用者通过服务集成来访问服务，从而保证了服务的位置独立性。

服务管理和监控：主要是监控服务的性能和可用性，管理服务的质量和安全，为整个系统提供一些辅助的功能。

5.2 工具简介

实验环境：操作系统为 Windows XP，本体编辑及服务组装工具 Protégé+OWL-S Editor，本体推理工具 Jena、Jess，语义服务注册中心（语义化的 UDDI）用 OWL-S/UDDI Matchmaker+Juddi，服务器使用 Tomcat5.5，编辑环境为 MyEclipse，数据库使用 MySqL5.0，还有 Axis，UDDI4J 等，下面简单介绍一下[52]。

Protege3.2+OWL-S Editor+Graphviz2.23：Protégé 为本体开发工具，用于建立本体及 Web 服务本体；OWL-S Editor 做为 Protégé 的一个插件，除具有 WSDL2OWL-S 功能外还可以组合服务，它是一个工具而不具有 API；Graphviz2.23 是开源的图表（计算机科学中数据结构中的图）可视化软件，在此作为 Protégé 的图形化插件；

OWL-S/UDDI Matchmaker+Juddi：语义服务注册中心，经传统 UDDI 的 tModel 与 OWL-S Profile 建立了映射关系。OWL-S/UDDI Matchmaker Client API：扩展了 UDDI4J 的对 UDDI 的查询、注册、调用服务功能，加入了语义的方法，结合 Jena（OWL-S API）在 OWL-S/UDDI Matchmaker 中进行服务的语义化操作；

jUDDI：是 Web Services UDDI 规范的一个 Java 实现，可以部署在支持 Servlet2.3 的任何 Java 应用服务器上包括：Tomcat、WebSphere 等，可以支持关系型数据库 MySql、DB2、Oracle 等；

UDDI4J：是一个 Java 类库，提供了一个 API 来与 Web 服务 UDDI 注册中心向结合。UDDI4J 的类库包含了 UDDI 客户机端应用程序的发布、发现和绑定 Web 服务的实现。UDDI API 可以分为查询 API 和发布 API；

Axis：Apache Axis 是 Apache Web Services 项目中的子项目，属于最早的一批用于构造基于 SOAP 应用的 Framework，它支持 WSDL1.1，可以自动生成

SOAP 服务，自动由 Java Object 生成 WSDL 的服务描述；

本体中推理机的选择：Jena 是针对本体进行推理查找服务，效率高，是语义 Web 和本体领域比较流行的开发工具；Jess 是对 Web 服务规则库进行推理，实现 Web 服务的自动组合，它是开放的，提供不同的规则系统给它，用户就可以进行不同领域的推理，它仅仅是推理引擎，效率难以优化。所以本文中研究中本体使用 owl 文件表示，使用 Jena 来推理，使用 Jess 来进行产生式规则推理；

OWL-S API：是用来实现对语义 Web 服务的描述，读写，以及调用原子的或者组合的 Web 服务。OWL-S API 提供了 Java 下读取，执行和生成 OWL-S 文档的 API。在内部实现上，OWL-S API 使用和 OWL-S 本体对应的数据结构，Java 包、Interface 和方法名都和 OWL-S 本体的概念相对应。对于一个服务，若只有一个操作，就生成一个 OWL-S 文件，文件名即为这个操作的名称；若有多个操作，就对应多个 OWL-S 文件。OWL-S API 还提供了一个执行引擎，既可以调用原子过程，有可以调用服务过程。原子过程自身带有 WSDL 接口或者通用即插即用（UPnP）特点，组合过程可以使用控制结构，如顺序(Sequence)、无序(split)和选择(choice)。另外，OWL-S API 中的所用对象都继承 OWL Resource 类，因而具有 accessor 函数，来获取 OWL 属性的值，也可获取各层数据模型（基于 Jena）用于更高级的检索[53]。

OWL-S VM(Virtual Machine, 虚拟机器) 提供一个完整的基于 OWL-S 的 Web 服务的调用环境，它包含以下三个主要部分：如图 5.2 所示。

Web Service Invocation（Web 服务调用）：包含两个主要部分，分别是 Axis's Service Invocation、OWL-S Web Service Invoker;主要是将服务转换成 OWL-S 格式并传给 OWL 推理引擎。

OWL-S Processor（处理器）：它包含了 Grouding Execution Rules、Process Model Execution Rules；主要是由由 Web 服务调用与服务提供者沟通，来决定服务如何被执行。

OWL-S Inference Engine（推理引擎）：包含 OWL JessKB、Jess、Jena 三个推理器，主要是处理从其它的服务取得的 OWL-S 相关规定，并将信息传给 OWL-S 处理器。

5.3 环境的搭建

在上述工具的基础上，搭建实验环境，按照如下步骤：

安装 Java 语言环境支持：JDK

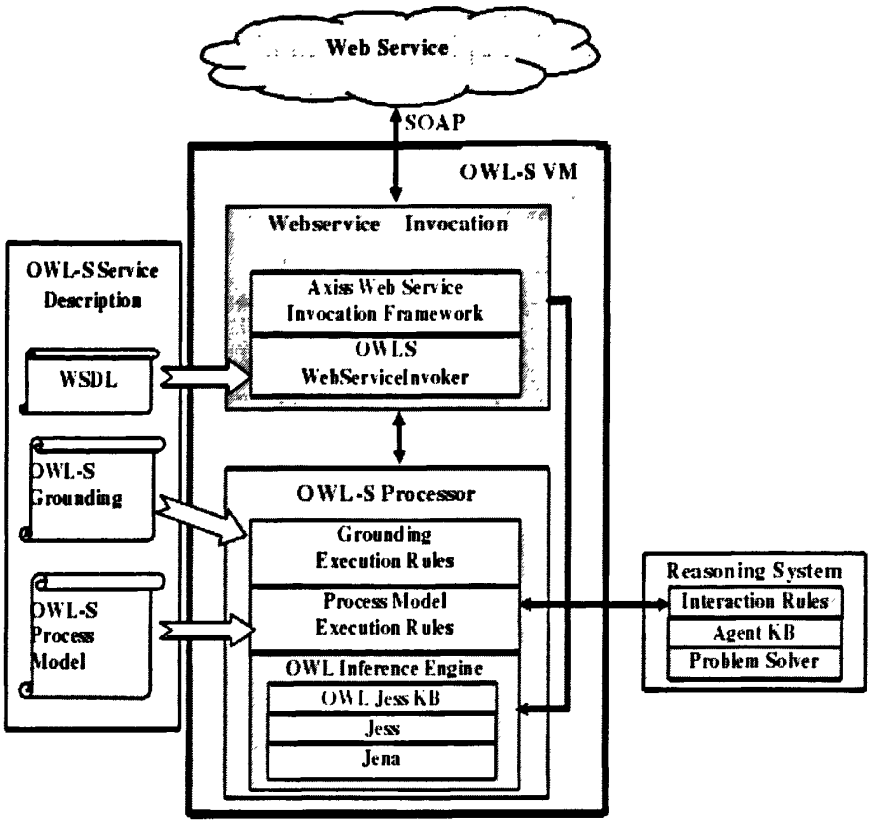

图 5.2 OWL-S 虚拟机工作原理图

安装运行的服务器：Tomcat

安装 UDDI 的数据库：MySql

在服务器 Tomcat 中加入 UDDI 的项目工程，具体是将 juddi-0.9rc4.zip 解压缩，将 webapps 目录下的 juddi 目录拷贝到 Tomcat 的 webapps 目录下，这样用户可以可以直接通过 Web 来注册和查询 UDDI 中心；

运行 jUDDI，解压目录中的 sql\mysql 目录下的 create_database.sql 和 insert_publisher.sql；执行 SQL 语句如下：

INSERT INTO PUBLISHER (PUBLISHER_ID, PUBLISHER_NAME, EMAIL_ADDRESS, IS_ENABLED,IS_ADMIN) VALUES ('juddi','juddi', 'guoyongshe@163.com','true','true');

这样就插入到表 publisher 中一条数据，就是一个用户，用户名：juddi，密码：juddi，在 uddi4j 中将用这个用户来创建 Web 服务。

配置 Tomcat5.5 的 MySQL 数据源：将 mysql –connector –java -3.1.11 -bin.jar 复制到 Tomcat 的 common/lib 目录中，并加入 classpath；且修改 Tomcat 安装目录子目录配置$Tomcat_home$/conf/server.xml，在<host></host>中添加：

<Context crossContext="true" path="/juddi" reloadable="true">

```
<Resource auth="Container"
        name="jdbc/juddiDB"
        type="javax.sql.DataSource"
        password="123456"
        driverClassName="com.mysql.jdbc.Driver"
        maxIdle="2"
        maxWait="5000"
        validationQuery="select count(*) from publisher"
        name="gys"
        url="jdbc:mysql://localhost:3306/juddi"
        maxActive="4"/>
</context>
```

启动 Tomcat，在 IE 的地址栏中输入 http://localhost:8080/juddi，则可以看到了 index.jsp 页面，点击“validate”选项，可以看到配置信息是否正确，如果正确，则 jUDDI 注册中心安装成功。如图 5.3 所示。

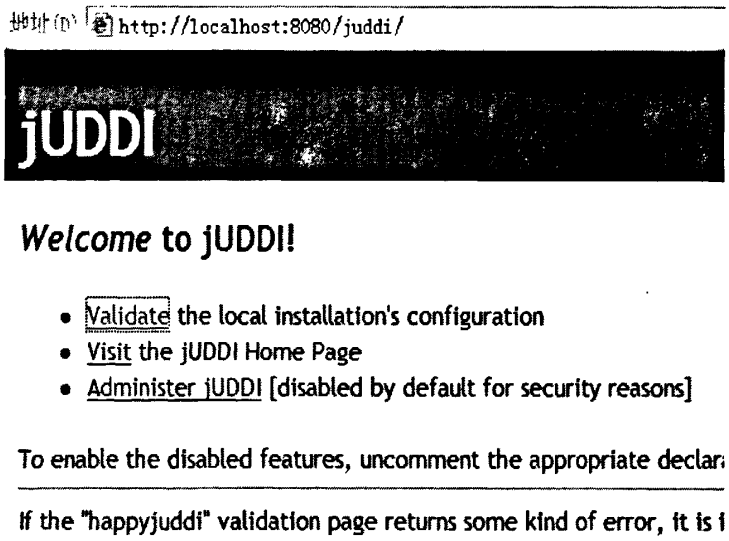

图 5.3 jUDDI 安装和检测

安装 OWL-S/UDDI Matchmaker：在 cmd 打开 DOS 命令行下，到 OWL-S/UDDI Matchmaker 的安装目录下运行 ant install-juddi-matchmaker 进行安装；然后进行测试，执行 ant run-tests，正确的话会提示 success。

将 Axis 相关的包文件放在 WEB-INF\lib 目录下。Axis 可选的包：activation.jar; mail.jar; xmlsec-1.4.beta1.jar 拷贝到 WEB-INF 目录下，生成客户端时需要使用。

将 Jena2.4\lib 下的 jar 和 jess.jar 文件全部加入 CLASSPATH。

至此，实验环境搭建完成。

5.4 领域本体的创建

对网络化制造资源的本体建模，建模的目的是使整个系统具有语义的基础。将网络化制造本体中所涉及的内容概括为三种抽象类型进行表达，即概念、概念的属性以及属性间的关系，在此基础上，根据制造领域特点，通过进一步细分和增加语义信息，并叠加公理定义，形成具有复杂语义关系，支持推理的网状结构。本体的创建使用 Protégé 工具。

领域本体的创建依据第 2 章的本体的建模实例，来进行创建。具体如 5.4 图。

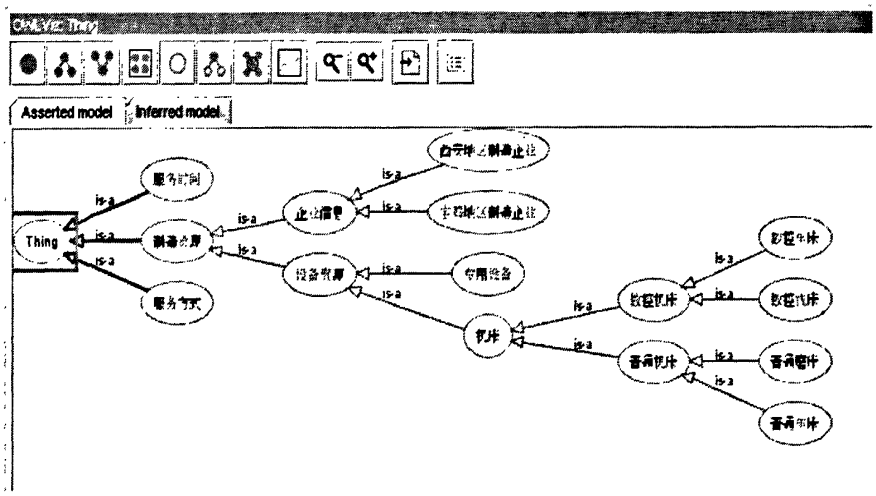

图 5.4 网络制造设备资源领域本体的创建

5.5 对本体的推理

在网络制造环境中，企业分为大企业和小企业，产品分为计算机产品和家电产品，按照不同的分类建立一个本体如图：我们对本体进行推理；原本体的关键代码如下：

```
<?xml version="1.0"?>
  <!DOCTYPE Ontology [
    <!ENTITY example "http://example.org#" >
    .....
    ]>
<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
```

```
 xml:base="http://www.w3.org/2006/12/owl2-xml#"
 ......
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 URI="file:/E:/Reason/ReasoningExample/test/MR.ttl">
<SubClassOf>
    <Class URI="&MR;FamilyProduct"/>
    <Class URI="&example;Product"/>
</SubClassOf>
<Declaration>
    <Class URI="&MR;FamilyProduct"/>
</Declaration>
<SubClassOf>
    <Class URI="&example;ComputerProduct"/>
    <Class URI="&example;Product"/>
</SubClassOf>
<SubClassOf>
    <Class URI="&example;LargeCompany"/>
    <Class URI="&example;Company"/>
</SubClassOf>
<EquivalentClasses>
    <Class URI="&example;MultiProduct"/>
    <ObjectIntersectionOf>
        <Class URI="&example;Product"/>
        <ObjectHasValue>
            <ObjectProperty URI="&example;Own"/>
            <Individual URI="&example;PC"/>
        </ObjectHasValue>
    </ObjectIntersectionOf>
</EquivalentClasses>
<SubClassOf>
    <Class URI="&example;SmallCompany"/>
    <Class URI="&example;Company"/>
</SubClassOf>
.....
```

```
<DataPropertyRange>
    <DataProperty URI="&example;name"/>
    <Datatype URI="&xsd;string"/>
</DataPropertyRange>
<SubDataPropertyOf>
    <DataProperty URI="&example;registeredName"/>
    <DataProperty URI="&example;name"/>
</SubDataPropertyOf>
……
<Declaration>
    <Individual URI="&MR;LocalCompany"/>
</Declaration>
<ClassAssertion>
    <Class URI="&example;ComputerProduct"/>
    <Individual URI="&MR;Server"/>
</ClassAssertion>
<Declaration>
    <Individual URI="&MR;Server"/>
</Declaration>
……
</Ontology>
```

相对应的本体编辑器 protégé 的本体实例图 5.5 如下所示，

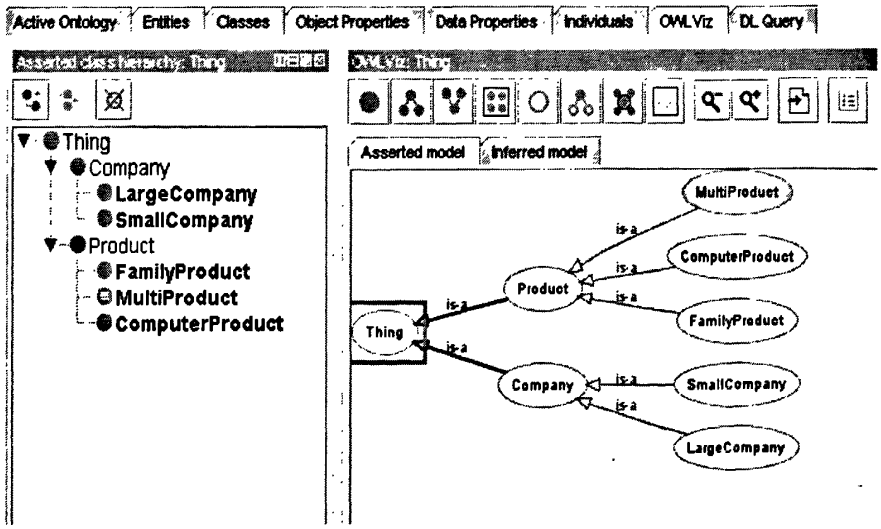

图 5.5　进行推理验证的本体结构

运行模式分为两种情况：在无推理模式的情况下和带有推理功能的情况下，在没有推理的情况下，产生的输出文件的得到的是真正本体中表达的个体和陈述集合。输出结果没有产生额外的陈述，因为程序并没有推理出额外的陈述，结果输出如下

1.Individual: LocalCompany
2. name : LoalCompany is a SmallCompany
3. type : http://www.w3.org/2002/07/owl#Thing
4. type : http://example.org#SmallCompany
5.
6.Individual: PC
7. name : Server Machine
8. type : http://www.w3.org/2002/07/owl#Thing
9. type : http://example.org#Product
10.
11.Individual: DELL_Company
12. name : DELL_Company is a Big Company.
13. type : http://www.w3.org/2002/07/owl#Thing
14. type : http://example.org#LargeCompany
15.
16.Individual: Server
17. type : http://www.w3.org/2002/07/owl#Thing
18. type : http://example.org#ComputerProduct

在有推理的模式下，我们看到了 RDFS 语义能够推理出一些结果。子类和子类的属性对会产生作用。以 RDFS 模式运行应用程序所产生如下面所示。

1.Individual: LocalCompany
2. name : LoalCompany is a SmallCompany
3. type : http://example.org#Company
4. type : http://example.org#SmallCompany
5. type : http://www.w3.org/2002/07/owl#Thing
6. type : http://example.org#Product
7. type : http://example.org#ComputerProduct
8. sameAs : file:///E:/Reason/ReasoningExample/test/MR.ttl#LocalCompany
9.

10. Individual: DELL_Company
11. name : DELL_Company is a Big Company.
12. type : http://example.org#Company
13. type : http://example.org#LargeCompany
14. type : http://www.w3.org/2002/07/owl#Thing
15. type : http://example.org#Product
16. type : http://example.org#ComputerProduct
17. sameAs : file:///E:/Reason/ReasoningExample/test/MR.ttl#DELL_Company
18.
19. Individual: Server
20. type : http://www.w3.org/2002/07/owl#Thing
21. type : http://example.org#Product
22. type : http://example.org#ComputerProduct
23. sameAs : file:///E:/Reason/ReasoningExample/test/MR.ttl#Server
24.
25. Individual: PC
26. name : Server Machine
27. type : http://www.w3.org/2002/07/owl#Thing
28. type : http://example.org#Product
29. type : http://example.org#ComputerProduct
30. sameAs : http://example.org#PC

5.6 基于语义服务的描述

针对网络制造环境中的一个流程进行服务组合，对本体进行封装形成三个服务，分别是订单服务，制造服务，送货服务来够构成。订单服务 OrderService 的输入是用户的信息和订购产品的信息构成订单，制造服务 ManufactureService 根据订单进行生产，完成订单的制造任务，送货服务 SendService 接受到产品制造任务完成后，就执行送货任务。整个的服务组合流程完成了从订单到送货的一系列流程如图。

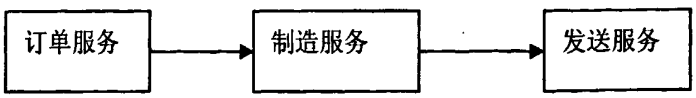

图 5.6 网路制造的业务流程图

订单服务 orderService 的描述文件 WSDL 如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://example"
        xmlns:apachesoap="http://xml.apache.org/xml-soap"
        xmlns:impl="http://example"
        xmlns:intf="http://example"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
   <schema elementFormDefault="qualified"
         targetNamespace="http://example"
         xmlns="http://www.w3.org/2001/XMLSchema">
     <element name="ordering">
       <complexType>
         <sequence>
           <element name="userInfo" type="xsd:string"/>
           <element name="productInfo" type="xsd:string"/>
         </sequence>
      </complexType>
     </element>
     <element name="orderingResponse">
      <complexType>
        <sequence>
          <element name="orderingReturn" type="xsd:string"/>
        </sequence>
      </complexType>
     </element>
  </schema>
  </wsdl:types>
    <wsdl:message name="orderingRequest">
      <wsdl:part element="impl:ordering" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="orderingResponse">
      <wsdl:part element="impl:orderingResponse" name="parameters"/>
```

```
    </wsdl:message>
    <wsdl:portType name="OrderService">
    <wsdl:operation name="ordering">
     <wsdl:input message="impl:orderingRequest" name="orderingRequest"/>
     <wsdl:output message="impl:orderingResponse" name="orderingResponse"/>
      </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="OrderServiceSoapBinding" type="impl:OrderService">
       <wsdlsoap:binding style="document"
              transport="http://schemas.xmlsoap.org/soap/http"/>
     <wsdl:operation name="ordering">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="orderingRequest">
           <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="orderingResponse">
           <wsdlsoap:body use="literal"/>
        </wsdl:output>
     </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="OrderServiceService">
  <wsdl:port binding="impl:OrderServiceSoapBinding" name="OrderService">
   <wsdlsoap:address
    location ="http://localhost:8080/ServiceComposition/services/OrderService"/>
     </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

制造服务 ManufactureService 的描述文件 WSDL 描述和发送服务 OrderService 的 WSDL 描述文件类似于订单服务的 WSDL 文件。

5.7 基于语义服务的组合

利用已经搭建好的语义 Web 服务，进行语义 Web 服务进行组合，对三个原子服务分别是订单服务，制造服务，发送服务；订单原子服务地址位于 http://localhost:8080/ServiceComposition/services/OrderService?wsdl，其中包括了

order()订单的生成方法，由用户信息和产品信息生成订单，制造服务地址位于 http://localhost:8080/ServiceComposition/services/ManufactureService?wsdl，其中的方法为 manufacturing()接受到订单服务的结果后，进行制造；发送服务的地址位于 http://localhost:8080/ServiceComposition/services/SendService?wsdl，包括的方法是 sending()接受到制造服务的完成结果，进行发送产品。现在我们来对对这三个服务进行组合。

将 WSDL 文件转换为 OWL 文件：WSDL 用于精确的描述了 Web Services 三个方面的问题，Web Services 做什么？即描述了服务提供的操作，Web Services 驻留在那里？即描述如何访问服务，如 URL 等具体协议的地址信息，Web Services 如何进行调用？即描述调用的方式，如数据格式的详细信息、访问服务操作所需的协议的详细信息等。这些功能转换为相应的 OWL-S 本体文件，每一方法对已一个 owl 文件，按照服务中的方法来进行服务组合，我们对订单服务 http://localhost:8080/ServiceComposition/services/OrderService?wsdl 导入转化为 OWL-S 的本体文件，如图 5.7 所示

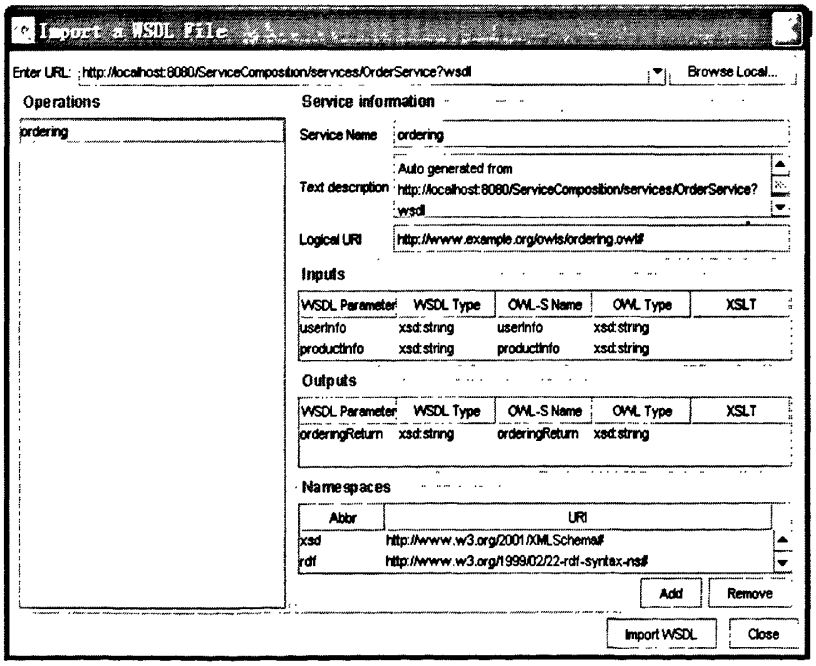

图 5.7 WSDL 转化为 OWL-S 本体

对应生成的文件为 ordering.owl，语义服务的描述的关键代码如下：

```
<?xml version="1.0" encoding="GBK"?>
<rdf:RDF
    ……
```

```
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
    xml:base="http://www.example.org/owls/ordering.owl#">
  <service:Service rdf:ID="orderingService">
    <service:presents>
      <profile:Profile rdf:ID="orderingProfile"/>
    </service:presents>

  </service:Service>
  <profile:Profile rdf:about="#orderingProfile">
    <profile:hasInput>
      <process:Input rdf:ID="userInfo">
        <process:parameterType rdf:datatype
              ="http://www.w3.org/2001/XMLSchema#anyURI">
            http://www.w3.org/2001/XMLSchema#string
        </process:parameterType>
        <rdfs:label>userInfo</rdfs:label>
      </process:Input>
    </profile:hasInput>
    <service:presentedBy rdf:resource="#orderingService"/>
    <profile:hasOutput>
      <process:Output rdf:ID="orderingReturn">
        <rdfs:label>orderingReturn</rdfs:label>

      </process:Output>
    </profile:hasOutput>
    <profile:hasInput>
      <process:Input rdf:ID="productInfo">
        ......
      </process:Input>
    </profile:hasInput>
    <profile:textDescription>
      </profile:textDescription>
    <profile:serviceName>ordering</profile:serviceName>
```

```
</profile:Profile>
<process:AtomicProcess rdf:about="#orderingProcess">
  <process:hasInput rdf:resource="#productInfo"/>
  <process:hasOutput rdf:resource="#orderingReturn"/>
  <process:hasInput rdf:resource="#userInfo"/>
  <service:describes rdf:resource="#orderingService"/>
  <rdfs:label>orderingProcess</rdfs:label>
</process:AtomicProcess>
<grounding:WsdlGrounding rdf:about="#orderingGrounding">
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding
      rdf:ID="orderingAtomicProcessGrounding"/>
    </grounding:hasAtomicProcessGrounding>
  <service:supportedBy rdf:resource="#orderingService"/>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding
rdf:about="#orderingAtomicProcessGrounding">
  <grounding:owlsProcess rdf:resource="#orderingProcess"/>
  <grounding:wsdlOutput>
    <grounding:WsdlOutputMessageMap>

    </grounding:WsdlOutputMessageMap>
  </grounding:wsdlOutput>
  ......
     <grounding:WsdlOperationRef>
     ......
      </grounding:operation>
    </grounding:WsdlOperationRef>
   </grounding:wsdlOperation>
  <grounding:wsdlInput>
    <grounding:WsdlInputMessageMap>
      <grounding:wsdlMessagePart rdf:datatype
       ......
      <grounding:owlsParameter rdf:resource="#productInfo"/>
```

```
        </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    ......
  </grounding:WsdlAtomicProcessGrounding>
</rdf:RDF>
```

与订单服务类似，对制造服务和发送服务同样生成 manufacturing.owl 和 sending.owl 文件，所有的原子服务全部导入如图 5.8 所示，给进行服务组合提供了必要的原子服务，要进行服务组合，必须保证原子服务的可靠性和完整性。

使用 OWL-S Editor 按照业务逻辑的组合方式来进行服务组合，按照业务流程由订单服务到制造服务，再到送货服务，来进行组合。具体是先建立 CompositionService 组合本体文件，对应的三个方面的描述文件为 myprofile，myprocess 和 mygrounding。CompositionService 的 describedBy 的属性值为 myprocess，presents 的属性值为 myprofile，supports 的属性值为 mygrounding。在 myprocess 对三个原子服务按照流程进行组合，由订单服务开始，接受到用户信息和产品信息后，产生订单；把订单交给制造服务，由制造服务完成产品

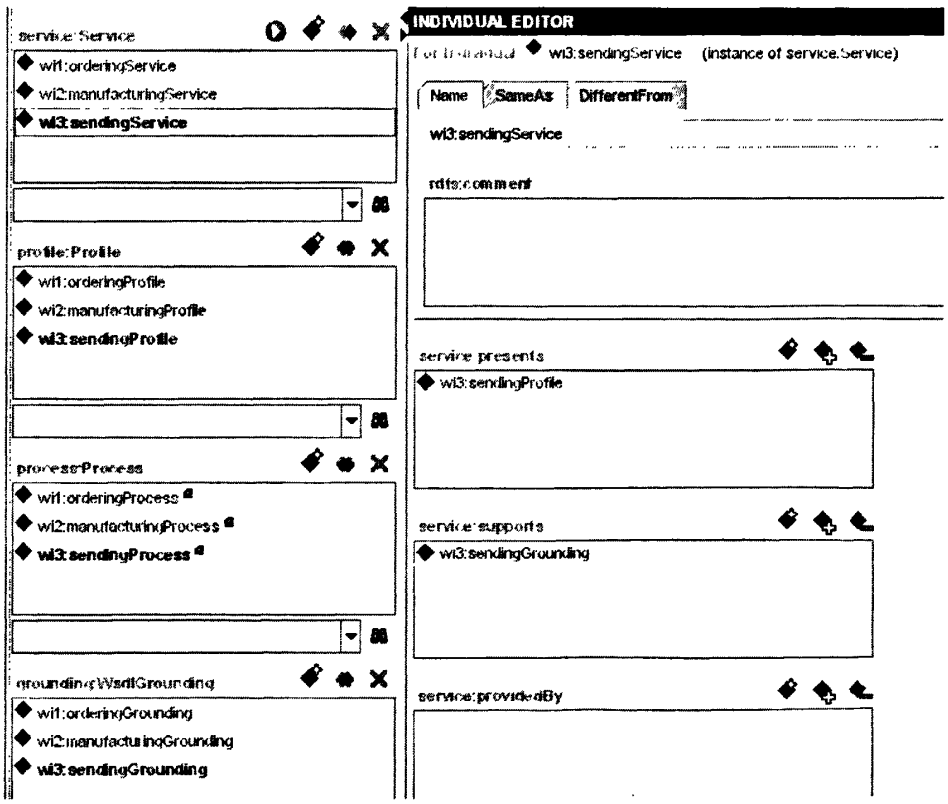

图 5.8　全部的原子服务的导入

制造，把产品输出给发货服务，发货服务接受到产品后调用发货服务进行货物

发送，整个组合服务就完成。在进行组合前，对三个单个的原子服务进行发布和单独调用进行测试，保证原子服务的完整性。

组合后的如图 5.9 所示：对服务的流程上有三个原子服务构成了整个大的业务流程。整个执行的过程用文件 CompositionService.owl 来描述。

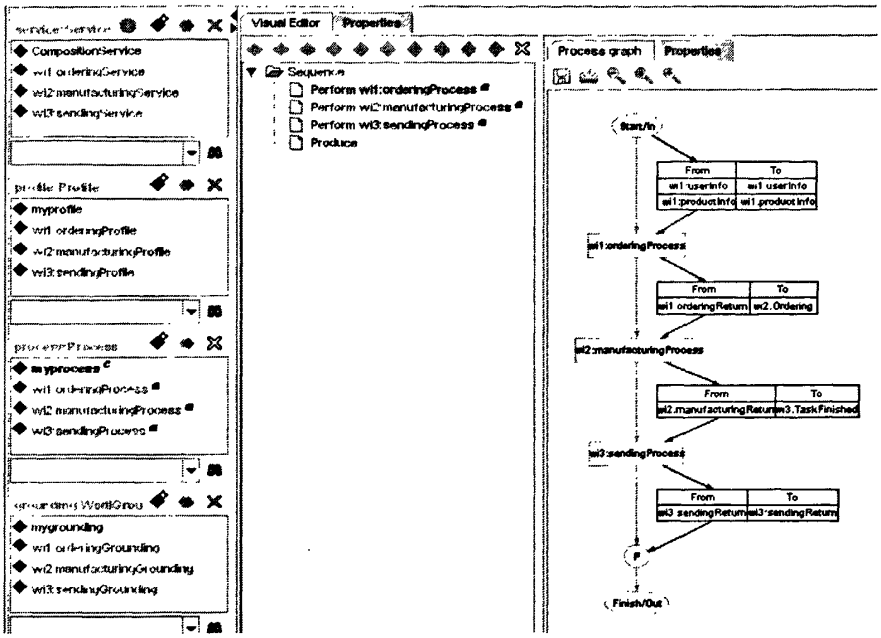

图 5.9 对原子服务组合生成组合服务

执行组合服务 CompositionService，如图 5.10 所示，完成了业务需求的组合流程结果。

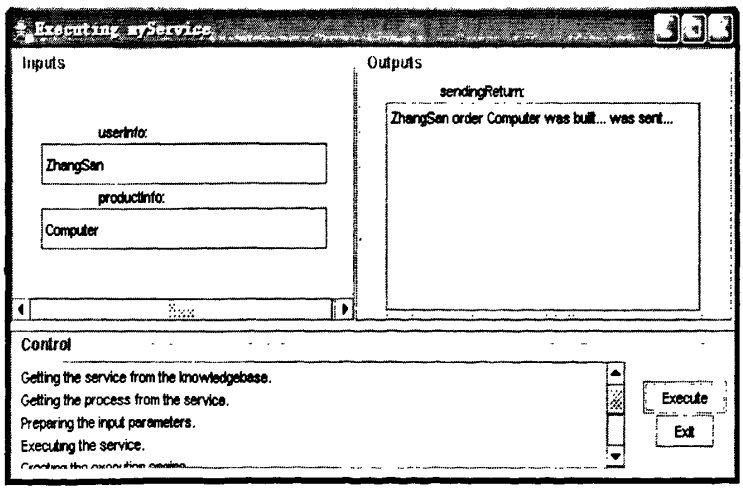

图 5.10 组合服务执行的结果验证图

5.8 本章小结

本章对语义服务组合模型的实现上进行具体的验证过程，从实验的背景，实验所需的工具进行介绍的基础上，搭建了语义 Web 服务组合的实验环境，对网路制造环境中领域本体进行创建，在本体中应用推理机进行推理验证，然后对本体知识进行封装，得到语义 Web 服务。在对语义 Web 服务进行详细描述的基础上，采用第 4 章中所建立的语义 Web 服务模型的来进行服务组合实验将三个基本的原子服务进行组合。实验结果表明实现了语义 Web 服务的组合，还有待进一步进行更加详尽的完善。

6 结 论

6.1 总结

本文首先了网络制造环境下进行服务组合研究背景和研究意义，对国内外Web服务组合研究的现状进行了深入的分析，并对服务组合的各种方法和思想进行了详细的总结和归纳。接着介绍了语义Web服务的理论基础和其支撑技术，包括语义网的体系结构及各层的特点和功能，本体的概念，本体的形式化，本体的建模和本体的建模技术及实例等。针对Web服务发展过程中面临的问题：Web服务对语义功能的急切需求，在介绍了语义网及本体技术的基础上，并着重对语义服务及其描述本体OWL-S进行了详细的研究，指出了OWL-S必须与传统的Web服务规范相结合，才能实现语义Web服务的发现、选择、匹配和组合的目标。

本课题的主要研究工作包括：

1)在介绍了语义网及其体系结构的基础上，详细的分析了本体的建模过程，从本体的概念、建模、形式化表达到本体的实例；并对语义Web服务及其组合的框架 OWL-S 的顶层服务本体模型进行深入研究的基础上，详细分析了OWL-S与WSDL和UDDI的关系，为给语义Web技术和Web服务技术的结合奠定了理论基础。

2) 阐述了在网络制造环境中进行服务组合的前提是先进行语义 Web 服务的发现选择和匹配，对通过本体加入语义后，服务进行匹配时如何按照匹配要求对服务进行选择匹配，在需求解析的约束下，进行服务组合，并对服务组合算法进行了详细的描述和分析。

3) 提出了一种语义Web服务组合模型，通过该模型来处理Web服务在语义功能下，推理机根据规则从领域本体和功能本体中得到严格的本体服务描述，提高了服务发现和查找的效率，使得服务注册中心可以根据语义功能而进行服务匹配和组合。

4) 在针对语义Web服务组合的研究，提出一种语义Web服务的组合模型，以网络制造环境为背景，搭建组合模型实验系统，对组合模型进行合理性验证，来评估其实用价值。

总体的来说，本文的语义Web服务组合模型是在语义网和Web服务两者结合的基础上，对Web服务的发现、匹配、选择和组合进行了研究和实验。但是，本课题仅仅提出了模型的简单化验证，所设计的实验环境也仅仅基于简单的几

个点，关于模型的具体应用仍然需要大量和更加深入的研究工作。

6.2 展望

本课题的研究工作只是一个简单的起步研究工作。作为当前的发展和研究热点，对语义 Web 服务组合模型和组合框架的研究工作，无论是在广度方面的研究，还是在深度方面的研究，都需要更进一步的加强和继续深入。随着语义 Web 服务技术的不断发展，其应用的领域在不断扩大，面对大规模的应用集成系统，这些简单的组合方式及组合模型在动态复杂多变的应用环境中，无法满足大量应用的急切需求，接下来的研究工作需要继续深入细致的研究，需要来不断完善和不断地改进。这里对未来的研究工作做一个简单的概述：

1）通过研究和分析可以看出，所提出的组合模型都具有一定局限性，适用的范围有限，或是只是针对某个方面的具体应用。然而，一个完整的系统的语义服务组合平台的研究应该是综合性的，考虑到整个系统的方方面面。再者，在组合思路和方法上，还可以考虑综合不同的学科的知识以提高服务组合的效率和性能。

2）在研究服务组合模型时，涉及到领域本体的语义完整性和规范性，领域本体设计的太大会影响系统的性能，设计的太小出要进行本体的合并和语义消解，仍然需要通过人为设置的方法来进行实现。在未来的研究工作中，需要引入更先进的方法使得本体的建立和本体的合并自动化。

3）在实际应用方面，本课题提出的语义 Web 服务组合模型仍然需要在更深层面的进行分析和研究。实际应用中出现的情况会更复杂，比如服务调用的并行性，服务协调工作中服务的状态管理，及在调用的过程中服务失败的情况等，这些都有待于更进一步的深入细致的研究。

参考文献

[1] Biplav Srivastava, Jana Koehler，Web Service Composition – Current Solutins and Open Problems [C].International World Wide Web Conference. New York:ACM Press.2004:312-313.

[2] 袁庆霓,谢庆生等.基于语义的制造资源本体的建模技术研究[J].武汉理工大学学报,2009,5(10):121-125.

[3] Kim SM, Rosu MC..A survey of public web services[C].International World Wide Web Conference. New York : ACM Press.2004:312-313.

[4] 岳昆,王晓玲,周傲英.Web 服务核心支撑技术：研究综述[J].软件学报,2004,14(7):428-442.

[5] 向阳,王敏马甄.基于 Jena 的本体构建方法研究[J].计算机工程,2007,32(21):59-61.

[6] 张友生等编著.软件体系结构[M].第 2 版.北京:清华大学出版社,2007,5.

[7] 喻坚,韩燕波著.面向服务的计算——原理和应用[M].北京:清华大学出版社,2006,12.

[8] 袁庆霓.基于网络制造环境的制造资源共享服务语义关键技术 [D].西南交通大学,博士学位论文,2010.

[9] 毕强，韩毅.基于 OWL-S API 的数字图书馆服务组合应用研究[J].现代图书情报技术, 2009.12(19)28-33.

[10] 韩毅.基于 OWL-S 的数字图书馆服务组合应用研究[J].图书情报工作,2009.23(6)26-30.

[11] 周亮.基于OWL-S的Web服务发现[D].南京理工大学,硕士学位论文,2005.

[12] 王鹏.基于本体的 Web 服务发现关键技术与模型研究 [D].大连海事大学,硕士学位论文,2008.

[13] 梁本志.基于 SOA 架构服务组合的研究与实现[D].硕士学位论文, 2008.

[14] 刘家茂,顾宁,施伯乐.基于 Mediator 的 Web Services 无回溯反向链动态合成[J].计算机研究与发展, 2005.23(6)1153-1158.

[15]刘峰,谭庆平,杨艳萍.基于图论的 Web 服务合成算法[J].华中科技大学学报,2005.23(6)202-204.

[16] 杨晓英.基于 OWL-S 的 Web 服务语义描述方法的研究[D].北京邮电大学.硕士学位论文,2009.

[17] 王艳丽.基于 Qos 和语义的 Web 服务组合研究[D].重庆大学.硕士学位论

文,2010.

[18] 徐利谋,金可音,阳辉,汤双全.基于 OWL-S 的服务发现算法研究[J].计算机工 程与科学, 2007.23(6)64-67.

[19] 丁敏峰,吴波著.网络制造资源管理系统建模和开发[M].北京:化学出版社,2007,8.

[20] 龚玲,张云著.Web 服务原理和技术[M].北京:机械工业出版涩会.2010.1

[21] 汤益华.基于 OWL-S 的语义 Web 服务自动组合研究[D].中南大学,硕士学论文,2010.

[22]任平.一种语义 Web 服务注册中心的设计和实现[D].吉林大学,硕士学位论文,2009.

[23]杨亮.语义 Web 服务匹配模型的研究与实现[D].北京邮电大学.硕士学位论文, 2009.

[24] 贺文锐,何卫平.基于 Web Services 的网络化制造资源管理的关键技术[J].计算机研究与发展, 2004.27(11)1182-1386.

[25] 林清澄,彭文灵.基于 OWL-S 的 Web 服务匹配方法及其实现[J].微计算机应用, 2006.27(4)496-498.

[26] 杨晓英.基于本体的动态 Web 服务组合的研究与应用[D].大连理工大学.硕士学位论文,2008.

[27] 农嘉.基于 Web 的服务组合模型研究[D].桂林电子科技大学.硕士学位论文,2009.

[28]牟欣涛.基于语义的 Web 服务组合框架研究[D].中国海洋大学.硕士学位论文,2008.

[29] 董金祥.基于语义面向服务的知识管理与处理[M]:浙江大学出版社,2009,8.

[30] 陈小平著.语义网基础教程[M].北京:机械工业出版涩会.2008.4

[31] 闫宇.领域本体中的规则推理研究与实现[D].华东师范大学.硕士学位论文,2009.

[32]焦方俊,陈维斌.基于 OWL-S 扩展 UDDI 的研究 [J].计算机应用, 2007. 27(4) 231-234,238.

[33] Kim SM, Rosu MC..A survey of public web services[C].International World Wide Web Conference. New York : ACM Press.2004:312-313.

[34] 邓志鸿,唐世渭，张铭，杨冬青，陈捷.Ontology 研究综述[J].北京大学学报,2002,5(38):730-738.

[35]袁庆霓.基于网络制造环境的制造资源共享服务语义关键技术 [D].西南交通大学,博士学位论文,2010.

[36] 林清滢.应用 OWL-S 实现 Web 服务合成的语义描述 [J].信息技术, 2006. 7(4) 231-234,238.

[37] 吴乃鑫.本体理论在 Web 服务中的应用 [D].合肥工业大学,硕士学位论文,2008.

[38] 谭伟,范玉顺.网络化制造环境下服务匹配与合成问题研究 [J].计算机集成制造系统, 2005. 11(10) 1408-1413.

[39] 疏剑,谢庆生.制造业领域本体构建方法研究 [J].机械与电子, 2008. 11(10) 64-66,68.

[40] Eric Newcomer, Greg Lomow, Understanding SOA with Web Services. Addison Wesley Professional, 2004

[41] Christensen E,Curbera F,Meredith G,et al. Web Services Description Language[EB/OL].[2008-04-12].http://www.w3.org/TR/wsdl.

[42]Colan Mark.Serviee-Oriented Architecture expands the vision of Web Services, [EB/OL].http://www.ibm.com/redbooks,2004.04

[43] 柴晓路著.Web 服务架构与开发互操作技术[M].北京:清华大学出版社,2002.6

[44] 朱振杰著.SOA 的关键技术的研究与应用实现[M].成都:电子科技大学,2006.

[45] 梁爱虎著.精通 SOA[M].北京:电子工业出版社,2007.

[46] Ed Ort,Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools, Sun Microsystems, 2005.

[47] 顾宁,刘家茂,柴晓路著.Web services 原理与研究实践.北京:机械工业出版社,2006

[48] Eric Newcomer, Greg Lomow, Understanding SOA with Web Services. Addison Wesley Professional, 2004

[49] Christensen E,Curbera F,Meredith G,et al. Web Services Description Language[EB/OL].[2008-04-12].http://www.w3.org/TR/wsdl.

[50]周晖.基于 Web 服务的企业应用集成技术研究[D].浙江:浙江大学硕士学位论文,2004.3.

[51] Ed Ort,Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools, Sun Microsystems, 2005.

[52] Run S..A model for web services discovery with QoS. ACM SIGecom Exchanges.2003,4(1):1-10.

攻读硕士学位期间发表的论文

[1] 郭永社,王长元.基于 Globus Toolkit4 的网格复合资源实现 [J].西安工业大学学报增刊(陕西).2010.12(30):85-87.

致 谢

感谢导师王长元教授三年来以来的关心、指导和教诲。王长元教授追求真理、献身科学、严以律已、宽以待人的崇高品质对我将是永远的鞭策。

感谢导师王长元教授。从学位论文选题、开题报告、研究设计、中期考核到论文撰写，王长元老师始终给予学生细心的指导和不懈的支持。导师渊博的学识、严谨的治学态度、兢兢业业的工作作风、乐观积极的人生信条，都深深地影响着我，使我受益匪浅，终生难忘。在此，我再次向导师表示衷心的感谢和深深的敬意！

感谢西安工业大学给予我良好的学习和科研环境，感谢研究生部给予我生活和学习上的帮助。感谢支持和关心老师、师兄、师姐，感谢我们实验室的同学，感谢在论文工作中所有帮助过我的人。是您们的支持和关心，鞭策和激励着我在奋进，语言难以表达，感激之情，永存心中。

感谢家人对我的理解和鼓励，使我能够顺利完成学业。

感谢百忙之中评阅论文和参加答辩的各位专家、教授和老师！

学位论文知识产权声明

本人完全了解西安工业大学有关保护知识产权的规定，即：研究生在校攻读学位期间学位论文工作的知识产权属于西安工业大学。本人保证毕业离校后，使用学位论文工作成果或用学位论文工作成果发表论文时署名单位仍然为西安工业大学。学校有权保留送（提）交的学位论文，并对学位论文进行二次文献加工供其他读者查阅和借阅；学校可以在网络上公布学位论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存学位论文。

（保密的学位论文在解密后应遵守此规定）

学位论文作者签名：郭永社

指导教师签名：

日期：2011、4、23.

学位论文独创性声明

秉承学校严谨的学风与优良的科学道德，本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，学位论文中不包含其他人已经发表或撰写过的成果，不包含本人已申请学位或他人已申请学位或其他用途使用过的成果。与我一同工作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了致谢。

学位论文与资料若有不实之处，本人承担一切相关责任。

学位论文作者签名：郭永社

指导教师签名：

日期：2011.4.23