论文题目：基于 SOA 企业应用集成与设计的研究

专　　业：计算机应用技术

硕 士 生：吕鸣剑(签名)

指导教师：孟东升(签名)

摘　要

面向服务架构(SOA)是新一代的架构思想，用于分布式软件开发。由于 SOA 具有良好的松耦合、与平台无关等特性，很好的解决了系统的灵活性和互操作问题，因而具有广泛的应用。目前，作为企业新系统架构和企业应用集成的主要解决办法，SOA 正在成为研究领域的热点。在未来的软件开发世界里，SOA 将成为软件体系结构领域的统领者。

本文对企业应用集成 EAI 传统的实现技术进行了学习研究，分析了面向服务的软件体系结构出现的必然性，及其特点和优势。为了选择 SOA 的实现技术，特别对 SOA 在 Internet 环境下的实现技术——Web Service 进行了学习研究，主要包括 Web Service 的基础知识和协议规范，并将 Web Service 技术与传统的分布式计算技术进行了比较，更加突出 Web Service 的与平台无关、易于扩展、易于集成等特点。

本文结合 SOA 的设计理论知识，综合运用 DCOM、Web Service 等技术，研究提出了基于 SOA 的企业应用集成的一般模型。该框架模型可以在现有各种异构企业平台上构建一个平台无关、语言无关的中间层，以支持不同平台应用彼此无缝地连接和集成，并以一种松散的服务捆绑形式，快速、低代价地开发和绑定各种企业应用。另外，本文对 Web Service 互操作、旧有系统的 Web Service 转化问题做了研究和讨论。针对 Web 服务的安全问题，提出了利用 SOAP 头条目进行验证的解决办法；对于企业私有 UDDI 注册中心的组建，提出了实用，快捷的实现方法。

最后，结合中石油股份公司的基金项目管理系统集成案例，根据该企业自身的特点，在.NET 开发平台下，实践了基于 SOA 的企业应用开发。

关键词：　面向服务架构 企业应用集成 简单对象访问协议 Web Service

论文类型：应用研究

Subject: **The Study of Enterprise Application Integration and Design Based on SOA**
Specialty: **Computer Application Technology**
Name: **Lv Mingjian** **(signature)** Lv Ming Jian
Instructor: **Meng Dongsheng(signature)** Meng Dong Sheng

ABSTRACT

Service-Oriented Architecture (SOA) is a new generation of architecture ideology applying for distributed software development. Because SOA has solved wonderfully the problems of flexibility and interconnectivity for applications with its features of loose coupling, independent of platform etc, it will be applied widely. As a major solution to Enterprise Application Integration and new system architecture, SOA has been the hot topic at present. It will be the dominator of software architecture.

This dissertation introduces the traditional technologies of EAI and analyses the inevitability of the emergence of Service-Oriented Architecture ,including its features and advantages. Then, it lucubrates implementation technology of SOA under Internet-Web Service, mainly including the basis knowledge and protocols and criterions of Web Service, illuminates the traits of Web Service, such as platform independently、extendible and integration easily.

By combining with theoretical knowledge of the SOA and the integrated use of DCOM, Web Service technologies the dissertation propose a model for enterprise application integration based on the SOA. It is provided to build a middle layer which has nothing with any platform and language, so it can make the enterprise applications integrated together without gap and can develop and bind all kinds of applications quickly by loose-couple and low-cost way. This dissertation also discusses some problems about interoperability and Web Service transaction method of old enterprise information system. In additon, the solution of SOAP headlines validation was brought forward against Web Service security issues and a practical and efficient method was proposed for private enterprises UDDI registration center.

Finally, making an integration case of program management for a corporation, carries out the enterprise application development based on SOA under .NET platform according to the corporation's characteristics.

Keywords: SOA，EAI，SOAP, Web Service
Thesis: **Application Study**

学位论文创新性声明

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中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安石油大学或其它教育机构的学位或证书而使用过的材料。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做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：吕鸣剑　　日期：2007.5.15

学位论文使用授权的说明

本人完全了解西安石油大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安石油大学。学校享有以任何方法发表、复制、公开阅览、借阅以及申请专利等权利。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安石油大学。

论文作者签名：吕鸣剑　　日期：2007.5.15

导　师　签名：[illegible]　　日期：2007.5.15

第一章 绪论

1.1 论文的研究背景

随着企业计算的发展，企业级应用需求要求新的软件系统不再是从底层做起，而只是依据企业逻辑需求重新组织已有的数据存储，将现有的数据和事务通过新的渠道，比如 Internet 浏览器或者手持设备呈现给用户。另外，为了提高企业计算的高效性、可用性、规模性，现有许多的操作系统都是分布式操作系统，运行在许多机器之上。这样的企业级解决方案就必须协调运行在群组硬件之上，实现这种系统的一种方法就是将该系统组织成群组服务的模式，每一个服务都提供一组定义良好的功能集合。整个系统其实就被设计和实现为一组相互交互的服务，而将功能以服务的形式展现出来是该系统灵活性的关键。它使得系统中的某些服务能够充分利用其他的服务同时却无需考虑其物理位置。系统通过添加新的服务来不断的升级，这样就应运而生了面向服务的体系结构(Service Oriented Architecture, SOA)。SOA 定义了构成系统的服务，通过描述服务之间的交互提供特定的功能特性，并且将服务映射为具体的某种实现技术。

面向服务的体系结构(SOA)建立在分布式计算技术的基础上，这种体系结构本质上是动态的，它提供对服务的登记、发现和调用的支持。SOA 提供了一种标准的系统模型，使得驻留在网络上的企业应用能够被发布(Publish)，发现(Discover)和调用(Invoke)。SOA 的软件开发人员可以将企业应用系统以服务的形式通过网络发布，即任何服务应用程序都可以同其它位置的基于服务的应用系统交互。

1.2 研究的目的和意义

一般企业应用都是以应用为出发点去设计和开发的，也就是以公司的部门为基本单位设计和开发管理系统，各个系统之间相互独立，互不相扰。随着公司规模的扩大，独立系统的功能和数量一直都在发生着变化。IT 技术正在迅猛的发展，子系统的开发时间不同，所采用的技术自然也很有可能不同。一个业务请求很难在不同开发工具实现的子系统间实现有效的调用。

伴随着经济的飞速发展，企业的规模在不断地扩大，企业信息化所需的软件复杂度也在增加，传统的软件体系结构暴露出了它们的弊端，新的体系结构正在崭露头角，解决了过去软件开发中出现的种种问题。SOA 架构为软件体系结构注入了新的生命活力，解决了软件开发中存在的互操作性、扩展性等问题。Web Service 是面向服务的分布式计算框架，具有松散耦合、与平台无关、易于集成等优点，为 Internet 上的分布式应用提供了有效的支持[1]。

1.3 国内外研究现状

面向服务的体系结构是基于“软件变服务”思想，提出了一种新的解决软件重用和软件集成的方案。通过采用面向服务的体系结构，企业能够迅速便捷地构建开放的、模块化的、可重用、与平台无关、可扩展的应用系统。作为 SOA 的一种实现手段，Web 服务提供了基于 XML 标准接口的若干中间件，具有完好的封装性、松散的耦合性、协

议规范的标准性、以及高度的可集成性等特点，能够很好的满足 SOA 应用模式需求。传统的中间件厂商对 Web 服务的支持也是不遗余力。IBM 公司很早就推出了 WebSphere 产品，支持各种有关的 Web 服务标准：Java 技术的创立者 Sun 公司新提出了开放式软件架构 Sun-ONE，力图融合 Java 和 XML，而 Microsoft 公司的.NET 战略即以 XML 为基础，其新发布的 Visual Studio.NET 将成为 Web 服务的主要产品。针对业已公布的标准，许多大型企业(IBM，Microsoft，Sun，BEA 等)开始着手对基于 Web 服务的面向服务的体系架构予以实现和推广。

伴随着面向服务架构应用的稳健步伐，各个公司在实现各种 Web Service 和将已有应用转换成面向服务架构上取得了重大的进展[2]。在面向服务架构的研究中，研究有着眼于整个架构设计的原则与模型的讨论，也有针对具体的服务细节进行讨论。国内外已经有一些案例，将 SOA 的设计思想应用到医疗、电信、金融等各个行业[3-5]。 SOA 在德国邮政系统和金融行业中都得到了应用[6-7]。就 Web 服务核心支撑技术的研究而言，存在很多有待解决的开发问题。例如，在组合 Web 服务的实施方案中，服务组件或基本服务的定位、协调、通信及调用策略，服务执行结果的评估和正确性验证，高效服务质量管理策略的探索及服务质量代价模型的建立等[8]。对于具体的研究细节，服务的自动组合、服务设计的粒度、服务的契约，乃至服务的分析与设计都是研究的热点。德国的波茨坦大学主要研究了将人工智能技术应用于 SOA 的服务自动组合中。国家 863/CIMS 主题资助项目基于本体的 Web 服务查找和合成技术研究。通过应用本体领域分类概念和 DAML 语义标签，对 Web 服务进行标准化表示。

1.4 本文的主要研究内容和内容组织

SOA 和 Web Service 技术是本文主要的研究对象，SOA 在 Internet 环境下的企业应用，主要采用的技术手段是 Web Service。由于 Web 服务全部的规范、技术都是以 XML 作为底层核心和架构基础，以达到 Web 服务平台、语言和发布者能够相互独立的目的。本文主要针对基于 SOA 的企业应用集成和设计展开了研究。Visual Studio.NET 是 Microsoft 推出的一套完整的开发工具和集成平台，利用它可以方便、快速地开发出 C/S 的 WIN 应用程序和 Web 服务应用程序。本文详细研究如何利用 SOA 设计出共享性高和可扩展性好的企业应用系统，并以实例说明如何在.NET 环境下，采用 Web Service 技术来实现 SOA 集成。并探讨了 VB,Delphi 等已有各种语言的信息系统向 Web Service 转化问题，以及几种集成平台之间的 Web Service 互操作问题。

1.5 论文结构和章节安排

本章主要介绍了论文的研究背景及 SOA 的目的意义。

第二章应用集成首先介绍了应用集成的基本概念，然后分别从集成分类、集成实现架构方面详细讨论了传统的应用集成，并分析了传统应用集成的优点和不足。在指出传统 EAI 缺点的基础上，对 SOA 的概念、特点、编程模型、以及 SOA 架构的设计原则、分析方法、实现步骤等一系列理论进行了论述。

第三章介绍SOA的实现技术Web Service，描述了SOA和Web Service之间的关系、Web Service的协议栈和实现Web服务的关键技术。并对SOA中其他实现方法如CORBA、DCOM等进行了分析比较，指出了Web Service做为SOA架构主流实现技术的各种优点。最后，介绍了BPEL、ESB其他SOA相关技术。

第四章在第二、三章的基础上提出了一个SOA集成框架，本章中结合SOA架构的分析与设计方法和Web Service体系结构中多层体系结构的思想，将SOA架构分为客户访问层(Client Request Layer)、企业应用层(Enterprise Application Layer)、企业应用集成层(Enterprise Application Integration Layer)、消息传输层(Message Transport Layer)、服务提供与封装层(Service Provide and EncapsulationLayer)。并对每层的作用和实现技术进行了详细的分析和说明。另外对框架中的Web Service互操作问题进行了较深入的研究讨论，给出了对应的解决办法；对框架中的旧有系统Web Service转化问题，给出了COM方式的解决方案。

第五章中结合具体的应用实例—中石油股份公司基金项目管理系统和专家评审系统，分析了如何在实际的企业应用中利用Web Service技术实现SOA。本章中介绍了应用的背景、分析了企业的具体需求和业务流程。根据第四章提出的基于SOA的企业应用模型，在.NET下实践了SOA。

第六章总结全文，并对下一步的研究工作进行了展望。

第二章 面向服务的应用集成架构

2.1 企业应用集成的分类

随着计算机软、硬件技术的发展，特别是 Intranet 和 Internet 的出现，计算机在企业的业务系统中起着越来越重要的作用。一方面，基于 Intranet 的企业信息化已具有一定基础，采购、生产、销售、财务、售后服务以及人事等环节都在采用各种计算机应用系统参与管理。另一方面，许多企业在酝酿和实施基于 Internet 的电子商务，他们也迫切希望能够实现电子商务系统与企业内部现有应用系统间的无缝连接。

在企业信息化建设的过程中存在自动化孤岛的情况下，显然重新设计并实现所有的应用并使它们能够协同工作，形成一个整体并不是一个最好的方法，存在如下的弊端：

- 需要消耗大量的人力物力对企业各部门的业务流程进行重新调研。
- 耗费大量的时间进行新系统的设计和开发不利于企业在商业竞争中迅速提高综合实力。
- 员工需要重新适应新系统的操作规程，引入了新的培训和操作成本。

在考虑了这样一些弊端之后，企业从整合自身业务流程，利用现有资源等方面对应用开发方法提出了新的要求，正是在这样一种情况下面，企业应用集成 EAI 应运而生。

企业应用集成是对组织中完成不同业务功能的应用系统进行集成，在它们之间建立起可供数据交流和应用沟通的中枢系统，使用户可以透明地访问各个不同应用程序，展现给客户的数据仿佛来自于一个统一的数据源。文献[9]中给的应用集成框架原型如图 2-1 所示。从集成层次上分可以分为以下几类：

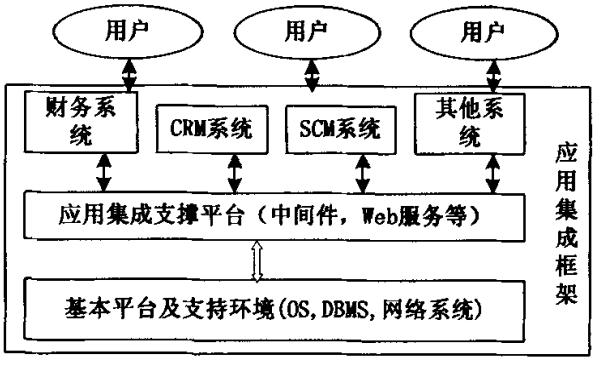

图 2-1 应用集成框架

1．面向数据的集成

数据集成聚焦于接口层次的应用和系统间的数据转化和传输，它给了大多数组织一种风险较低的切入企业应用集成的方式，其主要优势是较低的成本(因为在大多数情况下不需要修改应用程序)。

面向数据的集成模式将集成视为一种数据流系统，数据可以在文件、数据库以及其它信息库间流动，可以在应用间通过 API(如 SAP 的 R/3 BAPI)流动，也可以在通信中介

间流动。因此，实现对数据库、应用程序以及相关服务的接口就成为面向数据集成的关键问题。具体来讲，面向数据的集成方法又可以划分为两种类别—数据复制、数据聚合。

数据复制方式的目的是为了保持数据在不同数据库间的一致性，而数据库可以是同一厂商也可以是不同厂商的，甚至可以是采用了不同模型和管理模式的数据库。

数据聚合是将多个数据库和数据库模型集成为一种统一的数据库视图的方法。也可以认为，数据聚合体是一种虚拟的企业数据库，它包括了多个实体的物理数据库。数据聚合方法在分布的数据库和应用之间放置一个中间件层，该层与每一个后台的数据库用其自带的接口相连，并将分布的数据库映射为一种统一的虚拟数据库模型，而这种虚拟模型只在中间件中存在。应用就可以用该虚拟数据库去访问需要的信息。

2. 面向接口或方法集成

接口和方法集成包括直接的和严格的，在网络环境中的跨平台应用程序之间的应用到应用(A2A)的集成。它包括应用程序接口(API)、远端过程调用(RPC)、分布式对象、公共对象访问中介(CORBA)、Java 远端方法调用(RMI)、面向消息的中间件以及 Web 服务等各种软件技术。

接口和方法集成方法利用良好定义的应用接口实现对应用包和客户化应用的集成。它是目前得到最广泛应用的集成方法。在面向接口的集成中，它通过集成代理提供用以连接应用包和客户自开发应用的适配器来实现集成，适配器通过其开放或私有接口将信息从应用中提取出来。另外一些类型的适配器可以和应用通过面向消息的中间件(MOM),DBMS、文件系统或其它系统和应用间接相连，从而实现和应用的交互，甚至也可以屏蔽和应用间的信息传输。这种通过接口抽象的方法提供了集成不同类型应用的高效率，它也是面向接口集成方法的主要优势。

3. 面向过程的集成

面向过程的集成方法按照一定的顺序实现过程间的协调并实现数据在过程间的传输，其目标是通过实现企业相关业务过程的协调和协作实现业务活动的价值最大化。

面向过程的集成逻辑是一种过程流集成的思想，它不需要处理用户界面开发、数据库逻辑、事务逻辑等。过程逻辑和核心业务逻辑相分离，正是面向过程集成方法的最重要优势的体现，它可以通过改变应用程序和个人之间的信息流，而不改变应用程序本身，使应用更优化地为业务服务。

在实施面向过程的集成时，由于其实施对象会采用不同的元数据、平台以及业务应用类型，因此面向过程的集成技术必须具有足够的柔性，能够和不同的相关技术实现集成，如面向消息的中间件、面向事务的中间件、面向接口的集成代理等。在结构上，面向过程的集成方法在面向接口的集成方案之上，另外定义了过程逻辑层，而在该结构的底层，应用服务器、消息中间件提供了支持数据传输和跨过程协调的基础服务。面向过程的集成在某种程度上是一种对业务层次上的各种服务的抽象，因此，很多提供集成代理、消息中间件以及应用服务器的厂商也开始提供用于业务过程集成的产品。

2.2 传统的企业应用集成架构

2.2.1 点对点的集成

根据 META Group 的统计，经过相当一个时期不断的 IT 系统建设，一家典型的大型企业平均拥有 49 个应用系统，33%的 IT 预算是花在传统的集成上，通过零星的“点对点”连接，使众多的“信息孤岛”联系起来，以便让不同的系统之间交换信息。这使得企业的应用系统看起来像一张复杂的蛛网[10]。

在点对点的集成架构中，每个企业信息系统(Enterprise Information System，EIS)都紧密地与其它 EIS 通过它们的点对点连接在一起，它的优点是容易理解并且当只有少量系统需要集成时可以快速实现[11]。但如果一个 EIS 发生改变就会打破与它有关的应用集成；另一个缺点是每个 EIS 都要求有足够多的整合点来支持更多的系统集成，如果有 5 个互相集成的 EIS，就需要 10 个不同的整合点，如图 2—2 所示。因此，这种方法很难集成大量的应用系统，且集成的系统越多维护就越困难。

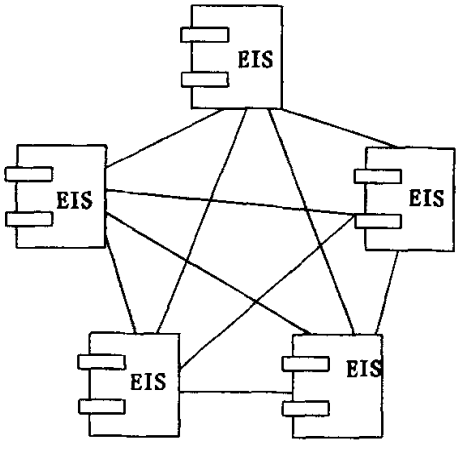

图 2-2 点对点企业应用集成

2.2.2 基于 MOM 的企业应用集成

MOM(Message Oriented Middleware)指的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网络应用系统的集成。目前流行的 MOM 中间件产品有 IBM MQSeries，Microsoft MSMQ，TibcoRendevous 等等。

MOM 产品在架构上又可分为以下两类：

- 基于 TCP 协议和队列管理的 MOM：如 MQSeries，MessageQ 等。这种 MOM 产品需要配置诸如队列连接通道等等信息，因而部署成本相比而言较高。
- 基于 IP 可靠多播的 MOMTibco/Rendevous，CenSoft CenEAI 等等，此类 MOM 产品在同一网段内采用多播协议，并在此基础上实现消息可靠传输。消息需要跨越不同网段的情况，采用 TCP 协议，这种架构的产品在配置部署上和可扩展能力上有较多的优势。

消息中间件的系统结构如图 2－3 所示，该系统结构支持点到点消息排队模型和推拉消息传递模型。

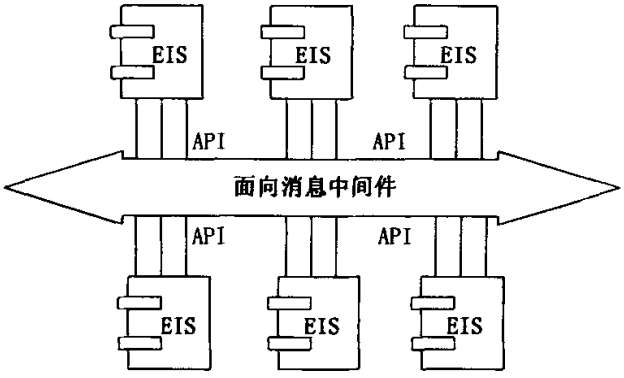

图 2-3 消息中间件的体系结构

MOM 与基于 CORBA/DCOM 的通信中间件相比，主要有以下优点：

- 消息异步通信，实现不同应用通信的位置透明，这种程序与网络复杂性的隔离，屏蔽了应用整合过程中网络通讯的复杂性问题。
- 提供自描述的消息格式，支持域操作，实现应用之间更灵活方便的信息数据交换。
- 通讯程序可在不同的时间运行。程序不在网络上直接相互通话，而是间接地将消息放入消息队列，因为程序间没有直接的联系。所以它们不必同时运行。消息放入适当的队列时，目标程序甚至根本不需要正在运行；即使目标程序在运行，也不意味着要立即处理该消息。

MOM 的缺点也比较明显，即规范较少，消息、格式大多是私有的，这样，不同的 MOM 产品之间并不能相互通信，需要有诸如连接 MQSeries 和 Tib/RV 的适配器。但是，随着 XML、JMS(Java Messaging Service)等规范的推出，MOM 的这一不足正在有所改善。

2.2.3 基于 J2EE/JCA 架构的企业应用集成

J2EE 应用服务器是如今企业应用集成中使用最广泛的一种集成平台。使用 J2EE 服务器作为集成 EIS 的平台的关键是如何实现 EIS 与 J2EE 平台的连接，以及如何在 J2EE 平台下开发和部署企业的业务逻辑。

JCA(J2EE 连接器架构，Java Connector Architecture)是用来简化 J2EE 应用服务器和 EIS 之间的集成而提出的一种规范。它定义了将 J2EE 应用服务器连接到不同的 EIS 的标准体系结构，解决了人们在面对与 EIS 系统集成时面临的许多问题。JCA 规范由以下三个部分组成：资源适配器、系统协议和通用客户接口。JCA 定义了两套标准的接口：一套接口是用于让资源适配器把兼容的应用服务器无缝的整合起来，另一套接口(通用客户接口)允许客户端用一种统一的方法使资源适配器与 EIS 连接。资源适配器为 EIS、

应用服务器和应用程序组件之间的交互提供连接。应用程序组件通过通用客户接口与资源适配器交互来访问 EIS。应用服务器和资源适配器一起提供对 EIS 访问时的连接、事务和安全的控制及管理。JCA 的作用主要体现在以下方面：

（1）简化了集成。

在 JCA 出现之前，每个 EIS 应用都有自己的编程接口，不同的 EIS 系统之间以及应用服务器和 EIS 系统之间的交互都意味着要对自身进行修改，从而针对一组特定的 API 编程。

而 JCA 框架使得任何实现了 JCA 规范的应用服务器都可以通过实现了 JCA 规范的某一 EIS 的资源适配器与该 EIS 资源相连；同时，也可使任意 EIS 资源通过它们自身实现了 JCA 规范的资源适配器与同样实现了 JCA 规范的某一应用服务器相连。

（2）为连接提供了 QoS 管理。

JCA 框架利用 J2EE 应用服务器中内建的事务、安全等服务，为连接操作的事务及等特性提供了支持。

JCA 的局限性和缺点在于：调用 EIS 应用时，JCA 采取同步消息传输方式；它不能处理来自 EIS 应用的异步消息或向 EIS 应用传递异步消息；JCA 没有提供基于 XML 的接口来实现与非 Java 的异构系统的集成；JCA 只定义了从客户端到 EIS 应用的单向调用，却没有定义 EIS 应用如何回调客户端。

2.2.4 传统集成技术的缺点

概括来讲 Internet 环境下传统集成技术的不足主要体现在以下三个方面。

（1）它们要求服务客户端与系统提供的服务本身之间必须紧密耦合，即要求一个同类的基本结构。这样的系统往往十分脆弱如果一端的执行机制发生变化，那么另一端便会崩溃。例如，如果服务器应用程序的接口发生更改那么客户端便会崩溃。

（2）基于 CORBA 等分布式对象技术不能实现透明地跨越防火墙通信。例如，防火墙有时会阻止 ORB 通信，也使得其不适合于在 Internet 环境企业应用集成的发展要求。

（3）传统的 DCOM，CORBA 或是 EJB 组件，由于每种组件都必须使用自己特定的规范来开发，组件之间的通信也必须使用特定的协议，这样不同组件之间无法进行直接的数据交换和数据共享[12]。

因此传统的集成技术由于其自身的局限性和互操作性问题，难以适应 Internet 环境下的企业应用集成的发展需求。

2.3 基于 SOA 的应用集成

2.3.1 SOA 思想理论

随着信息技术的发展，特别在计算机网络和 Internet 的时代出现了大量基于网络的大型分布式应用系统。随着公司业务的不断发展，对资源、数据的集中，决策支持统一的要求越来越急迫，需要将现有的多个应用系统进行集成和整合。并且，随着业务的快

速变化企业要应对竞争的新要求，需要不断更新业务流程和应用模式，建设新的应用系统，从技术上要求新的应用系统能快速搭建并实施，需要能够做到“随需应变”，由此面向服务的架构(SOA)应运而生。

SOA(Service-Oriented Architecture，面向服务的体系结构)不是一个新的概念，早在1996年，美国知名IT市场调研顾问公司Gartner最早提出SOA的预言，2002年12月，Gartner又提出了SOA是“现代应用开发领域最重要的课题”，并预计到2008年，SOA将成为占有绝对优势的软件工程实践方法，它将结束传统的整体软件体系架构长达40年的统治地位，主流企业现在就应该在理解和应用SOA开发技能方面进行投资。SOA到目前为止没有一个明确的定义，综合各方面的资料，本文给出的SOA定义如下：

1．SOA是组件系统模型

SOA软件系统是由不同的功能单元(称为服务)组装而成。服务之间是靠定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统、中间件容器和编程语言。这使得构建在这样的系统中的各种服务以统一和通用的方式进行交互。

2．SOA是Client/Server模型的扩展

SOA应用包含有服务的提供者(服务器端)和服务使用者(客户端)。SOA又不同于通常的C/S模型，它更强调的是软件组件之间的松散耦合和使用分散的标准接口。

3．SOA是概念模式，需要具体的实现技术

SOA不是新概念，已经存在了数年，但只是在出现了基于标准的集成技术(如Web Service)之后，SOA才开始被加速采用。在Web Service技术出现之前，SOA是用CORBA或DCOM技术来实现，由于CORBA和DCOM存在着许多的不足，所以SOA的思想一直没有被广泛采用。

2.3.2 SOA编程模型

SOA编程模型如图2—4所示，SOA的一个重要思想就是尽量重用现有的服务，客户程序使用了服务3提供的服务，而服务3重用了服务1和服务2提供的服务。服务的对外接口都使用了开放的标准来定义如IDL或WSDL。

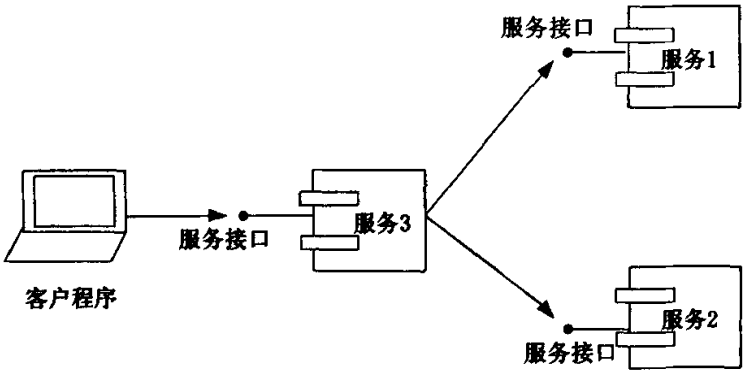

图 2-4 SOA 编程模型

1．SOA 的组成元素

SOA 的主要组件包括服务、动态发现和消息。

（1）服务是能够通过网络访问的可调用例程。服务公开了一个接口契约，它定义了服务的行为以及接受和返回的消息。术语“服务”常与术语“提供者”互换使用，后者专门用于表示提供服务的实体。

（2）接口通常在公共注册中心或者目录中发布，并在那里按照所提供的不同服务进行分类，就像电话簿黄页中列出的企业和电话号码一样。客户(服务消费者)能够根据不同的分类特征通过动态查询服务来查找特定的服务。这个过程被称为服务的动态发现。

（3）消息是客户消费使用服务的中介。因为接口契约是独立于平台和语言的，消息通常用符合 XML 模式的 XML 文档来构造。

2．SOA 的角色

SOA 有三个角色，服务使用者通过查询服务注册中心来查找需要的服务。如果服务存在，注册中心就给使用者提供服务接口的描述文档和服务的端点地址。

（1）服务消费者(Service Consumer)

服务消费者是一个应用程序、一个软件模块或 SOA 系统中的一个服务。它发起对注册中心中的服务查询，通过传输绑定服务，并且执行服务功能。服务消费者根据接口描述文档来使用服务。

（2）服务提供者(Service Provider)

服务提供者是实现服务接口的一个软件实体并可通过网络寻址来查找该实体，它接受和执行来自使用者的请求。它将自己的服务和接口契约发布到服务注册中心，以便服务使用者可以发现和访问该服务。

（3）服务注册中心(Service Registry)

服务注册中心是服务发现的支持者。它包含一个可用服务的存储库，并允许感兴趣的服务消费者查找服务并提供接口。

3. SOA 的特点

SOA 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。这种模型具有下面几个特征：

（1）松散耦合

服务请求者到服务提供者的绑定与服务之间是松耦合的。松散耦合旨在将服务使用者和服务提供者在服务实现和客户如何使用服务方面隔离开来。服务接口作为与服务实现分离的实体而存在，服务请求者不知道提供者实现的技术细节，比如程序设计语言、部署平台等。服务请求者往往通过消息调用操作——请求消息和响应而不是通过使用 API 和文件格式。服务实现的修改完全不会影响到服务的使用者。

（2）粗粒度服务

服务粒度(service granularity)指的是服务所公开功能的范围，一般分为，细粒度(fine-grained service)和粗粒度(coarse-grained service)。其中，细粒度服务是那些能够提供少量商业流程可用性的服务。粗粒度服务是那些能够提供高层商业逻辑的可用性服务。粗粒度服务可以灵活组合稳定性强、重用性高的细粒度服务，而快速形成新的业务逻辑。面向服务的体系结构(SOA)不要求使用粗粒度接口，但是推荐使用它们作为外部集成的最佳实践。服务编排可以用来创建运行由细粒度操作组成的业务流程的粗粒度接口。

（3）标准化的接口

服务描述的重点在于与几部分交互所用的操作：服务、调用操作的消息、构造这种消息的细节和关于向何处发送用于构造这种消息的处理细节的消息的信息。SOA 通过服务接口的标准化描述，从而使得该服务可以提供给在任何异构平台和任何用户接口使用。该接口隐藏了实现服务的细节，允许独立于实现服务基于的硬件或软件平台和编写服务所用的编程语言使用服务。

（4）无状态服务

服务应该是独立的、自包含的请求，在实现时它不需要从一个请求到另一个请求的信息或状态。服务不应该依赖于其他服务的上下文和状态。当需要依赖时，它们最好定义成通用业务流程、函数和数据模型，而不是实现构件(比如会话密钥)。

2.4 基于 SOA 的开发

2.4.1 SOA 设计的原则

SOA 建模是一个有挑战性的工作，因为服务的概念更进一步的提高了抽象的水平，它的最终目的是逾越业务和软件之间的鸿沟。因此，一定要将面向对象和面向组件设计的最好的实践经验，以及工作流和商业过程建模技术综合在一起，从而达到 SOA 建模的目的。

具体来说，为了能够保证 SOA 的优势能够体现的淋漓尽致，在设计企业系统架构时，应遵循一定的原则。下面从 SOA 整体和服务两个方面说明 SOA 的设计原则。

从架构整体角度来说：

（1）必须完全理解企业的业务需求，以及企业未来的发展趋势。

（2）必须清晰所有服务之间的关系，服务之间要保证良好的松耦合关系，或者将依赖性减至最少，从而使系统达到较高的灵活性和敏捷性。

（3）在配置和使用这些服务的系统中，服务应该被相互独立地配置和更新。

从服务本身角度来说：

（1）服务与外界的交互完全通过公共接口来完成的，服务内部的实现对于外部来说是完全透明的。

（2）在满足需求的前提下，服务提供的公共接口要尽可能的少，这样使用和维护

都会相对容易。

（3）服务的接口是静态的，应该被设计得不用破坏本身便可进一步扩展。

（4）设计中应该在不损失或损害相关性、一致性和完整性的情况下，尽可能地进行粗粒度建模。通过一组有效设计和组合的粗粒度服务，业务专家能够有效地组合出新的业务流程和应用程序。

2.4.2 SOA 开发

SOA 蕴含的内容不只是技术，它还是业务与技术间的一座桥梁。服务建模(Services Modeling)是为企业成功实施灵活的 SOA 架构的一种软件建模方法，图 2—5 展示了实现 SOA 时可能经过的步骤。

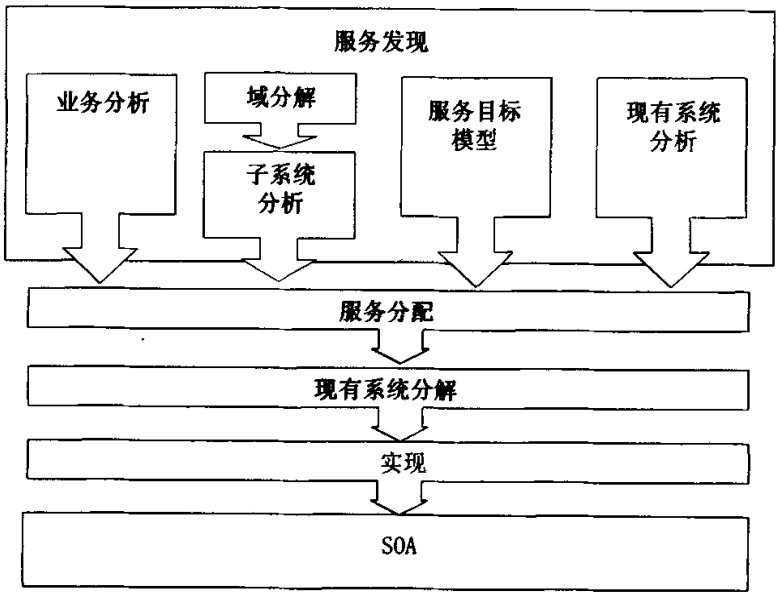

图 2-5 SOA 实现步骤

1．服务发现

服务发现(Service Discovery)是寻找和标识企业业务中的潜在服务的过程。服务是构成 SOA 的基本元素，但服务发现却不是一件容易的事情。服务发现是决定能否成功构建 SOA 的关键因素。服务发现的方法大致有以下几种：

➢ 业务分析

通过项目相关人员的会谈进行 BPM(Business Process Management，业务流程管理)，和直接需求分析是一个容易理解且非常合适的标识候选服务的方法。业务分析的主要目标是要找出企业中的业务流程，要明确和详细地描述每个业务过程。

➢ 域分解

域分解是自顶向下的服务建模过程。域是指业务领域，业务领域包括一系列的功能区域，域分解就是把业务领域分解成多个功能区域和一些高层次的业务用例。这些用例

是业务层面上的候选服务。

➢ 子系统分析

功能区域是业务流程中的一部分，而功能区域可能包含一个或多个子系统。功能区域是在业务层面上的概念，而子系统是技术层面上的概念。域分解后就能明确地了解每一个功能区域，而子系统能精确描述一个功能区域，子系统分析能从业务用例中提炼出系统层上的用例，这些用例也是很好的在技术层面上的候选服务。

➢ 现有系统分析

现有系统分析是自底向上的服务建模方法，它为支持业务过程的底层服务功能性实现提供低成本的解决方案。在这个过程中，分析和利用了来自遗留系统和打包应用程序的 API、事务和模块。在有些情况下，为了支持以服务为中心的功能模块化，需要将现有的资产和遗留系统分包成组件[13]。

➢ 服务目标模型

服务目标模型是用来验证自顶向下或自底向上的服务发现手段中捕捉到的候选服务，同时也可以发现未捕捉到的潜在服务。服务目标模型为每个候选服务定下它应该达到的目标，还可以为大目标细分为多个子目标。要达到目标就要实现目标对应的服务。

2. 服务粒度

选择正确服务的抽象级别是服务建模的一个关键问题。服务粒度太粗可能会降低它的重用能力，太细则变成紧密耦合。理论上，在不损失或损害相关性、一致性和完整性的情况下尽可能地进行粗粒度建模。在建模过程中可能需要将服务逐渐细化，但应该保证每个服务都具有一定的粗粒度特性。由于 SOA 并不等同于 Web Service 和 SOAP，因此可以使用不同的协议绑定来访问抽象级别不同的服务。

3. 系统分解

实现 SOA 很少是以全新的项目开始的，创建 SOA 解决方案几乎总需要涉及集成现有的遗留系统，集成的方法是将它们分解成服务、操作、业务流程和业务规则。从现有的系统分析就是根据现有系统提供的功能，把它分解成多个服务形式的组件。系统分解常用的方法就是把系统分解成多个 Web 服务，这样可以屏蔽系统的实现技术，以标准方式集成在一起。

2.4.3 SOA 的优点

面向服务的体系结构可以基于现有的系统投资来发展，而不需要彻底重新创建系统。如果组织将开发力量集中在创建服务、利用现有的技术、结合基于组件的方法来开发软件上，将获得如下几方面好处：

（1）利用现有资产—这是首要的需求。通过使用适当的 SOA 框架并使其可用于整个企业，可以将业务服务构造成现有组件的集合。使用这种新的服务只需要知道它的接口和名称。服务的内部细节以及在组成服务的组件之间传送的数据的复杂性都对外界隐藏了。这种组件的匿名性使组织能够利用现有的投资，从而可以通过合并构建在不同的

机器上、运行在不同的操作系统中、用不同的编程语言开发的组件来创建服务。遗留系统可以通过 Web 服务接口来封装和访问。

（2）商品化基础架构—在所有不同的企业应用程序之间，基础架构的开发和部署将变得更加一致。现有的组件、新开发的组件和从厂商购买的组件可以合并在一个定义良好的 SOA 框架内。这样的组件集合将被作为服务部署在现有的基础构架中，从而使得可以更多地将基础架构作为一种商品化元素来加以考虑。

（3）减少成本—随着业务需求的发展和新的需求的引入，通过采用 SOA 框架和服务库，为现有的和新的应用程序增强和创建新的服务的成本大大地减少了。

（4）持续改进业务过程—SOA 允许清晰地表示流程流，这些流程流通过在特定业务服务中使用的组件的顺序来标识。这给商业用户提供了监视业务操作的理想环境。业务建模反映在业务服务中。流程操纵是以一定的模式重组部件(构成业务服务的组件)来实现的。这将进一步允许更改流程流，而同时监视产生的结果，因此促进了持续改进。

（5）以流程为中心的体系结构—现有的体系结构模型和实践往往是以程序为中心的。应用程序是为了程序员的便利而开发的。通常，流程信息在组件之间传播。应用程序很像一个黑匣子，没有粒度可用于外部。重用需要复制代码、合并共享库或继承对象。在以流程为中心的体系结构中，应用程序是为过程开发的。流程可以分解成一系列的步骤，每一个步骤表示一个业务服务。实际上，每个过程服务或组件功能都相当于一个子应用程序。将这些子应用程序链接在一起可以创建能够满足业务需求的流程流。

2.5 本章小结

本章首先对传统的企业应用集成 EAI 的方法及其架构进行了分类介绍，在指出传统 EAI 缺点的基础上，对 SOA 的概念、特点、编程模型、以及 SOA 架构的设计原则，分析方法，实现步骤等一系列理论进行了论述。

第三章 面向服务的应用集成技术

3.1 Web Service 软件架构模型

Web Service(Web 服务架构)是由 W3C(World Wide Web Consortium，万维网联盟)制定的一套开放的标准的技术规范，W3C 对 Web Service 的定义如下："一个 Web 服务是为提供机器与机器间跨越网络的互操作而设计的一套软件系统，它有一个用机器能处理的格式(特别是 WSDL)来定义的接口。其它系统使用 SOAP 消息且以一种被这个 Web 服务预先指定的方式来与这个 Web 服务交互，一般情况下，这些消息使用 HTTP 协议进行传输，在传输前这些消息使用基于 Web 的相关标准来进行 XML 的序列化。" Web Service 的目的是让不同的软件应用程序能相互操作，而不管这些程序是用什么编程语言实现、运行在什么样的操作平台或架构技术上[14]。强大的互操作性和可扩展性是 Web Service 的表现特征。

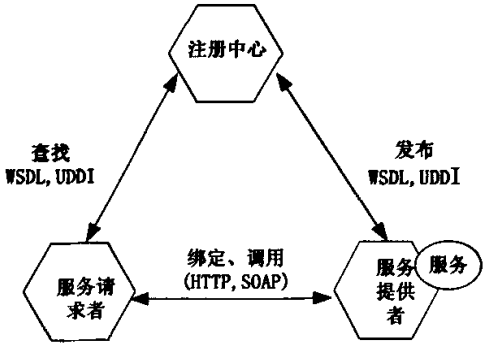

图 3-1 Web Service 结构图

如图 3－1 所示，Web 服务提供者提供可通过网络访问的软件模块(Web Service 中的一个 Web 服务实现)，Web 服务提供者定义了 Web 服务的描述，并把它发布到 Web 服务注册中心或一个 Web 服务的请求者中。Web 服务请求者使用查找操作从本地或 Web 服务注册中心搜索 Web 服务描述，然后使用 Web 服务描述与 Web 服务提供者进行绑定并调用相应的 Web 服务，实现交互。

3.2 Web Service 关键技术

在 Web Service 协议栈中，涉及了数据、消息、服务和注册库的描述定义，这些要用到具体的标准技术，具体包括 XML、SOAP、WSDL、UDDI 等。其中 XML 用来描述不同层次的数据，它使得不同平台、不同环境中的数据和消息得以互通。SOAP 协议用来交换 XML 消息；WSDL 用来统一描述服务；UDDI 提供了统一的框架和编程接口，与 WSDL, SOAP 相互结合来管理 Web Service，提供服务发布和服务发现能力。下面分别对这四种技术作简要的介绍。

1．可扩展标记语言(eXtensible Markup Language, XML)

XML 是一种定义数据的简单而标准的方法，可以看作是"Web 上的 ASCII 码"。

使用 XML 就好像可以使用自己喜欢的编程语言来创建任何一种数据结构，然后和他人的计算平台上使用的其他语言来共享数据。XML 的数据描述机制保证了它将成为 Internet 上共享信息的强大途径，因为：

（1）XML 具有灵活的可扩展性：它提供了通用的机制，可以让用户灵活的自定义各种数据，并且命名空间 (Namespace)和 XML Schema 等技术保证了 XML 的有效定义和管理；

（2） XML 具有自描述性：XML 具有的结构化特点使得数据上下文之间有结构和数据的关联，这样使得它不仅能够为人所阅读，更重要的是它提供的编程接口可以让机器(程序)方便地解析和处理 XML 数据；

（3）XML 具有开放性：XML 提供了中立的纯粹数据定义方式，它和具体的编程语言以及操作平台无关，通过和传统网络传输协议的结合，XML 数据非常适合在 Internet 中传输和处理；

XML 把数据和形式进行了分离：这是 XML 和 HTML 的根本不同之处。XML 只关心数据的内容，而其表现形式则交给程序或者 XML 样式语言来处理。

2. 简单对象访问协议(Simple Object Access Protocol, SOAP)

简单对象访问协议(SOAP)最先由 Microsoft 公司提交给 W3C 组织，并于 2000 年 4 月通过 1.0 版本。它是用于交换 XML 编码信息的轻量级协议，主要包括以下四个方面的内容：

（1）SOAP 信封(envelope)：定义了整体 SOAP 消息的表示框架，可用于表示消息中的内容是什么、发送方是谁、谁应当接受并处理它、这些操作是否必需。

（2）SOAP 编码规则：定义了数据的编码机制，通过它来定义应用程序中需要使用的数据类型，并可用来交换由这些应用程序定义的数据类型所衍生的实例。

（3）SOAP 远程过程调用(RPC)：定义了一个用于表示远程调用和响应的约定，例如，如何使用 HTTP 或 SMTP 协议与 SOAP 绑定，如何传输过程调用，在具体传输协议的哪个部分传输过程响应等。

（4）SOAP 绑定：定义了一个使用底层传输协议来完成在结点间交换 SOAP 信封的约定。SOAP 完全继承了 XML 的开放性和描述可扩展性，使用基于 TCP/IP 的应用层协议 HTTP, SMTP, FTP 等，可以与现有通信技术最大程度地兼容，SOAP 为使用 XML 在松散、分布的环境中对等的交换信息提供了一个简单的机制，它本身并不定义任何应用语义，如编程模型或特定语义实现，只定义一种简单的机制，通过一个模块化的包装模型和对模块中特定格式编码数据的重编码机制来表示应用语义。

绑定于 HTTP 之上的 SOAP 协议，可以跨语言、跨操作系统进行远程过程调用，实现了编程语言和系统平台的无关性。而以前的调用方式则和复杂的分布式对象标准或是中间件有密切的关系，从长期的眼光看，这些都不是高效的解决方案。XML 和 SOAP 这样的跨语言、跨平台的解决方案大大简化了不同企业系统之间的交互问题。

3．Web 服务描述语言(Web Service Description Language, WSDL)

WSDL 由 Microsoft、IBM、Ariba 三家公司在 2000 年 9 月推出，是由 Microsoft 公司的 SDL(Services Description Language)、IBM 公司 NASSL(Network-AccessibleServices Specification Language)合并后被 W3C 接纳所形成的标准。它是用来描述 Web Service 的 XML 语法，它用于定义 Web Service 以及调用方式。WSDL 文档可用于动态发布 Web Service、查找已发布的 Web Service 并绑定 Web Service。

在 WSDL 中，端点和消息的抽象定义与具体的网络分布和数据格式绑定是相互分离的，这样就可以抽象定义消息和端口类型，实现它们的重用。一个完整的 WSDL 服务描述是由一个服务接口和一个服务实现文档组成的。由于服务接口表示服务的可重用定义，它在 UDDI 注册中心被称为 tModel 发布。服务实现描述服务的实例，每个实例都是使用一个 WSDL 服务元素定义的。服务实现文档中的每个服务元素都被用于发布 UDDI businesssService。

4．统一描述、发现和集成规范(Universal Description Discovery and Integration, UDDI)

UDDI 规范由 Microsoft, IBM, Ariba 三家公司在 2000 年 7 月提出，它是在原有 Microsoft 提出的 DISCO(Discovery of Web Service)和 IBM 的 ADS(Advertisement andDiscovery of Services)的基础上发展而来的。UDDI 提供了在 Web 上描述并发现商业服务的框架，是面向 Web 服务的信息注册中心的实现标准和规范。UDDI 通过服务注册，以及使用 SOAP 访问这些注册信息的约定来实现上述目标。

UDDI 的核心组件是 UDDI 商业注册，它使用 XML 文档来描述企业及其提供的 Web 服务。从概念上说，UDDI 商业注册信息内容分为白页信息、黄页信息、绿页信息。白页中存放企业的地址、联系方式、企业身份识别等企业信息；黄页中存放基于标准分类的行业类别信息；绿页中存放 Web 服务的技术信息。多个合作站点之间可以无缝地共享注册信息。

UDDI 规范描述了 Web 服务的概念，同时也定义了应用程序编程接口(API)，提供描述各种 Web 服务的简单框架。UDDI 程序 API 规范分为两个逻辑部分：查询 API 和发布 API。查询 API 又分为两个部分：一部分被用来构造搜索和浏览 UDDI 注册信息的程序；另一部分在 Web 服务出现错误时使用。程序员可以利用发布 API 创建各种类型的工具，以直接与 UDDI 注册中心进行交互，便于企业技术人员管理 businessEntity 或 tModel 结构的发布信息。

3.3 选择 SOA 实现技术

SOA 是一种体系结构，在具体的应用中，需通过具体的技术来实现，如图 3-2 所示，面向服务的体系结构所涉及的技术至少包括 CORBA、DCOM 和 J2EE 等。面向服务的体系结构的早期采用者还曾成功地基于消息传递系统(如 IBM WebSphere MO)创建过他们自己的面向服务企业体系结构。最近，SOA 的活动舞台已经扩展到 Web 服务[15]。

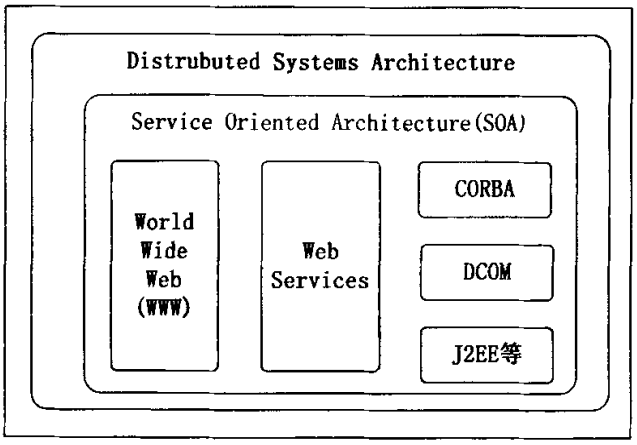

图 3-2 面向服务体系结构的不同实现

在这些实现技术中，CORBA、DCOM、J2EE 和 Web 服务都是企业应用架构设计中常要考虑选用的，接下来对面向服务体系结构的不同实现进行比较，以寻求目前条件下最为合适的面向服务实现技术。(J2EE 和 DCOM 在实现上基本一样，除了 J2EE 在跨平台能力上比 DCOM 要好之外，故下面的比较中，并不把 J2EE 作为比较对象。)

3.3.1 与现有 CORBA、DCOM 其他技术的比较

1. 基础 RPC 架构

在 DCOM 和 CORBA[16]中，客户进程与对象服务器之间的互相作用是作为面向对象的 RPC 式通信来实现的。为调用一个远程函数，客户程序要调用客户机存根。然后存根将调用参数打包成一个请求消息并调用传输协议将该消息传送到服务器。在服务器端，传输协议将消息传送给服务器存根，随后服务器存根解包请求消息并调用对象中真正的函数。在 DCOM 中，客户机存根被称为代理(Proxy)，而服务器存根被称为存根(Stub)。相反，CORBA 中的客户机存根称为存根(Stub)，而服务器存根称为框架（Skeleton)。有时，代理这个名称还被用来指 CORBA 中正在运行的存根实例。至于在 SOAP 和 Web 服务中，我们称客户机存根为服务代理(Service Proxy)，称服务器存根为服务实现模板(Service Implementation Template)。

DCOM 和 CORBA IIOP 有许多相似之处。这两个协议都使用端点标识符来识别服务器端中间件中的目标对象，且它们都使用方法标识符来确定待调用方法的签名。尽管 CORBA 和 DCOM 已经在各种平台上得到了实现，然而实际情况是建立在这些协议之上的任何解决方案都依赖于单一厂商的实现。因此，如果要开发一个 DCOM 应用程序，分布式应用程序中所有参与的节点都必须以 Windows 风格运行。如果要开发 CORBA 应用程序，应用程序环境中的每个节点都要运行相同的 ORB 产品。现在也有来自不同厂商的 CORBA ORB 能够相互操作[17-18]。但是那种互操作性并不能扩展到像安全与事务管理那样的更高级别的服务中去。

这两种协议都依赖于严格管理的环境。要找到能成功地在外部调用 DCOM 或 IIOP

的任意两台计算机的几率比较小。此外，程序员们必须处理数据排列和数据类型所需的协议唯一的消息格式规则。DCOM 和 CORBA 都是服务器对服务器通信的合适的协议。然而，它们在客户机对服务器通信方面都存在严重的缺陷，特别是当客户机遍布因特网时。

Web 服务技术组件提供了 SOAP 作为映射应用程序对象到网络协议的开放标准 ORPC。尽管 SOAP 不受特定的传输协议的约束，HTTP 还是成为了早先在 SOAP 采纳者中最受欢迎的协议。使用 HTTP 时，SOAP 信封使用 XML 作为请求和响应参数的编码方案。SOAP 消息实质上是一个遵循 SOAP 编码规则的 HTTP 请求和响应。SOAP 端点就是一个基于 HTTP 的、识别方法调用目标的 URL。与 CORBA 一样，SOAP 并不要求一个特定对象被连接到给定的端点上。相反，需要由实现者来决定如何将对象端点标识符映射到服务器端的对象上。在 SOAP 中检查方法名称的名称空间 URI 与在 DCOM 或 CORBA 中检查方法名称的接口 ID 在功能上是相同的。

Web 服务技术提供了一个全新的编程模型来利用开放因特网标准建立分布式应用程序。这一全新的分布式计算解决方案采用特定因特网技术的开放性解决了许多 CORBA 和 DCOM 的互操作性问题。特别是，Web 服务：

- 使用 HTTP 来实现防火墙友好和不确定有效负载；
- 将 XML 作为一个编码模式使用，它能比 DR 和 CDR 更为广泛地被采用；
- 提供关于 HTTP/SOAP 服务器环境的免费的经济价值建议，或提供关于 ORB 框架的收费的经济价值建议；
- 使用广泛深入的 URL 因特网概念来解决对象识别问题；
- 提供的不只是互操作性的承诺。厂商们正积极工作以证明它们的 SOAP 实现确有互操作性。

从以上比较，可以看出，CORBA/DCOM 都是如何在技术上实现互动。而 Web 服务不仅实现技术上的互动，同时，考虑如何去掉技术相关性，从业务的层次如何降低“耦合”度。Web 服务中部分标准已经比较成熟，如用于通信的 SOAP（XML 消息传递），建立请求者与响应者关系的 WSDL（Web 服务描述语言），是目前发展比较快的两个标准。

根据以上分析，可以发现与其他任何 SOA 实现技术相比，Web 服务更有可能成为提供有效的、可靠的和可打展的机器到机器交互的标准。

3.3.2 Web Service 的优点

1．采用更加中立的技术标准

Web Service 是以 XML 为基础的独立于平台与编程语言分布式软件通信技术。用 Web Service 实现 SOA 将使 SOA 的实现独立于具体的厂商。相比之下 CORBA、DCOM 和 RMI 就不能做到完全中立。DCOM 明显地受 Microsoft 公司的约束；RMI 只能用于 Java 应用程序间的通信；尽管 CORBA 有一个通过 IDL 定义的平台无关的服务约定和

一个支持不同语言和平台的实现，但是它需要 IIOP 的特定通信协议和标准化的有限格式，且 ORB 的实现中不同厂商间区别很大[19]。

表 3-1 SOA 的实现技术比较

	CORBA	DCOM	RMI	Web Service
倡导者	OMG	Microsoft	JCP	W3C
操作平台	Windows,Unix, Linux,Solaris	Windows	Windows,Unix, Linux,Solaris	Windows,Unix, Linux,Solaris
编程语言	C/C++,Java 等	C/C++,.net 等	Java	Java,C/C++,.net 等
传输协议	IIOP(不同厂商实现有差异)	ORPC	JRMP	HTTP,FTP,SMTP 等
接口定义	IDL	MS-IDL	Java interface	WSDL
通讯消息	远程对象	远程对象	远程对象	SOAP,XML 消息

2. 自描述性[20-21]：Web Service 互操作性的目的是提供从一个软件应用程序到另外一个软件应用程序无缝的、自动的连接。SOAP，WSDL 和 UDDI 协议定义了一种自描述的方式，使计算机能够在程序级上发现并调用 Web Service，从而实现这种无缝、自动的连接。

3. 松散耦合：以前的分布式的对象模型，如 MICROSOFT 的 COM+，OMG 的 CORBA 和 SUN 的 RMI，虽然也是松散耦合的，但是这些系统有一个共同的缺陷，那就是他们无法扩展到 Internet 上，因为他们要求一个同类基本结构，这样的系统往往比较脆弱：如果一端的执行机制发生变化，那么另一端便会崩溃。例如，如果服务器应用程序的接口发生更改，那么客户便会崩溃。Web Services 技术使得分布式对象之间的耦合更加松散。当一个 Web Service 的实现发生改变的时候，只要它的调用接口不变，调用者是不会感到这一变化的。Web Service 实现的任何改变对他们来说都是透明的。

4. 异步消息通讯

需要通过同一通信通道对请求做出立即响应的服务消费者(同步)，比那些可以等待并且可以从另一个通信通道接收响应的消费者(异步)更加紧密地与该服务绑定在一起，也就是说，同步服务比异步服务更加紧密地耦合。CORBA、DCOM 和 RMI 都使用了同步的信息传输协议，Web Service 通过使用 HTTP、SMTP 可以达到异步的效果。Web Service 客户端还可通过发送一回调接口给服务提供者，服务提供者可以在任何时候通过回调将服务处理结果返回给客户端，而客户端在发送完请求后无须等待返回。

5. 使用标准协议规范：作为 Web Services，其所有公共的协约完全需要使用开放的标准协议进行描述、传输和交换。这些标准协议具有完全免费的规范，以便出任意方进行实现。

如前所述，SOA 是理论概念模型，需要具体的实现技术。SOA 本身也并没有限定

实现它的技术。前面讲到的 CORBA 和 DCOM 也可以实现 SOA，但由于 CORBA 和 DCOM 本身存在着不足，所以用 CORBA 或 DCOM 实现的 SOA 并没有得到十分广泛的应用。到目前为止 Web Service 技术是实现 SOA 的最佳选择。

XML 独立于操作平台和编程语言，有强大的自描述能力，使它成为构建松耦合程序的标准。Web Service 技术是基于 XML 的异构环境下的软件通讯技术，Web Service 加上 SOA 思想将是下一代企业应用系统的开发和构建方法。用 Web Service 构造服务组件，用 WSDL 描述服务接口，用 SOAP 作用服务间的消息协议，把使用广泛的 HTTP 作为传输协议，这些都将成为实现 SOA 的标准技术。

3.4 其他相关技术

1．BPEL 实现服务灵活组装

WS-BPEL 是基于 XML 定义的流程描述语言，它位于几个 XML 规范之上:WSDL 1.1, XML Schema 1.0 和 XPathl.0。其中 W SDL 消息和 XML Schema 类型定义提供了 BPEL 流程所用的数据模型；XPath 为数据处理提供支持；所有的外部资源和伙伴被表示成 WSDL 服务。WS-BPEL 描述的业务流程指定了一组 Web 服务操作的可能执行顺序及其相互依赖关系、提供 Web 服务的合作伙伴及其在业务流程中扮演的角色、服务间共享的数据、故障及补偿处理等等一列问题。为了实现这此功能，WS-BPEL 引入了变量合作伙伴链接、活动、相关性、作用域等关键元素。WS-BPEL 为 Web 服务在业务流程管理中的应用提供了基础条件，既解决了 Web 服务的集成问题，同时也促进了业务流程管理自身的发展。虽然从理论上来说，WS-BPEL 的使用能够解决业务流程管理的集成与异构问题，而且它也确实能够解决一些实际问题，但是，由于 XML 自身的局限 (如对于非结构化信息的描述)，以及在人员参与流程交互方面的不足等一系列的问题，WS-BPEL 还需要在很多方面进行改进 [22]。

2．企业服务总线

企业服务总线(ESB，Enterprise Service Bus)是在 SOA 的基础上提出的构建基于面向服务体系结构(SOA)解决方案时企业所使用基础架构的关键部分。通过利用出现的通信、连接、转换和安全的标准，ESB 成为企业与合作伙伴之间强大、可以承受的、基于标准的信息骨架。ESB 使得业务过程的路径更加平滑，而且减少了集成对业务过程各个步骤提供支持的组件所需的时间、尝试和成本。这种方法的一个强大的优势是它允许企业内部的开发团队构建新的应用。这种应用是能够被集成的并且根据要求轻松的被组装进 ESB 中。还有其它的优点包括保护昂贵的技术投资，减少市场反应时间和提高现有软件资产的复用率。从功能上看，ESB 提供了事件驱动和文档导向的处理模式，以及分布式的运行管理机制，它支持基于内容的路由和过滤，具备了复杂数据的传输能力，并可以提供一系列的标准接口[23]。

3.5 本章小结

本章首先介绍了 Web Service 的概念、特点、架构及协议栈，接着着重讨论了 Web

Service 的核心技术：XML、SOAP、WSDL、UDDI，并对 SOA 中其他实现方法如 CORBA，DCOM 等进行了分析比较，指出了 Web Service 做为 SOA 架构主流实现技术的各种优点。最后，介绍了 BPEL 、ESB 其他 SOA 相关技术。

第四章 基于SOA的企业应用集成框架

一个理想的EAI解决方案应该要满足以下的一些条件[24]：

- 基于工业标准：尽量减少在异构环境之间对私有适配器和连接器的需要。
- 松散的耦合，即请求不必针对特定应用的API。
- 异步执行方式。使得在等待第一个应用的响应时可以执行第二个应用。
- 可靠性。保证消息被投递一次且仅仅一次。
- 安全性。必须支持鉴别、授权标准以保护被交换信息的完整性和机密性。

从前面所介绍的一些传统的EAI解决方案中我们可以看出，传统的EAI解决方案是私有的和复杂的，因而不能实现必要的灵活性和适应性。而Web服务技术出于使用标准的Web协议(HTTP, SMTP等)和一系列标准协议(XML, SOAP,WSDL，UDDI等)而满足了上述所有条件，为EAI提供了一种崭新的方法。

本章提出了一种基于Web服务的企业应用集成框架(Web Service Architecture-Based Integration Framework，简称WSA)，它和传统的EAI解决方案最大的不同在于跨企业应用的整合，同时也考虑不同平台开发Web服务在实际应用中的互操作问题，通过把各种已有系统转化为Web服务从而在根本上使得企业应用集成是完全基于Web服务的。因而WSA-EAI使得各种应用系统的整合更方便、易用、便宜以及更具动态可扩展性。

4.1 框架的功能结构

基于Web服务的EAI是一种面向服务层的松耦合的企业应用集成方案，可以最大限度地同时满足性能和灵活性的要求。同时Web服务是一种正在快速发展的技术方法，它的不断改进和完善必将带来企业应用集成领域格局上的变化。Web服务彻底改变了传统的EAI中点对点的集成处理方式，为EAI提供了一种新型、标准的集成策略。

面向服务的集成将传统的集成对象与开放的、高灵活性的Web服务整合在一起。面向服务的集成提供了一些抽象的接口，通过这些接口，系统可以进行交互，而不是使用底层的协议和自定义的编程接口，来规定系统如何与其它系统进行通讯。面向服务的集成使得企业能够在已有的应用中提供可重用的服务的功能。基于Web服务技术，对于企业原有应用，在不需要对原有系统进行修改、不影响原有系统功能的情况下，只要在原有系统的基础上加上一个SOAP接口，就可以将现有的、用不同技术实现的系统互联起来，提供相互的数据交流和访问操作，进而各种不同的系统可以互相协作，形成一个更为强大的大系统。

结合第二章讨论的几种传统的集成架构模型，在比较它们的优劣后本文设计了一个基于Web服务的EAI框架模型，图4-1较为详细地描述了该模型。

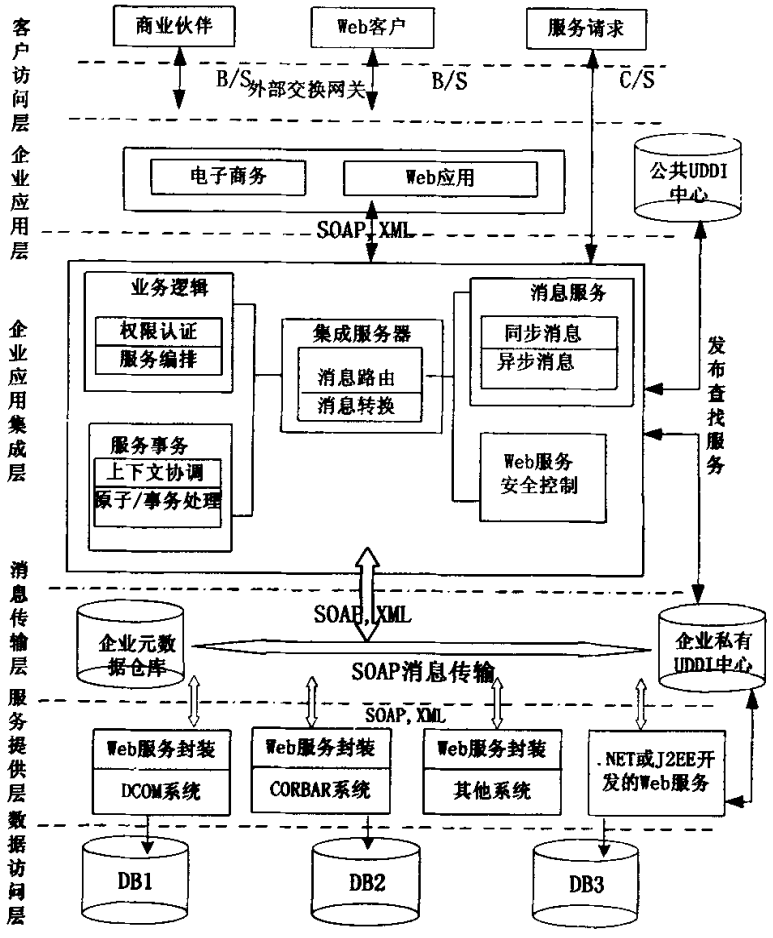

图 4-1 WSA 企业应用集成框架

如图 4-1 所示，WSA-EAI 整个系统结构以分层的方式进行构建，从上到下依次分为：客户访问层(Client Request Layer)、企业应用层(Enterprise ApplicationLayer)、企业应用集成层(Enterprise Application Integration Layer)、消息传输层(Message Transport Layer)、服务提供与封装层(Service Provide and EncapsulationLayer)。

其它的一些辅助模块包括：客户交互网关(Client Interaction Gateway)、企业防火墙(Enterprise Fire-wall)、公共 UDDI 注册中心(Public UDDI Register Center),企业私有 UDDI 注册中心(Private UDDI Register Center)及企业元数据模式仓库(Metadata Repository)。各层的主要功能描述如下：

（1） 客户访问层(Client Request Layer)

对于一个企业来说，它的客户一般分为以下几类：Web 客户、一般的服务请求者和商业合作伙伴。它们通过企业防火墙和外部交互网关后访问企业应用层的典型应用，充当了 Service Requester 的角色，它们可能会通过各种协议方式(如 HTTP, RMI-IIOP 等)来访问企业应用层，而对客户来说这些实现的细节都是透明的。

（2）企业应用层(Enterprise Application Layer)

企业应用层是各类B/S模式的企业应用，提供用户使用界面，负责与用户交互。它利用了企业集成层所提供的服务接口和一些基础设施，实现了企业各种应用的开发、和重新整合。该层包括了现有的一些典型的Web应用、电子商务应用等。

（3） 企业应用集成层(Enterprise Application Integration Layer)

企业应用集成层是 WSA-EAI 系统的核心部件，包含了多个组件：集成服务器(Integration Server)、用户身份认证(User Identity)、消息服务(Message Service)、服务事务(Service Transaction),Web服务安全控制(WSA-Security)。其中集成服务器是企业应用集成层的核心部分，也是其它几个服务功能模块的信息交换中介，同时它也负责对外提供一个统一的应用调用处理界面，它包含了SOAP消息路由和消息格式转换两个主要的功能。用户统一身份认证负责对企业的内部和外部用户进行身份认证和权限管理。消息服务提供了基于消息中间件(如JMS)的消息服务。服务事务提供了原子/业务事务，也负责协调事务的上下文。Web服务安全控制主要是基于WS-Security机制，对WS-EAI系统集成交互进行安全访问控制。

关于业务逻辑的问题，在某些系统开发实例中，采用下面的做法：在Web服务中调用业务规则，由业务规则再调用数据层组件来执行实际的数据操作；本文以为按照SOA理论中“业务驱动服务”的规则，业务逻辑可分为与Web服务无关和与Web服务有关的两种，从理论框架的理想模型的角度来说，与Web服务无关的业务逻辑其位置宜在应用层，与Web服务有关的业务逻辑一般应在企业应用集成层，可以对下层的细粒度Web服务进行组合和编排。

（4） 消息传输层(Message Transport Layer)

消息传输层作为消息总线连通了企业应用集成层和服务提供与封装层的消息交互，并为企业应用集成层进行消息格式转换功能提供了元数据映射模式。消息总线采用基于XML的SOAP信息格式，它采用了两种消息模式：同步通信消息和异步通信消息。同步通信消息主要适用于实时性比较高的应用，需要调用企业应用服务以后立即得到响应，这里主要用到了HTTP这种SOAP调用/绑定实现方式。SOAP消息在消息总线上既可以通过同步调用的方式传输，也可以采用异步通信的方式进行传输。

（5）服务提供与封装层(Service Provide and Encapsulation Layer)该层主要是对企业内部各种EIS系统进行Web服务封装，为WS-EAI系统提供各类Web服务接口。服务提供与封装层在Web服务的架构中起到了服务提供者的角色，为客户和企业的其它增值应用提供服务。

（6） 数据访问层

用于提供应用及服务所需的数据。

WS-EAI还包括一些其它的辅助模块：

（1）企业防火墙

处在集成框架最前部，用于过滤和阻止对企业应用系统的非法访问和黑客攻击。

（2）外部交互网关

处在企业防火墙之后，对于已经过企业防火墙的合法用户，它是企业应用的门户，是外部调用的统一入口。它接收外部对企业信息资源调用的请求，将请求按照预定义的规则分发给不同的企业应用，然后由企业应用层负责处理请求。

（3）企业元数据模式仓库

属于消息传输层。主要存放企业应用集成中所涉及到的所有数据元模型、规则等。例如 SOAP 消息 Schema、企业数据元模型 Schema 等。它是一个存储规则和模式的数据仓库。

（4）企业私有 UDDI 注册中心

企业应用以 Web 服务的方式封装后，所有关于这些 Web 服务的描述文件-WSDL 均要在这个注册中心注册。对这些服务的调用均要首先在注册中心搜索以决定调用的端口和方式。

（5）公共 UDDI 注册中心

某些企业应用需要提供给许多外部的用户进行查询、调用，这时就需要把这些企业应用进行 Web 服务的方式进行封装后，服务的描述文件 WSDL 就要到公共的 UDDI 注册中心进行注册。

WSA-EAI 框架基于 Web Service 技术，在这个框架下面，企业的各种信息系统通过 Web Service 封装器将各自的功能进行封装，这些信息系统既包括企业已有的旧应用，也包括新开发的 Web Service 应用。如果是企业原来就有的应用系统，需要首先将其封装成 Web Service 组件。如果注册中心是私有的，则集成的是企业内部的应用系统；如果注册到公有注册中心，则可以通过 Internet 集成不同企业之间的不同系统。集成引擎通过接口将各类应用封装成 Web 服务部件后发布到 UDDI 注册中心；并通过接口调用相应的应用。它是连接各类应用的桥梁，采用的是松散耦合的方式，即任何应用都可以调用对应的接口连接到系统中来，实现了“即插即用”，具有很强的灵活性。

调用 Web 服务的客户端可以是一个基于浏览器的应用程序、组件或者是另一个基于 XML 的 Web Service 等。无论是 Intranet 内部的客户端程序还是通过 Internet 穿越防火墙登录到企业的用户，通过企业内部的集成服务器，查询私有 UDDI 中相关的 Web 服务，获取内部发布的 Web 服务位置并绑定相应的 WSDL 描述文档，然后内部服务集成代理通过内部 SOAP 消息总线自动将 WSDL 描述文档装载到代理平台中，并生成相应的接口，同时服务代理利用 XML Schema 的工具快速地理解应用交互需要使用的数据结构，使用 SOAP 技术与服务资源层的应用系统进行交互，最后请求代理将最后结果送回到客户端。

这里，集成平台代理就如同 SOAP 客户端，而后台的应用系统就如同 SOAP 服务端，其交互过程如下：首先，当客户端通过 WSDL 文件得到相应的 Web 服务中提供的接口

和数据结构，并提出服务请求得到响应时，就建立了服务通道，SOAP 客户端向 SOAP 服务端发出一个用 SOAP 协议封装的 XML 格式的字符串包(服务请求)。而 SOAP 客户端并不关心方法是怎样完成的，完成的过程对 SOAP 客户端是完全透明的。SOAP 服务端通过侦听器接收到 SOAP 客户端发来的 SOAP 请求后，解读出其中的请求信息，然后从绑定文件中找到相应的信息，作进一步处理后，通过消息封装组件和消息传输组件将响应的 SOAP 包(XML 格式的字符串包)返回给 SOAP 客户端。这个请求和响应的整个通信过程都是基于内部 SOAP 消息总线交互完成的。

4.2 WSA-EAI 框架的特点

（1）实际应用的互操作性。WSA-EAI 是基于 Web 服务的，所以它采用一系列的开放标准如 UDDI, SOAP, HTTP 等，基于现存的开放标准很大程度上消除了企业应用系统间的互操作问题。不仅如此，WSA-EAI 在采用 WS-I 组织推荐的 WS-I 基本概要的基础上，同时考虑了不同厂商的 Web 服务之间的互操作问题并给出了相应的开发标准/规则。

（2）灵活性。虽然 WSA-EAI 考虑了不同的集成层次的需要，但是也可以根据实际需要进行相应的裁减，充分体现了该模型框架的灵活性。同时，由于它是建立在发布服务的应用程序和使用服务的应用程序之间的松散耦合的基础上的，这使得企业应用易于更改，可以连接异构系统和不同的操作环境。开发人员可以在实际应用整合中根据需要，有选择地挑选需要的模块从而组成企业自身所需的系统整合原型。

（3）事务处理机制与健壮性

在框架的事务处理方面可以使用 BPEL 的事务功能。Web 服务业务流程执行语言(Business Process Execution Language for Web Scrvicas,简称 BPEL4WS 或 BPEL)是一种工作流定义语言，它使企业能够描述既能使用又能提供 Web 服务（简称 WS）的复杂的业务流程。BPEL 构建在 XML 和 Web 服务的基础上，支持 Web 服务技术的协议群，包括 SOAP, WSDL, UDDI, WS-Reliable Message, WS-Addressing, WS-Coordination 和 WS-Transaction。BPEL 是早期两个工作流语言(WSFL 和 XLANG）的综合。BPEL 综合了两者的特点，为描述业务流程提供了丰富的语义词汇。两种重要的补充规范 WS-Coordination 和 WS-Transaction 用来协调和增强短期和长期运行的企业事务的可靠性。这对于分布式业务流程的成功实现是至关重要的。

BPEL 中的长期运行的事务被集中在作用域(scope)上而作用域是可嵌套的。作用域是允许活动分组的结构化的活动，并为相应的一组活动定义公共的执行上下文在作用域和它的父作用域之间有个 WS-Transaction 中的同意协议(agreement protocol)，用于确定作用域所代表的长期运行的事务的结果。目前的 BPEL 长期运行的事务假定作用域和它所嵌套的所有作用域被包含在单个流程中并由单个 BPEL 引擎来管理，但是 WS-Transaction 中的同意协议并没有作这个假定。所以，未来的 BPEL 扩展可能支持分布于流程间甚至跨 BPEL 引擎的长期运行的事务。

通过作用域机制，BPEL 允许定义可在错误情况下一起被撤销的活动集。这样的活动集是一些工作单元和些事务。撤销已完成活动的所需操作被称为补偿处理程序(compensation handler)。作用域的故障处理程序可能使用补偿处理程序来撤销这个作用域中进行的操作。在作用域中进行的活动要么已全部完成，要么全部被补偿。BPEL 提供了以特定于应用程序的方式定义故障处理和补偿的能力，补偿处理程序允许流程的创建者定义用于撤销个工作单元并把数据恢复到执行该工作之前的样子的特定操作。在这种环境中，WS-Transaction 规范被用来注册对提供的撤消通知感兴趣的参与者。

由于被调用的Web服务、流程本身或网络等故障都可造成事务的终止，所以在BPEL中补偿与故障处理机制密不可分。通过将事务封装成作用域来实现这个处理，可以将作用域看作是一个可补偿的、可恢复的工作单元的封装。BPEL 能够在活动嵌套的不同级别捕获并处理错误，可以在任何作用域中定义故障处理程序和补偿处理程序。有补偿处理程序和故障处理程序的作用域可不受约束任意深地被嵌套。

当故障在作用域中发生时，作用域中的正常处理被中断，发出的故障被传给捕获故障的处理程序。每个处理程序包含一个在作用域需要被补偿时运行的活动。在这样一个处理程序中的活动必须看到容器数据与作用域完成时是一样的。由于各活动共享容器和由 while 活动引起的循环，所以，支持补偿功能的正在完成中的作用域为数据保存一个快照，以备处理程序将来叫能要用到。一旦作用域成功地完成，它的补偿处理程序就已准备好按相反的顺序被执行。

BPEL 中定义了两种补偿：显式补偿或隐式补偿。显式补偿的调用方法是使用 compensate 活动。例如：<compensate scope="Sa"/>。该活动只能在父作用域的 fanlt Handler、和 compcnsationH andler 活动中调用。在没有显式处理程序的情况下， BPEL 提供缺省的补偿处理程序和故障处理程序。这些隐式处理程序的行为是按完成相应作用域的相反顺序自动运行可用的补偿处理程序。

有了 BPEL 提供的事务处理机制，集成框架平台在局部系统出错的时候能更好的恢复，使系统获得更好的健壮性。

（4）可扩展性。传统的 EAI 解决方案一般都基于某一种特定的协议、开发平台甚至特定的开发语言，因而具有较弱的扩展性。而 WSA-EAI 通过底层的 Web 服务允许企业把大的应用划分为小的独立的逻辑实体并且包装它们，以提供多个集成的连接点，因而具有良好的可扩展性。

（5）数据完整性。外部应用通过 WSA-EAI 集成平台调用企业应用系统，WS-EAI 屏蔽了企业应用系统执行时的逻辑和细节，不直接操作最底层的企业数据，保证了数据的完整性和一致性。

4.3 Web 服务互操作性研究

在图 4-1 所提出的 WSA-EAI 框架中，应用集成层存在着各种各样不同的企业应用系统，架构也各不相同，有基于 DCOM 架构的，也有基于 CORBA 或者其它架构的，

要实现这些不同架构系统的集成，需要对这些系统用 Web 服务进行封装，而进行 Web 服务的开发主要有 J2EE 和微软的.Net 两大平台，由于平台的架构、开发环境和开发语言都不相同，在这两种平台开发的 Web 服务进行相互调用时就会产生一些问题，本节主要对这些互操作性问题进行一些讨论和分析。

4.3.1 不同平台下的 Web Service 互操作问题

1. J2EE 调用 VS.NET 下开发 Web 服务

以 J2EE 选取的 BEA 的 WebLogic 集成平台为例。其步骤如下：

（1）在.NET 中打开 Web 服务，在菜单中选择 Debug/Start，在浏览器中打开该 Web 服务。然后右键单击保存“Service Description”链接，把该文件名的后缀设为.WSDL；

（2）在 WebLogic Workshop 中创建一个服务控件，该控件对应于第一步中的 WSDL 文件，然后就可以在 Workshop 中把该 Web 服务当作普通控件使用了。

然而，当按照 1 述步骤后，在 Workshop 中调用时封装后控件的返回告警信息方法时，应用服务器返回了调用错误信息，提示未定义 Return DataSet 类型，这说明一该.NET Web 服务中的 DataSet 类型不能在 Workshop 中被应用服务器正确解析。

2. SoaptoolKit 调用 VS.NET 互操作

用 SOAP Toolkit 的 API 来当做 Web Service 的客户端，调用出. NET 开发的服务器端，这个意义就是，同样都是微软开发 Web Service 的工具，理论上应该是最兼容的。

SOAP Toolkit 或. NET 的 Web Service 都是以 SOAP 规格为基础，来开发 Web Service 的应用程序的。而且以 SOAP 为基础的应用程序，客户端与服务器端通过 WSDL 的服务描述的规范，只要符合 SOAP 消息协议，这两端就可以分开定义与实现。也就是说，使用 SOAPToolkit 的客户端可以调用. NET 实现的 Web Service，反之亦然。

可以把 WSDL 想像成服务合约，使用 SOAP Toolkit 开发的应用程序可以转换成. NET 应用程序，但是维持服务合约不变。例如原来为 SOAP Toolkit 的服务器端虽然已经转换成. NET 的 Web Service，但是客户端还是可以持续使用 SOAP Toolkit 的 SoapClient 对象去访问这个服务。访问可以采用两种方式，使用 SOAPToolkit 高层 API(High Level API)和低层 API(Low Level API),，本节以使用 SOAPToolkit 高层 API 为例予以说明。

用高层 API 的 SoapClient 对象当做客户端，调用 Web Service 的步骤跟其他类型的客户端并无不同，惟一要注意的是，WSDL 的地址要跟引入 Web 引用时一样，使用"http://www.test.com/service 1.asmx?wsdl"的方式来取得 WSDL 的内容。

WSH 的程序代码如下：

```
Option Explicit
        COnst WSDL_URL = "http://localhost/AspNet/Service1.asmx?
Dim Hello WorldObj
        Set HelloWorldObj=CreateObject("MSSOAP. SoapClient")
```

```
HelloWorldObj.mssoapinit WSDL_URL
WScript.Echo HelloWorldObj.HelloWorld( )
Set HelloWorldObj=Nothing
```

执行此段 WSH 程序代码，原来看似简单的 Web Service，却还是执行失败。错误的消息大意说某个服务函数不能被初始化。因为参数中包含复杂数据类型，却不能被展开。其实不能被展开的原因很简单，因为 HelloWorld 函数根本没有输入参数，从 WSDL 的内容中很容易看出来。

4.3.2 进一步研究 Web Service 互操作问题

我们先来看一项具体的调查数据。一项来自 Jupiter Research 的最新调查研究表明：互操作性是 65%的 IT 决策者们在构建内部应用开发平台时考虑最多的问题[25]。为降低成本有越来越多的企业采用 Web 服务。互操作性的重要性甚至超过了安全性问题和如何支持行业标准的问题，因为考虑这两个问题的人数分别为 48%和 44%。Web 服务互操作性问题现在已成立专门的组织 WS-I 来制定一系列的标准规范来保证不同开发厂商之间 Web 服务的互操作性，然而由于历史原因在实际应用过程中，仍然存在一些互操作问题，本节将重点对实际互操作影响比较大的几个问题进行阐释并提出相应的对策。

通过认真的研究、分析后，发现跨平台 Web 服务集成所面临的几个常见的互操作问题的根源在于 J2EE 技术和.NET 之间的交互类型、基本数据类型和结构以及命名空间问题。

1．交互类型不同

WS-I 基本规范提出采用文本 XML 实现互操作。它禁止对 soap：Envelope 或派生的 soap：Body 元素使用 soap:encodingStyle 属性。因此 RPC/encoded 和 Document/literal 是 WS-I 标准支持的两种格式，但是有许多 Web 服务工具不支持 RPC/encoded，所以 Document/literal 成为唯一的实际互操作性标准。

然而，采用 RPC/encoded 方式来处理在 XML 脚本或 WSDL 出现之前就已经存在很久，同时一些编码规则仍旧不能够采用 XSD 来表述。尽管 SOAP 编码被认为 Web 服务互操作性问题的主要根源，以及 ASP.NET WebMethod 基础设施缺省采用 Document/literal，但是现在大部分 J2EE Web 服务仍然缺省采用 RPC/encoded 方式。

下面举一个同班学生之间的关系为同班同学这个实例来说明由交互方式不同带来的互操作性问题。先定义一个 Student 类：

```
public class Student implements java.io.Serializable{
private Student classmate;
private String name;
public Student getClassmate(){
    return this.classmate;
}
```

```
public void setClassmate(Student classmate){
    this.classmate = classmate;
}
```

考虑初始化两个 Student 实例：A 和 B，并使 A 成为 B 的同学，使 B 成为 A 的同学，这样就会出现一个循环的对象引用。

```
Public Student makeClassrnate(Student A,Student B){
        A.setClassmate(B);
        B.setClassmate(A);
        Return A;
}
```

然而利用 Document/literal 方式没有简便的方法来实现这种循环的对象引用。原因在于：Document/literal 方法基于 XML 脚本定义了消息类型作为固定类型。XML 脚本利用 XSD 基本类型作为叶子节点来表示自然树结构。一个循环的对象引用很难用树结构表示。在这种情况下，由于 Student 类的实例 A 和 B 之间的循环引用，A 和 B 的序列化成为递归循环。利用一般的 J2EE 开发平台如 BEA 的 Workshop，最终将看到堆栈溢出意外。虽然这并不意味着 Document/literal 方法没有办法来解决它，但是需要开发人员更加深入的工作。如果采用 SOAP 编码作为消息绑定，在循环引用方面应用 SOAP 编码规则的多引用，这个循环引用关系就可以简单地表示。通过在 J2EE 平台上运行 Web 服务来响应将张三和李四互相成为同学关系的 SOAP 请求，可以看一下序列化 XML 的载荷。下面的 SOAP 编码相应消息显示了对象的循环引用是比较容易表示的。

```
    <soapenv:Body
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<makeClassmateRcsponse xmlns="http:/instanceCycle.test">
<makeClassmateReturn href="#id0" xmlns =""/>
</makeClassmateResponse>
<multiRef   id="id0" soapenc:root="0"
soanenv:encodineStvle="http://schetnas.xmlsoao.ore/soan/encoding/"
xsi:type=ns-410023638:student " xmlns: ns-410023638 =.
"http://instanceCycle.test" xmlns="">
<name xsi:type="xsd:string">ZhangSan</name>
<classmate href ="#idl">-
</multiRef>
<multiRef   id="id1"soapenc:root="0"
soanenv:encodineStvle="htto://schetnas.xmlsoao.ore/soan/encoding/"
xsi:type="ns-410023638:Student" xmlns:ns-410023638=
```

"http://instanceCycle.tcst" xmlns=""

<name xsi:type="xsd:string"> Li S i</name>

<classmate hrei="#id0"/>

</multiRef>

/soapenv:Body>

SOAP 使用本地 ids 和 idl 的类型 ID 来对编码元素 Student A (ZhangSan)和 Student B(LiSi)定义唯一的标志符，另外一个本地 unqualified 属性 href 用来定义这两个 ID 值的引用。对象 A 和对象 B 之间的循环引用通过灵活的 SOAP 编码规则可以很方便地表示。

然而当一个.NET 客户来与这个 Web 服务交互时，.NET 客户将不能够正确地逆序列化成类似于前面的 SOAP 响应消息。.NET 客户期望的描述张三和李四是同学的响应消息如下所示：

```
<soap:Body
   soap: encodingStyle="htt p://schemas.xmlsoap.org/soap/encoding">
 <tns:makeClassmateResponse>
  makeClassmateResult href= "#idl"/>
 </tns:makeClassmateResponse>
 <types:Student id="id1"xsi:type="types: Student">
 <name xsi: :type ="xsd:string">ZhangSan</name>
 <classmate herf="#id 2" />
 </ types:Student>
 <types:Student id="id2" xsi:type="types:Student">
 <name xsi: :type ="xsd:string ">LiSi</name>
 <classmate herf="#idl"/>
 </ types:Student>
</soap:Body>-
```

多引用访问器编码是很强大的，但是它很难在 XSD 中表达，因此在不同的平台之间的实现有些细微的不同。这个例子说明了 SOAP 编码的不同。Document/litera 通过文本文档来传递信息并基于 XML 脚本校验和逆序列化对象的 XML 表示。不同于 Document/literal，RPC/encoded 模型使用 SOAP 编码规则来表述抽象 SOAP 数据模型，并依赖厂商的 SOAP 库来提供抽象数据模型的具体实现。Document/literal 是高度平台独立的，不需要中间的规范来填补空白，因此这种方式对于不同厂商的实现是开放的。尽管 SOAP 是标准，由于缺少 XML 脚本支持，SOAP 实现还是有细微不同。

2. 基本数据类型和结构不同

弱类型的集合对象、包含空元素的数组、日期、时间值和特定的本地数据类型都给互操作性带来一定的问题。

（1） 弱类型的集合对象

在.NET 中以下几种调用返回类型是可以被解析成标准的 xml 文档的：返回型是简单类型，诸如 int, string 等;返回类型是结构体；返回类型是类的对象；返回类型是数组。

返回结果最好不要是 dataset，因为微软的 dataset 是经过 xsd 规范，同时和平台有关。即在定制 Web Service 的时候必须清楚自己的 Web Service 为哪种平台调用，如果仅仅是.net 的话，用 dataset 可以[26]。如果是 J2EE 平台的话，最好不要用 dataset，可以使用相应的结构体数组来代替。

（2）当心空数组、空的日期和时间值

通过 Web 服务发送空数组可能产生问题。有些工具包将空数组识别为单个空值，而另外一些则将其表示为一组空数组元素。因此在通过 Web 服务发送对象数组时，要始终确保让数组包含有效数据。Java 可以识别空的日期和时间值，但.NET 不能。因为在 Java 中，java.util.Date 和 java.util.Calendar 被分类为引用类型。在.NETFramework 中，System.DateTim。被视为值类型。引用类型可以为空，而值类型却不能。如果要跨越 Web 服务发送空日期值，则始终以复杂类型发送值，并将复杂类型的值设置为空。这将有助于避免空的日期值被曲解(从而引发异常)。

含有空元素的数组问题的解决方法：

- 在客户端传值前进行处理，不要传入含有空元素的数组，最好对数组每个元素都赋以初值；
- 在设计 Web 服务的服务器端时，对传入的每个参数进行类型校验，对于不符合要求的参数类型或不能确定类型的参数给出提示。

3．本地数据类型和 XSD 数据类型转换问题

XML Schema 通过提供大量的类型模型来增强互操作性。因为 XMLSchema 能够识别 Web 服务所使用的特定的数据类型，所以可以构建 WSDL 消息及操作。XSD 提供了大量的类型以及简单的结构，但是每种编程语言都有一套自己的本地数据类型。本地数据类型与 XSD 数据类型之间的一对一的映射是不存在的，因此在翻译过程中可能丢失信息或者接收端不可能生成某些本地数据类型的映射。

无符号的数值类型(如 xsd:unsignedInt,xsd:unsignedLong,xsd:unsignedShort 和 xsd:unsignedByte)是典型的例子。在 NET 中，uint, ulong,ushort 和 ubyte 类型直接地映射到那些 xsd 类型中，但是 Java 语言没有无符号的数值类型。

无论数据类型是数值类型还是引用类型，信息传递的一方都可能出现问题。数值类型的对象位于栈中，但是引用类型的对象位于堆中。这意味着引用类型可能有空指针，但是数值类型不能有空值。如果 XSD 类型在一种语言中被映射成了数值类型，而在另一种语言中被映射成了引用类型，那么这可能导致问题的出现。例如 xsd:dateTim。被映射成了 System.DateTime，这是 C#中的数值类型。它也被映射成了 Java.util.Calendar，而它是 Java 中的引用类型。在 Java 中当引用类型没有引用任何对象时将其赋空值，然而如果.NET Web 服务从 Java 客户端接收到数值类型为空值的数据时，将抛出 System.FormatException 异常。

本地数据类型和 XSD 数据类型转换问题的解决办法：

- 考虑互操作性，不要公布那些在 Web 服务方法中的数值类型，可以创建封装器方法来公布并传递那些数值类型[27]；
- 对于因为不同语言数值类型与引用类型的问题，可以定义复合类型来封装数值类型，并将这个复合类型置为空来表示空引用。

数据类型和结构问题小结：

为了在使用数据类型时能够达到更好的互操作性，本文归纳出一般的规则如下：

（1）尽量多地使用简单数据类型。避免使用那些因为不同平台而导致的不同复合类型如 ArrayList, Tree，甚至公共的 Hashtable；

（2）即使是简单的数组一般都很难在不同的 Web 服务间获得良好的交互性，因而要特别注意数组中的内容，确保数组中的元素在每个平台上的含义都是相同的，并且避免使用含有空元素的数组；

（3）因为 XSD 与每个平台的本地数据类型很难保证是一一对应的关系，所以应该注意每个平台都是如何实现一些本地原始类型的，如 floatdouble dates 和 times。

4.4 原有系统向 Web Service 转化

在 SOA 框架中 Web Service 的产生除了新开发外，另一个途径是从已有的旧系统转化，由于 Web Service 是近几年兴起的技术，在此以前，各企业已经开发了许多信息系统，因此研究 SOA 架构下的旧系统向 Web Service 转化具有重要意义。

在 Internet 环境下，除了在.NET 环境下开发 Web Service,企业原有的各种用 VB、VC、ASP、Delphi 等语言编写的应用,如果能转化为 Web 服务,这将是企业应用集成最快捷、最经济的方式,这也成为企业关注的焦点所在。现有技术主要有 COBAR 以及 COM 两种转化技术，本文主要论述 Web Service 的 COM 转化技术。

COM 技术是微软公司的一项表现优秀的技术,以前的应用系统中有不少功能模块都用 COM 实现,以达到软件复用的目的。随着网络技术、电子商务技术的发展，新的技术构架 Web Service 技术，已迅速成为应用发展的重点。借助 XML 技术，可以实现各种应用程序资源的互联互通，大大促进了软件源的共享，从而成为新型的提供分布式的(全球性的)信息整合手段和应用系统 EAI 解决方案。旧有的 COM 模块在 Internet 环境下能否继续发挥功能复用的作用？这是企业应用集成中的一个普遍问题。

在多层架构(Multi-tier)中，通常会使用 COM 所开发的软件组件作为业务逻辑层。(Bisiness Logic Layer)事实上，多层架构与 Web Service 的概念可以互相对应。以三层式架构为例，分为客户端、业务逻辑层、数据库服务。各层分工负责，客户端访问业务逻辑层的服务时，无须知道每个服务与数据库处理的细节。只要每一层与其他各层通信接口定义明确，就可以根据需要与执行性能更换各层中的服务组件。

而 Web Service 的概念正好与这种多层次的概念对应起来。Web Service 的客户端与服务器端采用的是标准的 HTTP 等通信协议，服务消息包采用 SOAP 协议，因此服务接口可以明确定义。由于服务接口已成为标准，服务组件的开发将不会限于特定平台。正

是由于 COM 与 Web Service 在组织架构上的相似之处,解决企业用原有的各种用 VB、VC、ASP、Delphi 等语言编写的应用转化为 Web Service 问题,可以分以下几步来做：

- 首先用 COM 开发 Web Service 服务器端,并且注册到 Windows 系统中。
- 建立了服务端组件后,接下来便是让在 Internet 另一端的用户知道如何访问这个 Web Service，而这个部分便是靠 WSDL 对服务进行描述,在微软的 SOAP ToolKit3.0 中.可以使用 WSDL Generator 快速产生这个服务描述 WSDL 文件与服务组件 WSML 文件。WSML 是微软 SOAP Tookit 额外用来描述如何使用 COM 服务接口(Interface)文件。
- 开发客户端应用程序。

要注意的是，最后这些用 COM 组件所开发的服务组件，都必须注册到 Windows 系统中。而用来产生 WSDL 和 WSML 的 WSDL Generator，也必须已经注册并且包含类型库(typelibrary)的 COM 组件导入。而且 COM 组件的服务接口(Interface)必须继承自 IDispatch 接口。

VB6 转化 Web Service 举例：

1. COM 组件的开发

（1）新增一个 ActiveX DLL 项目。这个 ActiveX DLL 将会实现 IDispatch 接口。

（2）在 Project Explorer 与 Project Window 中命名此 Project 与 Class 名称。

新增一个 ActiveX DLL 项目,将 Project 名称改为 HelloWorldSvcRpcVb，将 Class 名称改为 HelloWorldServer。

需要注意，Project 加上 Class 的名称将会组成这个 COM 组件的 ProgID，因此所建立的 ActiveX 组件的 ProgID 即为：

HelloWorldSvcRpcVb. HelloWorldServer

稍后这个 COM 组件建立并成功注册到 Windows Registry 之后，便可以在 WindowsRegistry 中找到这个组件的注册信息。而在 Web Service 的建立过程中，我们将根据 ProgID 识别这个组件。

（3）在.cls 文件中撰写服务函数。

这个 Web Service 服务函数是 HelloWorld，没有输入参数，但是返回一个字符串“HelloWorld From VB!”。

```
Public Function HelloWorld( )As String
Hello World= " Helloworld   From VB! "
End Funcfion
```

注意 Visual Basic 在这里所用的数据类型，这个函数所返回的数据类型是字符串。因为 Web Service 或 COM 基本上都属于 RPC(Remote Procedure Call，远程过程调用)，这些服务所能使用的数据类型都有限制，在开发 Web Service 时必须注意是否支持所使用的数据类型。当然本例中的字符串是默认被 Web Service 支持的。

（4）存储这个项目到一个目录之下，这个目录设为：

[driver：]\Program Files\MSSOAP\SampleshHelloWorld\ServiceLRpc\VbSrv。

（5）建立这个项目。

选择[File]__[Make HelloWorldSvcRpcVb．dll]菜单命令，若建立过程中没有错误，将会在上述目录中产生 HelloWorldSvcRpcVb．dll，并自动在 WindowsRegistry 中注册这个 ActiveX DLL。所有注册过的 COM 组件都可以在 Windows Registry 中 HKEY-CLASSES-ROOT 下找到。

2. 使用 WSDL Generator 产生 WSDL 和 WSML 描述文件

建立了服务端组件后,接下来便是让在 Internet 另一端的用户知道如何访问这个 Web Service,而这个部分便是靠 WSDL 对服务进行描述,在 SOAP ToolKit3.0 中。可以使用 WSDL Generator 快速产生这个服务描述 WSDL 文件与服务组件 WSML 文件。WSML 是微软 SOAP Tookit 额外用来描述如何使用 COM 服务接口(Interface)文件。如果自己根据服务函数的规格撰写符合 Web Service 的服务描述，是一件很烦琐而且很容易出错的事情。

SOAP Toolkit 3．0 提供了 WSDL 和 WSML 产生工具，可以用这个工具导入 COM 组件，不用管 WSDL 与 WSML 的架构细节，由工具根据 COM 服务接口的定义，产生 WSDL 和 WSML。但是要注意的是，使用 WSDL 产生器以前，这些 COM 组件一定要先注册，而且含有 COM 组件的 DLL 文件必须包含类型库(Type Library)。

使用 WSDL Generator 的步骤如下：

（1）打开 WSDL Generator;

（2）选择服务组件，服务组件就是所实现的 COM 组件,请输入此组件的实际路径,WSDL Generator 将根据这个名称产生 WSdl;

（3）选择服务接口;

首先导入用 VB 6 开发的 Web Service，选择这个服务组件中想要对外提供服务的服务接口(如 HelloWorldServer)或是服务函数(如 HelloWorld)。一个 ActiveX DLL 可以有许多 COM Class 与服务接口，而每一服务接口又可以提供许多服务函数(methods)。WSDL Generator 可以只选择想要提供的服务与函数，没有选择的服务，不会出现在 WSDL 文件中。对于用 VB 6 所写的 COM 组件而言，WSDL Generator 所看到的是：

. CoClassName：HelloWorldSvcRpcVb(这是 VB 6 的项目名称)。

. InterfaceName：HelloWofldServer(这是 VB 中的 Class 名称)。

. Method：HelloWorld(这是 Method 名称)。

可以把这些名称与以前用 VB 6 开发服务器端组件的项目名称、Class 名称取得对应。值得注意的是，若服务函数中的参数或返回结果使用了 SOAPToolkit 不支持的数据类型，则在自动产生的 WSDL 文件中，函数中所使用的未被支持的类型会暂时以 " ?????? "，表示，稍后用户必须自行更正此数据类型，否则访问这个 Web Service

时将会发生解析错误。

对于用VC6，Delphi等所写的COM组件, WSDL Generator所看到的与上述类似。

（4）设置Listener的URL与类型；

使用SOAP Toolkit API访问Web Service，在服务器端必须有个Listener(接受服务请求者)，这个Listener可以是ISAPI或ASP。若选择ISAPI，则产生的WSDL文件中服务地址为WSdl所在的URL。在URI字段中，必须输入Listener所在的服务地址，而不是一个本地计算机中的实际路径。譬如，在本例中WSDL的服务地址为：http://localhost/Web Service

HelloWorld/Service/Rpc/VbSrv/HelloWorld.WSdl

（5）产生WSDL文件

单击Select按钮,选择要存放WSDL的路径,SOAP消息默认编码方式为UTF-8,保持这个设置值不变即可,另外WSDL Generator会在指定的存放目录下放置两个文件,即WSDL和WSML,这个实际目录必须对应到前面步骤中的虚拟目录。

. WSdl在这个文件中描述了调用Web Service服务函数的细节。如果服务器端Listener为ISAPI，在<soap：address>段就会以.WSdl .asp扩展名作为服务路径的结尾，如果服务器端Listener为ASP,就会以.asp扩展名作为服务路径的结尾。

. WSml是微软用SOAPToolKit实现Web Service时专有的文件,用来将COM组件与WSDL的服务产生对应。除了少数特别的服务如自定义类型转换器，WSml只用在服务器端,客户端并不需要。

当服务器端组件与服务描述文件都已准备完毕后,客户端便可以开始通过以HTTP、SOAP通信协议调用这个网际服务了。

3．客户端程序的编写

当Web Service的调用者前端为窗口应用程序时，(使用VB6)如果使用SOAP Toolkit，则客户端必须安装mssoapl.dll(在WindoWS Registry中注册)。而应用程序是通过SoapClient调用Web Service的，所以不管是VB 6的项目还是VC 6的项目，在调用SoapClient时，都必须先加入mssoapl.dll的引用(reference)。

客户端VB 6项目：

[driver:]\ProgramFiles\MSSOAP\SamplesHelloWorld\Client\Rpc\Vb\HelloWorldCliRpcVb.vbp

VB 6的项目引用mssoapl．dll，也就是取得SOAP的Type Library的方法是，选择[Project]_[References]菜单命令，将弹出对话框，其中有所有注册在WindoWS Registry中的ActiveX组件。

若MS SOAP组件已经注册，则菜单中会出现Microsoft Soap Type Library的选项，这时选中它，表示将在这个项目中引用Soap前端组件。

引用了Microsoft SOAP组件之后，可以在VB 6的对象浏览器(Object BroWSer)中

查看这个对象的方法(Methods)与属性(Properties)。我们将依此建立 SoapClient 对象，并调用 mssoapinit 方法。

用 VB 6 撰写窗口应用程序作为 Web Service 客户端，调用 Web Service 的程序代码如下所示。

```
Private Client As SoapClient
        Private sConnectedWSDL As String
Private Sub Connect()
        SetClient=New SoapCknt.
        Client.mssoapinit sConnectedWSDL
End Sub
Private Sub Command l_Click()
        Connect
        MsgBox CStr(Client.HelloWorld)
End Sub
Private Sub Form_Load ()
        SConnectedWSDL="http://localhost/Web
   Service/HelloWorld/Service/Rpc/VbSrv/HelloWbrid.WSDL"
End Sub
```

A SP 调用 W eb Service 的基本代码

```
            <%Dim soapclient
          '设置 WSdl 文件路径变量
          Const WSDI. URI=" G:
\Myproject\server\server\mainContractWSdl "
          '创建 SoapClient
          set soapclient=ServerC reateObject("MSSOAP.
SoapClient" )
           Soapclient ClientProperty(" ServerH TTPR equest ")= True
           Soapclientm ssoap in it WSDL_URL %>
```

对于用 VC6、Delphi 等所写的 COM 组件，向 Web Service 转化的步骤和方法与此类似。限于篇幅，不多叙述。

4.5 本章小结

本章在第二、三章的基础上提出了一个 SOA 集成框架，并对其具有的各功能模块实现技术进行了论述分析，对集成框架中的 Web Service 互操作问题进行了较深入的研究讨论，给出了对应的解决办法；对框架中的旧有系统 Web Service 转化问题，给出了 COM 方式的解决方案。

第五章 案例分析—中石油股份公司项目管理系统

系统的开发环境：

- 开发语言及技术：微软公司.NET 平台 C#语言，ASP.NET、ASP 等程序设计语言和技术；
- Web 及事务服务器：采用微软公司 IIS 6.0＋微软公司的 VS.NET2003＋SOAP-TOOKIT3.0，提供了 ASP 和 ASP.NETWeb 应用程序以及 Web Service 的容器和运行环境；
- 数据库服务器：采用 SQLSERVER 2000 作为数据库服务器；
- 操作系统：WINDOWS 2000 SERVER。

5.1 系统功能及集成需求分析

项目管理系统简介：

中国石油勘探开发研究院中国石油集团科学研究院为了加强应用基础研究和原创性技术研发工作，鼓励本院中青年骨干积极开展应用基础研究和技术创新，以提升其发展后劲，决定设立中国石油勘探开发研究院中青年创新基金(以下简称"创新基金")。创新基金用以资助对石油工业的持续发展具有创新意义的、有探索性的科研项目。包括：在学术上有创新的基础研究和应用基础研究；在技术上具有独创性、先进性的新工艺、新产品、新材料的开发性研究；以及对石油工业发展有深远影响的战略、管理、经济、环保研究等。为了迅速即时地方便广大创新基金申请者申报项目，开发了 B/S 模式的基金项目管理系统，以 ASP 语言实现。随后，在使用过程中，为了配合管理系统共同工作,减少项目评审、立项的过程中人为的干扰因素，充分体现公平、公正、公开的原则，又设计了专家系统以 ASP.NET 实现。项目管理工作流程如图 5－1 示：

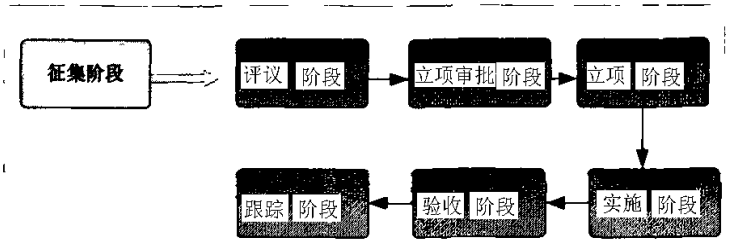

图 5-1 项目管理工作流程

1. 征集阶段

征集阶段是指从发布项目（课题）征集信息到通过项目（课题）初审，进入项目年度库的过程。公司项目管理中心全年受理有关单位的项目（课题）建议，申报材料为"项目（课题）建议书"，该类建议通过专家评审立项后将作为基金项目予以支持。

2. 评议阶段

评议阶段是指项目年度库中的项目（课题）建议经专家评议，建议立项项目（课题）进入储备项目（课题）库的过程。该阶段将组织有关专家对上一阶段征集到的项目进行评审，评审结果作为项目立项的参考依据。

3. 立项审批阶段

立项审批阶段是指储备项目（课题）库中的项目（课题）经公司项目管理处审批确定立项的过程。对于一般项目，是指直接从储备项目（课题）库中择优选取课题建议书报公司审批确定立项的过程。该阶段通过审批的项目进入项目领域库。

4. 立项阶段

立项阶段是指已经确定立项的项目（课题）登录申报管理系统并签订项目（课题）合同任务书的过程。

5. 实施阶段

实施阶段是指监督、检查项目（课题）单位按照项目（课题）任务书计划完成有关任务的情况。

6. 验收阶段

验收阶段指项目（课题）从申请验收，经项目经费决算确认、专家验收到中石油公司批复验收确认的过程。

现有的项目管理系统包含用户管理模块、项目汇总模块、立项管理、项目初选、项目结题、资料搜索、相关通知、申报项目、提交附件。之后，又用 ASP.NET 开发了专家系统，是为了在项目评审、立项的过程中，减少人为的干扰因素，配合项目管理系统共同工作而设计的。主要功能是对准备立项的项目进行专家会审，提高项目立项的科学性和公正性。主要包括专家系统管理人员添加系统用户以并按照权限模版管理用户模块、为专家分配项目模块、各位专家进行评审打分模块等模块。但是，前后两系统以不同的语言实现，分别实现了各自的功能，在协调工作方面存在如下问题：

- 项目管理系统中存在的项目登记、审批模块，不能完全满足基金项目的发展需求，新的专家系统要继承并沿用项目管理系统的数据，并且两个系统之间要有数据的互换。
- 在现有的项目登记表中，项目的名称、编码也存在一些不规范的地方，发现经常有项目描述错误、编码重复的现象；
- 项目管理系统的用户没有一个有效途径了解相关项目的评审情况，不能及时地做出工作调整，修改存在的缺点，来促进本单位的项目申报工作。在项目管理中产生的矛盾随着公司的项目申报的增长和竞争日显突出。

申报者的信息提交给了先前的系统，后面的专家系统存在大量的手工操作，没有一个规范、高效的工作流程，没有一个信息化的管理系统，导致信息资源不清晰，信息无法共享，信息追踪困难等问题的产生，制约着基金项目申报工作进一步的拓展。为了能够减少并抛弃手工录入资料，互相协作，共同完成基金项目申请管理的任务，使得项目

申报者方便快捷地了解项目的评审情况，必须进行集成。集成的需求可以表达如图 5—2 所示：

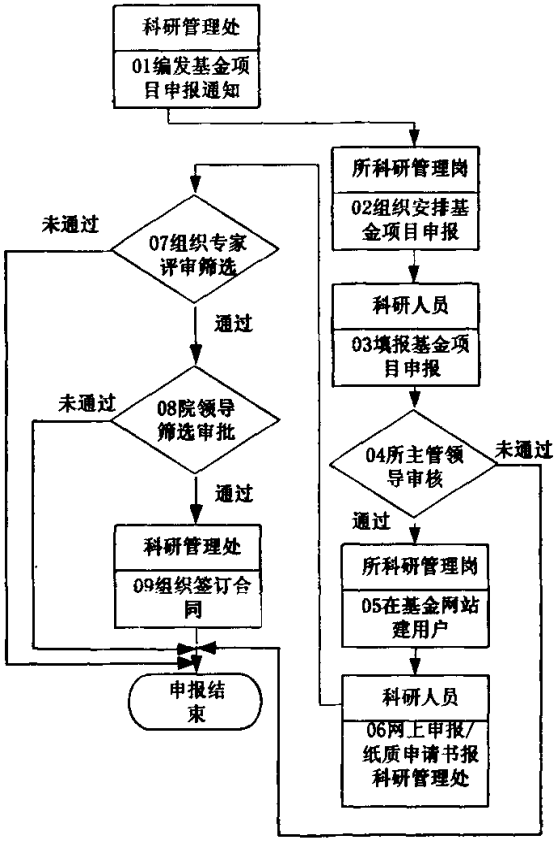

图 5-2 系统集成流程图

5.2 系统集成设计

5.2.1 系统模式及总体架构

1. 系统模式

本系统是为了解决现有管理模式的不足而设计的，所以实现的最重要的一个目标便是解决项目管理系统内部的管理不足以及与外部专家系统上行下达的资源共享，使之能更好地为上下级部门提供各项服务，最终提高全公司的生产管理效率。所以，本系统要解决的问题的重点放在了安全实用的用户权限、账号的管理，用户协同工作，灵活方便的多条件查询，信息的交互共享，格式规范的报表的生成等方面。

为了使系统真正能达到信息共存、维护方便、简单、节省办公成本、减少数据冗余、软件升级方便等要求，本系统决定采用基于 Web 的分布式体系结构，即 B/S 模式下的多层分布式系统。

在基于 B/S 结构的工作模式中，通常都是用户端通过 IE 或 NETSCAPE 等浏览器选择感兴趣的内容，提出请求给 Web 服务器。Web 服务器再到相应的数据库服务器提取

相应的内容返回给客户浏览器，实现一种更广泛意义上的信息共享。系统采用基于浏览器、应用服务集群、数据服务集群三层体系架构，利用中间件技术、Web Service 技术实现系统集成、数据交互和分布式计算，充分保证应用程序平台的可扩展性(application extension)，应用程序平台环境(application environment)的健壮性和易用性，保证客户可用性和高级安全性。此外，系统核心平台提供开放接口以允许第三方软件供应商提供基于此平台的应用服务，应用平台能管理这些应用服务。

2. 系统总体设计

由第 4 章提出的系统整体架构模型图，将 WSA-EAI 框架应用于"中石油基金项目管理"系统中，结合项目的实际情况，采取了较为灵活的集成策略，对 WSA-EAI 进行了相应的取舍和补充，整个系统的框架模型如图 5-3 所示。该模型框架总体上由客户访问层，应用层、应用集成层、数据访问层、数据层等部分组成。

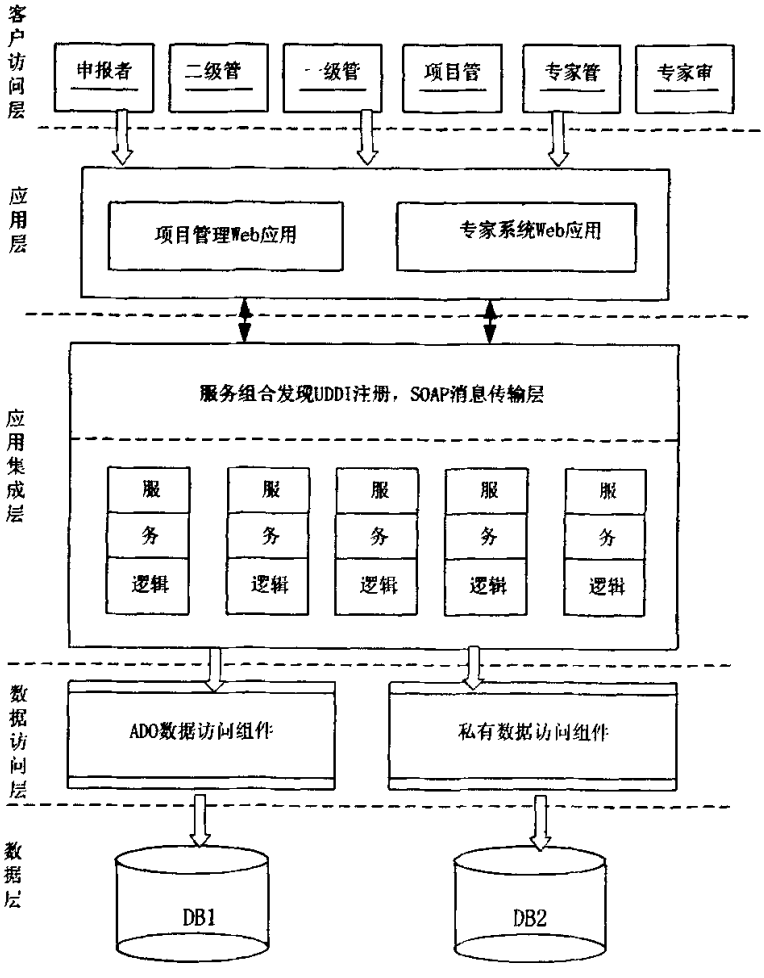

图 5-3 集成系统框架设计

（1）客户访问层

采用 B/S 模式，以 IE 浏览器实现客户访问。项目申报者、单位二级管理人、一级管理人员、项目管理人员、评审专家、专家系统管理人员、以及项目系统管理人员不同角色的用户，通过 IE 浏览器，在任何时间、地点，只要能上网，就能够使用应用系统。在此层，对于项目管理系统的管理人员，在专家系统赋予特殊的角色，用以浏览专家系统的一些信息，故建立单一登录模块供其使用。

（2）应用层

它是本集成系统向企业和个人所提供的各种 Web 应用。这些应用包括用户管理、项目汇总、立项管理、项目初选、项目结题、资料搜索、相关通知、申报项目、提交附件、专家系统添加并管理专家模块、为专家分配项目模块、专家评审模块、按照权限模版管理权限等模块。这些应用由不同角色的用户进行调用。

（3）应用集成层

企业应用集成层是 SOA 系统的核心部件。企业应用集成层、服务提供与封装层、企业私有 UDDI 注册中心以及公共 UDDI 注册中心一起构成了 Web 服务的标准运行组件。在实际实施时，与 SOAP 消息传输层合并为一层，企业应用集成层自身包含了多个组件用户身份认证、消息服务、服务事务、Web 服务安全控制等。集成层的服务实体由 VS．NET 平台下开发的 Web Service 构成。SOAP 消息传输层、服务查找、绑定、以及代理类的生成由 VS．NET 下的 Web 服务开发工具自动完成。

（4）数据访问层

项目管理系统采用 ADO 技术，专家系统采用 ADO.NET+私有数据访问组件，主要提供应用系统所需的各种数据，主要包括数据存储与管理、数据传输和数据采集等[28-30]。

（5）数据层

SQLSERVER2000 数据库。使用其视图、存储过程、锁技术以及．NET 框架下的 DataSet、DataAdapter 等技术。

5.2.2 集成系统流程和结构的设计

1．原项目系统中权限设置

原项目系统中安全方面及系统组织的权限设置如下：

系统管理员权限为 1；局长权限为 3；副局权限为 4；处长权限为 5；副处权限为 6；管理权限为 7；其他权限为 8；企业用户企业权限为 9；二级权限为 10；个人权限为 11。

不同权限用户所影响审批 status 值含义设置：

status=1 表示项目基本资料完善；status=2 表示申报书提交；3 表示单位 pass；4 表示初选 pass；5 表示项目 pass；6 表示 pass 原因；7 表示二级单位推荐；8 表示单位推荐通过；9 表示初选通过；10 保留为空；11 表示项目立项；12 表示合同书提交；13 表示请提交进度报告；14 保留为空；15 表示提交进度报告；16 表示提交结题申请；17 表示提交结题报告；18 表示项目结题；19 表示项目存档。

2．集成系统中多级权限的管理和流程组织设计

系统中不同职务的人员，对应了相应的权限。权限设计较多，管理复杂，进行集成时把不同职务加入各自安全角色，按照角色组织系统流程来简化并进行安全有效的管理，并对多种权限进行保留，留待新的工作需求出现时予以使用。共分以下几种角色：

系统管理人员角色、公司项目管理人员角色、单位项目管理人员角色、部门项目管理人员角色、项目申报者角色、专家系统管理人员角色、评审系统管理人员角色、评审专家角色。

以下分别对其功能进行介绍：

项目系统的系统管理人员登录后，可以进行以下工作：添加公司项目管理人员用户，并对其信息进行维护修改。

公司项目管理人员登录后，可以进行以下工作：添加单位项目管理人员用户，并对其信息进行维护修改。可以进行初选工作，查看管理立项工作。

单位项目管理人员登录后，可以进行以下工作：添加部门项目管理人员用户，并对其信息进行维护修改。可添加项目申报者，可以查看管理本单位申报项目的情况。

部门项目管理人员登录后，可以进行以下工作：可添加项目申报者，并对其信息进行维护修改。可以查看管理本单位申报项目的情况。

项目申报者进入系统后，可以进行以下工作：申报项目，提交附件，查看自己所报的项目情况，阅读公司的通知。

专家系统管理人员登录后，可以进行以下工作：添加公司专家系统管理人员用户，并对其信息进行维护修改。

专家系统管理人员登录后，可以进行以下工作：添加项目专家评审人员用户，并对其信息进行维护修改。

项目评审专家进入系统后，可以进行以下工作：查看需要自己评审的项目情况，进行评审，并将评审结论予以提交。阅读公司的通知。

集成系统的运行实现方式如下：

项目管理系统的各类用户按角色分配以不同权限，用户登录后，进入不同权限的用户界面（主要以各界面的功能菜单不同区分权限）。申报者，部门管理人员，单位管理人员，项目管理人员等在各自的页面针对项目进行各自的业务处理，处理结果对相应的项目在项目主表中的状态产生影响，以 status 值 VALUE 表示。

在专家系统中，针对不同角色的用户建立权限模板，当不同角色的用户登录时，按照不同的权限模板号，提供不同的功能菜单界面。各类用户在各自的用户页面，由不同的功能菜单完成各自的业务逻辑（从执行效率角度出发，在对数据库操作时可以利用在私有数据访问组件 DBConn 中封装存储过程的技术来完成业务逻辑的执行），针对具体的项目进行相关的业务处理，处理结果对特定项目在专家系统的项目表中的状态产生影响，以 statu 表示。

按照 UML 统一建模语言进行建模，集成系统的运行流程的简要时序作用图设计如图 5－4 所示：

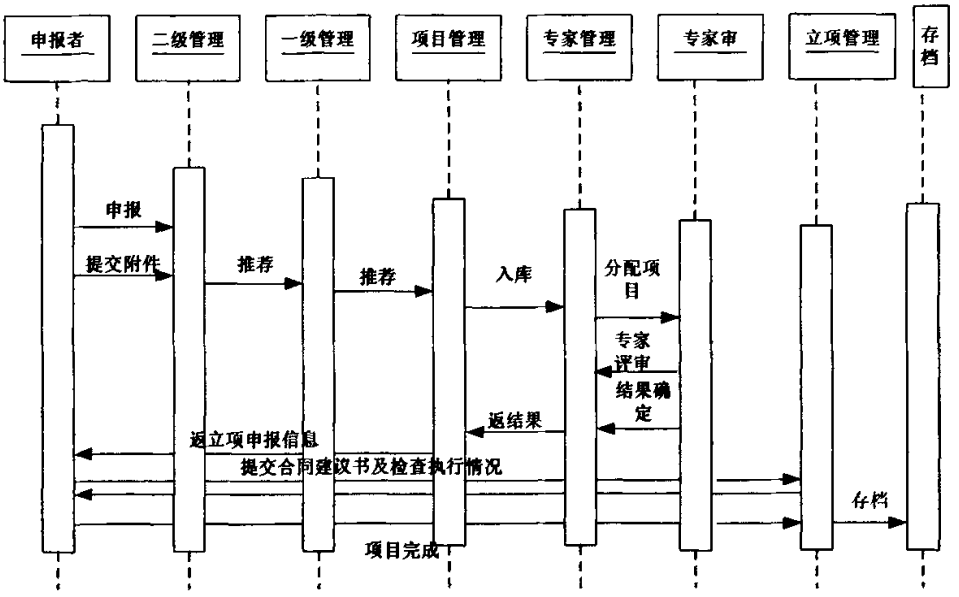

图 5-4 集成系统时序图

3. 按照本文提出的 SOA 设计方法，可以对原有的项目管理系统和专家系统从结构上做以下处理：

（1）将项目管理系统分解为项目申报，提交附件，项目推荐，项目初选，立项管理其他功能和查询等部分。如图 5-5 所示。

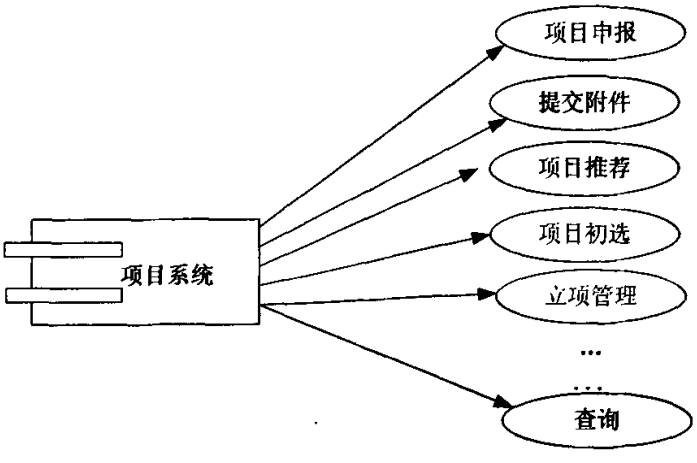

图 5-5 项目系统分解

（2）将专家系统分为项目入库，分配专家，项目评审，确定结论，提供各类查询。如图 5-6 所示。

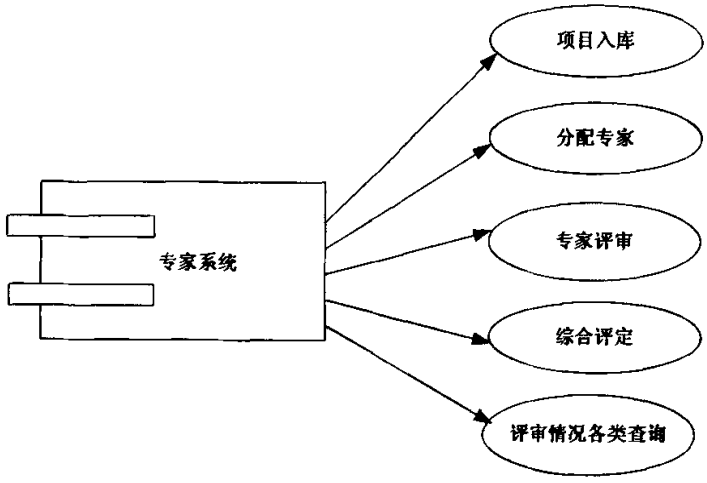

图 5-6 专家系统分解

（3）项目系统中的项目管理人员，申报用户的查询都会用到专家系统的评审情况信息。专家系统的查询服务提供了各种条件下项目信息查询，所以，项目管理系统的项目管理人员，申报用户都可以直接引用专家系统的项目评审查询服务。如图 5-7 所示。

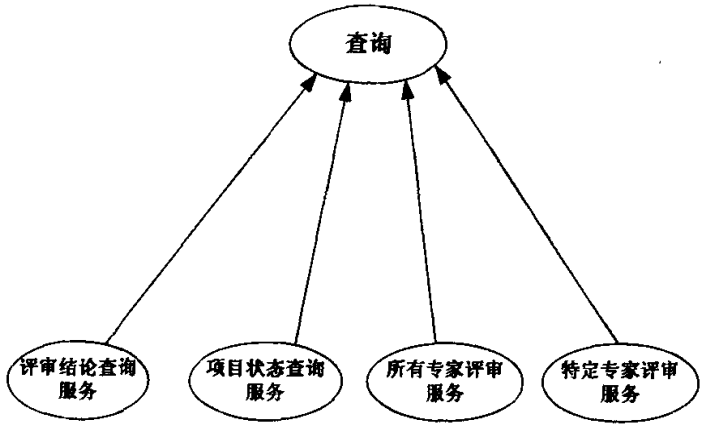

图 5-7 查询服务调用

5.3 基于 SOA 的集成实现

5.3.1 客户访问层

对于一个企业来说，它的客户一般分为以下几类：Web 客户、一般的服务请求者和商业合作伙伴。它们通过企业防火墙和外部交互网关后访问企业应用层的典型应用，充当了 Service Requester 的角色，它们可能会通过各种协议方式(如 HTTP, RMI-IIOP 等)来访问企业应用层，而对客户来说这些实现的细节都是透明的。为了促进和方便申报基金项目，本案例的客户访问层采用 B/S 模式，用 IE 浏览器实现。无论申报者在何时何地，只要能上网，就可以进入系统进行申报。另外，除了项目管理系统的管理人员可以

调用专家系统的集成功能，辅助其管理项目申报外，对于项目管理系统的系统用户角色，可以在专家系统赋予其特殊的权限角色，使其获得比管理人员更多的权利，主要是在专家系统系统能执行更多的浏览功能。为此，设计了一个单一登录模块，介绍如下：

1．在项目管理系统相关用户的应用页面设置链接，指向专家系统的登录页面。

2．在项目管理系统中设置 Session 会话变量，保留其登录的用户名和密码，并用正确的格式传递参数：

<a href="http://localhost/prof/loginx.aspx?UserName=<%=session("username")%>&Password=<%=session("Password")%>">

3．在专家系统利用权限模板为此类用户新增一个特别权限模板，供其使用。这样，项目管理系统的特别用户登录到专家系统时，不用再输入用户名和密码即可进入专家系统，以特别权限模板行使其权利。

5.3.2 应用层

本系统的应用按角色划分而不同，主要角色有：项目申报者、项目二级管理者、一级管理者、项目管理人员、系统管理人员、专家评审者、评审管理人员、专家系统管理人员等。角色不同，其调用的职能应用也就各自不同，现以项目管理人员为例，介绍应用层的设计情况：

5.3.2.1 管理人员主要应用模块：

管理人员登录后，出现：〖首页 | 项目初选 | 立项管理 | 项目结题 | 项目资料搜索 | 用户管理 | 添加相关文件 | 相关文件 | 退出系统〗9 个操作栏目。

1．首页：为本系统的首页，有用户的欢迎信息、系统说明和已经立项的项目展示和附件模版下载等。

2．项目初选：申报单位所有申报的创新基金的项目都在此以项目的申报前后顺序排列(后申报的项目在前)，内设：按专业汇总、按单位汇总、管理初步选定的项目三个栏目。

默认为按专业汇总，管理者可以点击后面的专业来查看此专业的项目。由于每年申报的创新基金的项目很多，管理者可通过此功能初步选定一些项目，然后参加会审，减少管理者每一个项目都会审的复杂程度。默认的申报的项目并没有通过初选，管理者需要在此设定哪一个项目通过初选，然后会审后立项。点击任何一个项目的名称可查看此项目所有提交的项目资料，包括项目简介、《申报书》、以后提交的《合同书》、《项目进展情况》、《结题报告》等。在以后的任何时候，点击任何一个项目的名称均可查看此项目所有提交的项目资料。管理者可在任何一个认为可以通过初选的项目后面打“v”后，点击下面的按钮来确定此项目通过初选。通过初选的项目才可立项。

管理者可以选择：按单位汇总、按专业汇总来查看/初选项目。

管理初步选定的项目栏目是为了防止管理者操作失误而将一些不应该通过初选的项目通过初选，在此栏目中，管理者可以选择不应该通过初选的项目，将其从初步选定的项目中去掉。

3．立项管理：此处所有项目都是通过初选的项目。此栏目也分为：项目立项、管理立项项目、管理未批准项目三个子栏目。

管理者可在项目立项栏目中,调用专家系统的评审结果模块，设定立项项目的拨款经费，然后立项。立项后的项目不允许申报单位修改任何信息，不允许删除。

管理立项项目栏目是为了防止管理者不小心将不应该立项的项目立项，在此处可以将已经立项的项目设置为不立项。

管理未批准项目栏目中显示的是所有没有立项的和没有通过初选的项目，管理者可以通过此栏目来设置某项目没有被批准的原因。点击后，在下面的原因中选择即可。表示管理者已经设置了此项目没有被批准的原因。

4．项目结题：此栏目中显示的是已经提交了结题报告申请结题的项目，管理者可以查看结题报告并让此项目通过结题。

5．项目资料搜索：管理者可以通过在搜索框中填入任何和一个项目或者几个项目相关的信息来搜索，如可以填入项目的单位名称、项目负责人姓氏、项目负责人的电话号码等。

6．用户管理：管理者可以查看任何用户的信息，可以删除用户(注：如果删除一个用户，此用户相关的项目信息均被删除)，可以查看用户的登录次数、用户的最后一次登录时间、用户名、密码等。

7．添加相关文件：管理者可以在此添加一些和创新基金相关的文件，加入的文件将在相关文件栏目中展示，所有人均可查看。也可以发布通知。

8．相关文件：管理者可以在此修改、删除已经添加的文件，任何别的用户选择相关文件后，没有修改和删除的权限。

5.3.2.2 集成信息传递

下面介绍集成信息传递方面主要的三个模块情况。

1．入库模块的实现

由 Promanagel.aspx 页面显示所有传递过来的项目管理系统的项目信息，点击后转向 Editprojectl.aspx 页面，此页面显示特定项目的所有信息，Editprojectl.aspx 页面的后台代码调用存储过程 addProjectl 实现申报项目在专家系统的入库操作。

Editprojectl.aspx 页面逻辑流程如图 5－8 所示：

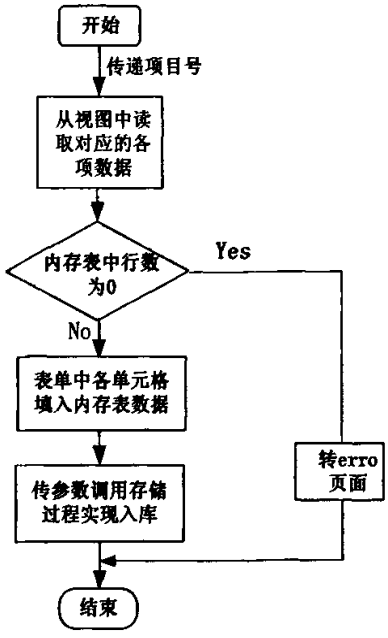

图 5-8 入库模块流程图

由于在 B/S 模式下，网站的响应时间是设计中要考虑的一个重要因素，存储过程在执行效率方面有着优良的性能，故业务逻辑中对数据库的操作，采用数据组件 DBConn 内封装存储过程的技术。addProjectl 的代码如下：

```
        CREATE      PROC addProjectl
      @projectid int
   AS
   declare @deptID int
   declare @SpecID int
   declare @deptName varchar(50)
   declare @SpecName varchar(50)
   if not exists(select * from project where projectid=@projectid)
   begin
      select    @deptName=projectdept,@SpecName=projectspec    from    v_allproject    where
projectid=@projectid
      if (not exists(select * from spec where specname=@SpecName))
         insert into dbo.spec(specname) values(@SpecName)
      if (not exists(select * from dbo.dept where deptname=@deptName))
         insert into dbo.dept(deptname,deptaddress) values(@deptName,'系统自动添加,请人工修
改')
      select @SpecID=specid   from spec where specname=@SpecName
      select @deptID=deptid from dbo.dept where deptname=@deptName

      insertinto
dbo.project(projectID,projectname,projectAuther,projectdept,projectprofile,startdate,enddate,status,Project
```

```
Money,Spec,attachFile)
        select projectID,projectname,projectAuther,@deptid,projectprofile,startdate,enddate,'申请
',projectMoney,@SpecID,附件地址 from v_allproject where projectid=@projectid
    end
    GO
```

2. 附件信息集成模块

原项目管理系统中项目申报者申报项目时所提交的附件，其内容没有保存在该项目对应的数据库 NEWCXDB 中，其中只有每个项目对应的地址，显然，从项目管理系统的数据库中无法把内容传递到专家系统的数据库中去。原专家系统的专家评审模块中，专家工作界面上，工作内容中每一项待审项目其项目号为专家系统自己的随机号码，根数据库中 NEWCXDB 项目号码不同，而项目的附件地址与项目号码有关联，怎样使专家能够拿到申报者提交的立项建议书创新基金合同书，项目进展情况等附件进行审阅呢用到如下技术：

（1）在对应 ASP.NET 的 CS 文件中利用 System.Web.UI.HtmlControls.HtmlTable TableView 的动态数据可加链接的功能，给相应的单元格对应的动态数据库内容增加链接，代码：

```
this.TableView.Rows[3].Cells[1].InnerHtml=ds.Tables
["maintable"].Rows[0]["projectname"].ToString();
this.TableView.RoWS[3].Cells[1].Attributes.Add
("onclick","location='http://localhost/xm/proshowl.asp?id="+ID.ToString() +"'");
```

（2）把本页接受到的动态项目号码变量作为参数，传递给项目管理系统对应功能单元，实现互相调用。但有一个重要问题，传递的随机号码在项目管理系统中没有对应的内容，原项目系统中附件只与本系统的项目号码关联，如何使得专家系统费力传出的值对项目管理系统有用？

（3）改写专家系统评审模块对应的网页 CS 文件，使其动态呈现项目号码，以使可用。阅读 CS 文件，找到动态呈现随机号码的数据库来源，是一视图，其内容：

```
    CREATE  VIEW dbo.V_MyProject
AS
SELECT t1.projectrandom, t1.projectname, t1.projectprofile, t1.enddate, t1.status,
      t1.ProjectMoney, t4.SpecName, t2.profID,t2.IsAppraise,t3.profname,t1.attachfile
FROM dbo.project t1 INNER JOIN
          (select projectrandom,profid,case when evaluatedate is null then '待评' else '已评阅'
  end as IsAppraise from dbo.evalproject) t2
      ON t1.projectrandom = t2.projectrandom INNER JOIN
      dbo.profuser t3 ON t2.profID = t3.profid inner join spec t4
      on t1.Spec=t4.SpecID
```

可见，项目随机号码 projectrandom 涉及两个表 dbo.project 和 dbo.evalproject，改写此视图，改写两个表 dbo.project 和 dbo.evalproject 的设计项，使其提供项目编号；同时，由于专家评审模块中，专家工作界面上，工作内容是由上一模块专家系统管理人员的分配模块而来，包含随机号码 projectrandom，修改此模块，使其传递项目编号。

```
……
private void btnPass_Click(object sender, System.EventArgs e)
        {
            try
            {
                string ID;
                ID=this.GetID().ToString();
                DBConn.ExeSql("exec addProject " + ID);//添加项目
                //添加评审专家
                string strProfID;

                foreach(ListItem li in this.ListAleady.Items)
                {
                    strProfID=li.Value.Substring(0,4);
                    DBConn.ExeSql("exec addProf " + ID + "," + strProfID);
                }
            }
 ……
```

a. 其中相关的 addProf 存储过程如下：

```
 CREATE     PROC addProf
     @ProjectID int,
     @ProfID int
AS
declare @ProjectRandom int
select @ProjectRandom=projectrandom   from project where projectid=@ProjectID
if(@ProjectRandom is not null)
Begin
if (not exists(select * from evalproject where projectrandom=@ProjectRandom and
profID=@ProfID))
     insert into dbo.evalproject(projectrandom,profID) values(@ProjectRandom,@ProfID)
if(not exists(select * from evalprojdetail where projectrandom=@ProjectRandom and
profID=@ProfID))
insert into dbo.evalprojdetail(projectrandom,profid,sn,sn_name)
      select @ProjectRandom,@ProfID,sn,[Name] from dbo.snname
End
GO
```

b. 对其涉及 ProjectRandom 内容，修改此存储过程，并且发现此 addProf 存储过程涉及两个表 evalprojdetail， dbo.evalproject，对两个表进行修改设计。

c. 另外还发现，评审专家的工作内容模块涉及两个视图 V_evalProjdetail，

V_evalproject，以及另外的两个存储过程。

```
CREATE VIEW dbo.V_evalProjdetail
AS
SELECT t1.sn_name, t1.notion, t1.num, t1.sn, t1.profid, t1.projectrandom, t2.SNMemo,
      t2.SNMaxNum
FROM dbo.evalprojdetail t1 INNER JOIN
      dbo.snname t2 ON t1.sn = t2.sn

CREATE VIEW dbo.V_evalproject
AS
SELECT dbo.evalproject.*, dbo.profuser.profname AS profname,
      dbo.project.projectID AS projectID
FROM dbo.evalproject INNER JOIN
      dbo.profuser ON dbo.evalproject.profID = dbo.profuser.profid INNER JOIN
      dbo.project ON dbo.evalproject.projectrandom = dbo.project.projectrandom
```

涉及的存储过程有：

```
create    PROC AddDetailNotion
    @projectrandom int,
    @ProfID int,
    @Sn int,
    @ProjectNum int,
    @ProjectNotion varchar(600)
AS
update evalprojdetail set num=@ProjectNum,notion=@ProjectNotion
where projectrandom=@projectrandom and profid=@ProfID and sn=@Sn

GO
CREATE    PROC AddTotalNotion
    @projectrandom int,
    @ProfID int,
    @ProjectNotion varchar(600)
AS
update evalproject set totalNUM=(select sum(num) from evalprojdetail where
projectrandom=@projectrandom and profid=@ProfID),totalnotion=@ProjectNotion
where projectrandom=@projectrandom and profid=@ProfID

if (not exists(select * from dbo.evalprojdetail where projectrandom=@projectrandom and
profid=@ProfID and num is null))
      update evalproject set evaluatedate=Getdate()
        where projectrandom=@projectrandom and profid=@ProfID
GO
```

至此，进行了以上修改后，实现了设计功能。

3．网上信息平台

在实现了项目管理系统和专家系统的业务协作集成后，为了在专家系统的用户之

间，进行实时交流，设计了系统的专用信息交流平台，供本系统的用户实现网上实时互动。

（1）实现的功能

功能 1：在线用户之间可以收发短信。

功能 2：接受方开始在线，发送过程中，当接受方下线后，信息依然可以保留到下次该用户再登录时接收到。

功能 3：用户也可以给非在线用户发短信。

（2）实现的技术步骤

a．设计即时打开新窗口的 JavaScript 函数。

```
<script language="JavaScript" type="text/JavaScript">
        <!--
        function MM_openBrWindow(theURL,winName,features) { //v2.0

window.open(theURL,"_blank","status=no,resizable=0,toolbar=no,menubar=no,scrollbars=no,width=
300,height=160,left=200,top=150");
        }
        //-->
    </script>
```

b．利用 System.Web.UI.HtmlControls.HtmlTable 控件的传递参数和行为的功能，添加行为和参数。用到的主要代码：

```
            htmlrow=new HtmlTableRow();
            htmlcell=new HtmlTableCell();
            htmlcell.Attributes.Add("style","cursor:hand");
    htmlcell.Attributes.Add("onClick","MM_openBrWindow('SendMsg.aspx?ProfID="+
ds.Tables["maintable"].RoWS[5*rowCount+j]["ProfID"].ToString() + "')");
    htmlcell.InnerText=ds.Tables["maintable"].RoWS[5*rowCount+j]["ProfName"].ToString();
            htmlcell.Width="20%";
           htmlrow.Cells.Add(htmlcell);
```

c．发送功能设计。CS 文件的主要代码：

```
……
    private void btnSend_Click(object sender, System.EventArgs e)
    {
        ……
            string strSql;
            string Msg;
            Msg=this.txtSendMsg.Text;
            Msg=Msg.Replace("'","''");
            strSql="insert into Message(ProfID, Message, IsRead, MsgDate, SenderID) "
+
                " values(" + this.GetProfID().ToString() + ", '"+ Msg + "', 0, getdate(), "
+ Session["ProfID"] + ")";
```

```
                    DBConn.ExeSql(strSql);
                        this.txtMessage.Text=this.txtMessage.Text+"\n("+
    DateTime.Now.ToString() + ")" + Session["ProfName"] + "\n";
                    this.txtMessage.Text +=this.txtSendMsg.Text;
                    this.txtSendMsg.Text="";
                }
            }

    ……
```

d. 接收功能在设计上使用框架集技术。main.htm 页面使用框架集，包含 login.aspx 页面和 top.aspx 两个页面。其中 login.aspx 页面负责登录验证，另一个页面 top.aspx 负责对消息进行检测接收。

top.aspx.cs 主要代码设计：

```
        private void Page_Load(object sender, System.EventArgs e)
        {
            try
            {
                string strSql;
                if (this.Session["ProfID"]!=null)
                {
                    strSql="select * from Message where IsRead=0 and Profid=" +
Session["ProfID"];
                    DataSet ds=new DataSet();
                    DBConn.Fill(ref ds,strSql);
                    for(int i=ds.Tables[0].RoWS.Count-1;i>=0;i--)
                    {
                        this.Response.Write("<SCRIPT
language=javascript>window.open(\"Msg.aspx?MsgID="+ ds.Tables[0].RoWS[i]["MessageID"].ToString()
+
"\",\"_blank\",\"status=no,resizable=0,toolbar=no,menubar=no,scrollbars=no,width=300,height=160,left=2
00,top=150\");</SCRIPT>");
                        DBConn.ExeSql("update Message set IsRead=1 where MessageID=" +
ds.Tables[0].RoWS[i]["MessageID"].ToString()); //置消息为已读
                    }
                }
            }
            catch(System.Exception ee)
            {
                DBConn.WriteLog("判断是否有短信时出错：" + ee.Message);
            }
```

Top.aspx 框架在接收方用户登录时开始发挥作用，检测有无该用户的信息来通知用户。若有某用户的信息，信息在接收方以 Msg.aspx 阅读页面呈现给此用户，Msg.aspx

页面可以再回复给来信者。具体的作用机制为：

（1）发送方发送的消息中附带有接受方专家参数 ProfID，发送方参数 Sender ID；

（2）接收方登陆时， Top.aspx.页面的后台代码查询数据库表 message，把接收方参数 ProfID 为该用户的消息号 MessgeID 传给消息页面 Msg.aspx；

（3）Msg.aspx 消息页面有回复功能，在回复时，先以本条消息号的 MessgeID 做为查询条件，查找数据库表 message，找到 Sender ID，将其做为此次回复消息接收方的参数，再回复发送者。

5.3.3 应用集成层

应用集成层包含了 Web Service 的产生、注册、发现、查找、SOAP 消息传输层。由于本系统采用 VS.NET2003 作为集成平台，除了 UDDI 和 Web Service 编程，其余工作如 SOAP 消息传输层、Web Service 代理类的产生、发现和绑定，均可由集成平台完成，只需手工少量操作即可，故从简。

项目管理系统项目管理者需要根据专家系统的评审结果来决定立项情况，如何取得评审信息？由于专家系统采用 ASP.NET 实现，故采用 Web Service 结构进行集成，由项目管理系统进行调用。考虑到安全方面的因素，用数据层访问组件对其封装。应用集成层的 Web Service 服务介绍如下：

5.3.3.1 应用集成层的 Web 服务

1．名称：提交评审结果 prof.asmx

功能：提供评审结果

（1）设计返回值为 String 类型的 Web Service，以数据访问组件 DBConn 封装。

```
.....
public string tf(string name)
        {
            string t =DBConn.ExeSql("Select status FROM .v_PassProject
        Where projectname like '%" + name + "%'   " );
            return t;
                    }
.....
```

（2）此服务返回值形式为 String 类型，可以在 TextBox 对其进行显示。

2．名称：项目详细情况 xmj.asmx

功能：评审结果和平均打分

（1）设计返回值为DataSet类型的Web服务，利用SqlDataAdapter填充。

```
.....
  public DataSet xt(String zx)
        {
            String strConn=ConfigurationSettings("DBConnStr");
            SqlConnection cn=new SqlConnection(strConn);
        SqlDataAdapter cd = new SqlDataAdapter("select * from v_Passproject where
```

```
projectrandom like '%" + zx + "%'",cn);
                DataSet ds = new DataSet();
                cd.Fill(ds);
                 return ds;
            }
    ……
```

在项目实践中，同样利用DBConn组件对其进行封装：

```
……
public DataSet df(string zx)
            {
              DataSet t =DBConn.Fille(ref ds,"select * from v_Passproject   where projectrandom
like '%" + zx + "%'");
            return t;
            }
……
```

（2）此服务返回值形式为DataSet型，普通的TextBox等控件不能对其进行显示，为了较清楚地显示结果，客户调用端页面添加DataGrid控件，利用DataGrid绑定功能，接收Web服务传来的DataSet类型的值，呈现给用户，主要代码：

```
……
DataSet ds = new DataSet();
xl.Service1 zda = new xl .Service1();
ds=zdl.xt(txt1.Text);
DataGrid1.DataSource=ds;
DataGrid1.DataBind();
 ……
```

（3）考虑到此控件的显示结果不够美观，设计用表格对接收 Web 服务结果进行动态灵活地展示，效果如图 5—9 所示：

显示界面效果：

项目评审管理	
随机号	21453
项目名称	xxxt
申请人	xxxt
所属单位	北京大学
项目专业	物探
申请金额	10.0000
当前状态	评审通过
开始时间	2006-9-1 0:00:00
结束时间	2007-9-1 0:00:00
平均分	24
项目简介	xxxt

图 5-9 评审结果显示

3．名称：项目评审详细情况服务 xtzx.asmx

功能：提供评审此项目地所有专家各自的评审结论

（1）设计返回值为DataSet类型的Web服务，利用DBConn组件对其进行封装，DBConn组件代码设计见后面数据访问层的介绍。

```
……
public DataSet dh(string zx)
        {
          DataSet t =DBConn.Fille(ref ds,"select * from v_evalproject where projectrandom like
'%" + zx + "%'");
return t;
        }
……
```

（2）服务返回值形式为 DataSet 型，普通的 TextBox 等控件不能对其进行显示，为了较清楚地显示结果，客户调用端页面添加 DataGrid 控件，利用 DataGrid 绑定功能，接收 Web 服务传来的 DataSet 类型的值，也可以呈现给用户。考虑到此控件的显示结果不够美观，设计用表格对接收 Web 服务结果进行动态灵活地展示，主要实现代码类似 Web 服务 4，限于篇幅从略。

4．名称：项目评审详细情况服务 jc.asmx

功能：提供特定专家具体的评审意见

（1）设计返回值为DataSet类型的Web Service，利用SqlDataAdapter填充，此Web服务为多参数类型，以DBConn组件对其进行封装，主要代码如下：

```
……
public DataSet xj (string zx, string lt)
        {
          DataSet t= DBConn.Fille(ref ds, "SELECT sn_name, notion, num, SNMemo,
SNMaxNum from V_evalProjdetail where projectrandom like '%" + zx + "%' and profid like '%" + lt +
"%'");
return t;
        }
……
```

（2）考虑到 DataGrid 控件的显示结果不够美观，设计用表格对接收 Web 服务结果进行动态灵活地展示，设计以流程图 5－10 表达如下：

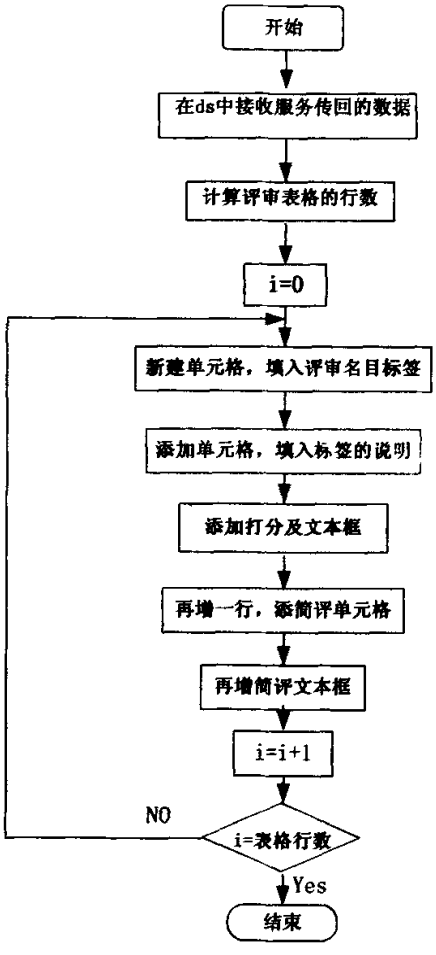

图 5-10 表格展示的流程图

5.3.3.2 安全问题

本文提出一种利用 SOAP 头条目解决 Web Service 安全问题的设计。

1．SOAP 结构

SOAP 是 Web 服务交换信息的标准协议。SOAP 利用 XML 来封装信息。对于 Web 服务来说，SOAP 通过 XML 格式的信息传递参数，进行 Web 调用。

无论是传递数据还是执行远程方法调用，SOAP 必须有一个统一的格式，即 SOAP 首先必须具有 XML 格式，并且由 Schema 规则定义其所包含的标记。总体上看，SOAP 消息主要包括以下 3 个主要元素：

（1）SOAP<Envelope>：它是整个 SOAP 消息的根元素，也是每个 SOAP 消息中必须有的元素，其他元素都在这个元素内部。

（2）SOAP<Header>：<Header>元素是 SOAP 消息中的可选元素，也就是说不是每个 SOAP 消息中都必须有<Header>元素。但如果有，必须是<Envelope>的第一个直接子元素。<Header>元素中可以包括多个头条目子元素。

（3）SOAP<Body>：这是每个 SOAP 消息中都必须有的元素，而且是<Envelope>元素的直接子元素。如果 Envelope 消息中没有<Header>元素，那么这个元素必须是<Envelope>元素的第一个子元素，否则它必须紧接着<Header>元素。

2．定义和处理 SOAP 头条目

Web 服务允许定义和处理 SOAP 头条目。定义 SOAP 头条目是通过继承 SoapHeader 类实现的。

（1）定义 SOAP 头条目

创建一个继承 SoapHeader 类作为头条目内容的元素，应该使用声明为公开成员的方法。例如：下面的类继承 SoapHeader，其中声明了 Usemame 和 Password 两个成员。

```
public class AuthHeaderCS：SoapHeader{
 public string Username;
 public string Password ；
}
```

（2）处理 SOAP 头条目

定义的 SOAP 头条目，可以在 Web 服务中使用。使用 SOAP 头条目分为如下 3 步：

a．在 Web 服务类中声明一个代表 SOAP 头条目类的变量。比如：

```
[Web Service(Description="Simple sample to demonstrate use of SOAP Headers")]
public class HeaderService{
public AuthHeaderCS sHeader;
```

b．在每个 Web 服务方法上应用 SoapHeader 特性。

```
[WebMethod(Description:"This method requires a custom soap header set by the caller")]
[SoapHeader(" sHeader" )]
public string SecureMethod( ) {
}
```

c．在每个应用了 SoapHeader 特性的 Web 服务方法中访问变量中成员。

```
public string SecureMethod( ) {
```

```
if(sHeader=: null)
return"ERROR: Please supply credentials" ;
string usr=sHeader.Username;
string pwd=sHeader.Password;
                    }
```

3. 实际应用

在实际使用中，为了加强 Web Service 安全，对使用 Web 服务的客户进行身份验证可以采用如下做法：

（1）定义 SOAP 头条目

（2）编写身份验证 Web 服务，在 Web 服务方法上应用 SoapHeader 特性。这个 Web 服务查询所在服务器数据库验证身份的合法性。

（3）在客户实际使用的 Web Service 中，添加验证 Web 服务引用，并且在调用端对头条目变量进行赋值。客户在调用实际 Web Service 时，即可自动利用 SOAP 进行身份验证。利用这种方法，在多个 Web Service 进行身份验证时，直接对身份验证 Web 服务加以引用，省去很多编程的麻烦。

5.3.3.3 UDDI 注册中心的实现

本文的 Web Service,从实施的简便快捷出发，并没有使用 UDDI 注册，其代理类的产生采用静态绑定机制，由系统自动完成查找和服务实现，考虑到某些企业可能要用到 UDDI，在此，对 UDDI 的实现做一些讨论。

UDDI 分为公有 UDDI 和私有 UDDI 两种，考虑到实际使用中的效率，一般企业往往不使用微软的公有 UDDI，而是组建本企业私有 UDDI。

私有 UDDI 实现，不用设计的过于复杂，应该知道，UDDI 的目的一是注册 Web 服务，二是使大家了解所注册发布的 Web 服务的功能和用法，以及对应 Web 服务的地址所在，所以本文提出组建 UDDI 的一种实用方法：即使用任何一种网站建设语言，编写一个 Web 应用，提供表单进行 Web 服务登记，添加相关的 Web 服务公布网页，提供 Web 服务的功能，用法，对应地址即可以进行测试。这样一种 Web 应用是较为简单和易于编程实现的，这样的私有 UDDI 实用且易于组建。

5.3.4 数据访问层

数据访问层，设计数据访问组件 DBConn，加强网页安全，并且把所有需要访问数据库的网页中数据库的访问部分抽象集中到数据访问组件中，在网页访问的相关部分，只需传入 SQLString 参数即可，在很多网页都需要对数据库进行访问时，这样做简化了编程，大大减少了工作量，并且便于以后的修改和功能扩展。

相关的数据库访问组件 DBConn 代码：

```
.....
public class DBConn
```

```
{
    private static string connstring ;
    private static System.Data.SqlClient.SqlConnection myConn;
    private static string logfile=@"c:\profdb.log";
    private static string path=@"c:\dbconn.txt";
    private DBConn()
    {
    }
    static private void InitialDB()
    {
        try
        {
            StreamReader xStream;
            string path;
            if(connstring=="")
            {
                path=@"c:\dbconn.txt";
                logfile=@"c:\profdb.log";
                xStream=File.OpenText(path);
                connstring=xStream.ReadLine();
                xStream.Close();
            }
        }
        catch
        {
            DBConn.WriteLog("读取数据库连接字符串出错");
        }
    }
    static public void InitialDB(string FilePath)
    {
        try
        {
            StreamReader xStream;
            logfile=FilePath + "profDB.log";
            xStream=File.OpenText(FilePath + "myconn.txt");
            connstring=xStream.ReadLine();
            xStream.Close();
            }
        catch
        {
            DBConn.WriteLog("读取数据库连接字符串出错");
        }
    }
    static public SqlConnection Instance()
```

```
        {
            if (myConn==null)
            {
                InitialDB();
                myConn=new SqlConnection(connstring);
            }
            if(myConn.State==0)
            {
                myConn.Open();
            }
            return myConn;
        }

        static public string ExeSqll(string strSql)
        {
            System.Data.SqlClient.SqlConnection sqlconn;
                    sqlconn=Instance();
                    System.Data.SqlClient.SqlCommand sqlcomm;
                    sqlcomm=sqlconn.CreateCommand();
                    sqlcomm.CommandText=strSql;
                    string d = (string)sqlcomm.ExecuteScalar();
                    return d;
        }
static public DataSet Fille(ref System.Data.DataSet ds,string sqlString)
        {
            try
            {
                System.Data.SqlClient.SqlConnection sqlconn;
                    sqlconn=Instance();
                    System.Data.SqlClient.SqlCommand sqlcomm;
                    sqlcomm=sqlconn.CreateCommand();
                    sqlcomm.CommandText=sqlString;
                     System.Data.SqlClient.SqlDataAdapter sqladapter=new
                 System.Data.SqlClient.SqlDataAdapter();
                    sqladapter.SelectCommand=sqlcomm;
                    sqladapter.Fill(ds);
                    return ds;
                }
            catch(System.Exception ee)
            {
                throw new System.Exception("填充数据不成功(" + sqlString + ")" + ee.Message);
            }
        }
……//省略了一些函数代码
```

}

5.3.5 数据层

5.3.5.1　与集成设计密切相关的主要数据库表

由于专家系统后于项目管理系统建立，并且在集成系统中处于辅助的地位，所以，在本次集成中，集成的主要任务放在对专家系统的改造上。专家系统数据层主要的数据表有：Profuser 专家基本信息表，Dept 单位信息表，Spec 专业信息表，tech 职称表，Degree 学历表，Project 项目评审表，EvalProject 项目评审人员结论表，Evalprojdetail 评审细节表， SNName 评审内容参考表，技术实现方面的权限模板表 template,菜单表 menu 等。与评审紧密相关的主要实体的 E-R 图表示如图 5－11 所示：（一些实体的属性做了省略。）

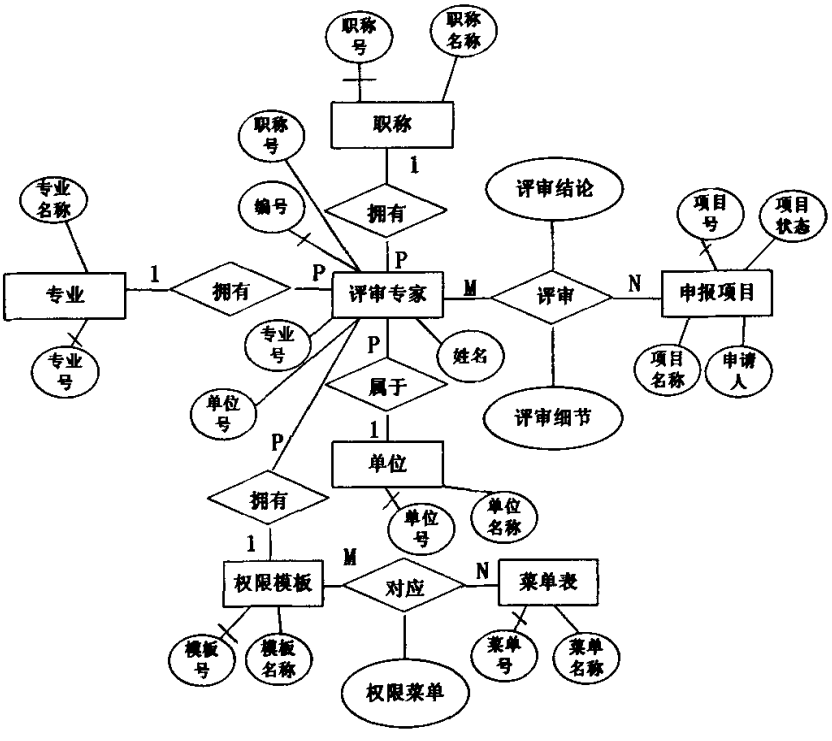

图 5－11 专家评审 E-R 图

其中以下四个表项与集成关系最为密切：

（1）Project 项目评审表：这是有关评审项目的主要记录表，记录专家系统中评审的所有项目的各项信息，包括项目编号，项目名称，项目申报人，项目申报单位，项目简介等，项目申请金额等，设计结构如表 5－1 所示。

表 5—1 Project 项目评审表

说明	列名	类型	长度	是否为空(0 非)
项目编号	projectID	int	4	0
项目名称	projectname	varchar	100	0
项目申报人	projectAuther	varchar	16	0
项目单位	projectdept	int	4	0
项目简介	projectprofile	varchar	600	1
开始时间	startdate	datetime	8	1
结束时间	enddate	datetime	8	1
状态	status	char	1	0
项目申请金额	projectMoney	money	8	0

（2）Profuser 专家基本信息表：用来存放专家系统中所有评审专家的编号，专家姓名，密码，对应的权限模板权限，所属单位，评审的项目总数，个人信息等。设计结构如表 5—2 所示。

表 5—2 Profuser 专家基本信息表

说明	列名	类型	长度	是否为空(0 非)
专家编号	profid	int	4	0
专家姓名	profname	varchar	16	0
专家密码	profpwd	char	20	0
所属单位	detpid	int	4	0
性别	sex	bit	1	0
出生日期	birthday	datetime	8	0
专业编号	specid	int	4	0
最高学历	maxdegree	int	4	0
毕业学校	school	varchar	50	0
手机	mobile	char	12	1
电话	telphone	varchar	18	1
住址	address	varchar	50	0
专家权限	templateid	int	4	0
电子邮件	email	varchar	50	1
评审项目总数	evaluatenum	int	4	0

（3）EvalProject 项目评审人员结论表：用来联系专家与项目的关系，对应上述 E-R 图的评审结论联系，记录特定项目各位专家的打分，和对项目的总评，设计结构如表 5—3 所示。

表 5—3 EvalProject 项目评审人员表

说明	列名	类型	长度	是否为空(0 非)
项目编号	projectid	int	4	0
专家编号	profID	int	4	0
专家总评	totalnotion	varchar	600	1
总体打分	totalNUM	int	4	0
评审时间	evaluatedate	datetime	8	0

（4）Evalprojdetail 评审细节表：用来联系特定专家与项目的关系，对应上述 E-R 图的评审细节联系，记录对于某一项目，特定专家对于项目各项指标的具体打分和评审情况，设计结构如表 5－4 所示。

表 5－4 Evalprojdetai 项目评审细节表

说明	列名	类型	长度	是否为空(0非)
项目编号	projectid	int	4	0
专家编号	profid	int	4	0
分项序号	sn	int	4	0
分项分数	num	int	4	0
分项评价	notion	varchar	600	1
分项名称	sn_name	varchar	40	0

5.3.5.2 项目申报信息的集成

1．专家评审系统项目状态分为以下几种：

申请：尚未进入专家评审系统的 申请项目，该项目在项目管理系统中存在，但是在评审系统中不存在的。

初定评审：已经确定要进行专家评审，但是还没有指定由那些专家进行评审。

评审中：已经确定评审专家，等待评审专家进行评审。

评审完毕：所有的专家已经评审完毕，最后一个专家对项目评审完毕。

不可操作：对于除其他状态以外的状态。

评审通过：经过评审，评审通过。

评审拒绝：经过评审，项目被拒绝立项。

项目管理系统要进行系统对接，需要提供表 5－5 的表项：

表 5－5 项目管理系统需提供的表项

说明	列名	类型	长度	是否为空(0 非)
项目编号	projectID	int	4	0
项目名称	projectname	varchar	100	0
项目申报人	projectAuther	varchar	16	0
项目单位	projectdept	Varchar	30	0
项目专业	ProjectSpec	Varchar	20	0
项目简介	projectprofile	varchar	600	1
开始时间	startdate	datetime	8	1
结束时间	enddate	datetime	8	1
状态	status	char	1	0
项目申请金额	projectMoney	money	8	0

该表提供了专家评审系统必须的项目管理内容，专家评审系统采用视图 V_soureprojec 和 V_allProject 来进行对接。另外，为了保持项目单位和专业的统一性，评审系统的单位名称应该同原项目管理系统保持一致。

实现代码 1：

```
CREATE VIEW dbo.v_soureproject
AS
SELECT t1.项目 ID AS projectID, t1.项目名称 AS projectname, t3.姓名 AS projectAuther,
      t4.单位全称 AS projectdept, t2.专业名称 AS ProjectSpec, t1.项目简介 AS projectprofile,
      t1.起始日期 AS startdate, t1.终止日期 AS enddate, t1.审批状态 AS status,
      t1.申请拨款 AS projectMoney, t5.附件地址
FROM NewCxDB.dbo.项目主表 t1 INNER JOIN
      NewCxDB.dbo.专业编码表 t2 ON t1.专业编码 = t2.专业 ID INNER JOIN
      NewCxDB.dbo.个人信息表 t3 ON t1.负责人 = t3.个人 ID INNER JOIN
      NewCxDB.dbo.单位信息表 t4 ON t4.单位 ID = t1.承担单位 INNER JOIN
      NewCxDB.dbo.项目附件表 t5 ON t1.项目 ID = t5.项目 ID
WHERE (t5.附件类别 = '申请书') AND (t1.审批状态 = '9')
```

实现代码 2：

```
CREATE VIEW dbo.v_allProject
AS
SELECT t1.projectID, t1.projectname, t1.projectAuther, t1.projectdept, t1.ProjectSpec,
      t1.projectprofile, t1.startdate, t1.enddate, CASE WHEN t1.status = 9 OR
      t2.projectid IS NULL THEN '申请' ELSE t2.status END AS status, t1.projectMoney,
      t1.附件地址
FROM v_soureproject t1 LEFT JOIN
      v_project t2 ON t1.projectid = t2.projectid
```

5.3.6 关于远程机器集成的几个问题

由于本文集成的系统为 B/S 模式，利用 IE 浏览器使用，故在一台机器上集成不同的语言即可达到需求。考虑到某些非 B/S 模式以及服务器负担较重的情况下，需要作成分布式系统，下面对分布式情况下的集成的几个问题进行讨论。

1．Web Service 的远程功能的应用

由于 VS．NET 平台的优良性能，调用远程 Web Service，跟调用本地 Web Service 并无多大差别，代理类都是自动生成，只需手工静态绑定或实现动态绑定即可，不多叙述。

2．远程机器数据层问题的解决

（1）安装配置活动目录。

安装 Active Directory,进行一系列复杂配置。安装配置完毕后，在管理工具中提供了 3 个工具：Active Directory 用户和计算机，Active Directory 域和信任关系以及 Active Directory 站点和服务。另外，系统还提供了 Active Directory Schema 和 ADSI，主要用于 Active Directory 开发的工具。Active Directory 域和信任关系以及 Active Directory 站点和服务工具主要用于管理多个服务器或多个域之间的关系，Active Directory 用户和计算机工具是配置互动目录的工具。

（2）安装配置 DNS。

安装 DNS 组件后，打开开始－程序－管理工具－DNS 菜单项，打开 DNS 控制台。

在 DNS 管理器窗口，展开控制台树，展开所有目录，直至看到“正向搜索区域”和“反向搜索区域”。右击正向搜索区域，选择新建区域，选择标准主要区域，按照提示完成设置。“反向搜索区域”。建立与此类似。

右击所建区域，选择新建主机，输入主机名称，IP 地址，完成域名和主机地址映射，按照类似步骤，添加网内其他主机并设置 DNS 服务器动态更新，以及对服务器转发程序进行设置。

（3）配置域方式局域网控制器，创建域用户帐户和计算机帐户。

在完成将 WINDOWS2000 Server 服务器升级为域控制器并进行相关配置后，为了使其他计算机登录到域中，必须为其创建对应的帐户。

单击开始，开始－程序－管理工具－Active Directory 用户和计算机，打开“Active Directory 用户和计算机”窗口。在左窗格“树”列表区单击 User,选择“操作”/“新建”/“用户”,进行用户名称和密码等设置，完成域用户帐户创建。

域中的计算机，要能登录，除了具有域名，还必须具有相应的计算机帐户。在上面的左窗格“树”列表区单击 Computers，“操作”/“新建”/“用户”，创建域用户登录所需的计算机帐户。

（4）工作站计算机加入域方式的局域网。

用鼠标右击“我的电脑”图标，从快捷菜单中选择“属性”。

在打开的“系统特性”对话框中，打开“网络标识”选项卡，单击“网络 ID”按钮，弹出“网络标识向导”。

选中“本机是商业网络一部分，用它连接到其他计算机”，下一步，选中“公司使用带有域的网络”，在弹出对话框，输入域帐户、密码、域管理员帐户、密码等，并对该用户操作计算机的权限予以设置。设置后重启计算机，就可以选择以域用户进行登录。

（5）应用数据库的高级复制订阅功能(以 SQLSERVER 2000 为例)实现远程计算机的数据信息复制。

以域用户身份安装 SQLSERVER 2000 数据库，并对需要复制的相关计算机，进行注册登记。

在企业管理器中，展开服务器组中的一个服务器，选中“复制”，右击，单击“配置发布、订阅服务器和分发”选项菜单，进入“配置发布、和分发向导”界面。选择分发服务器，创建分发数据库 distrubution。进行发布服务器，订阅服务器等设置。

在企业管理器中，单击发布服务器－复制，右击发布内容，选择“新建发布”，启动“创建发布向导”，进行发布模板、发布内容、发布代理、订阅类型等一系列设置。

在企业管理器中，单击订阅服务器－复制，右击订阅内容，选择“新建订阅”，启动“创建订阅向导”，进行订阅类型、订阅模板、订阅内容、订阅代理等一系列选择设置。

在企业管理器中，单击发布服务器－复制，右击发布内容，选择“属性”，在“属性”

框中，选择“状态”选项卡，对“调度”属性进行编辑，并启动调度代理运行。

经过以上各种配置，远程计算机上数据就能够自动复制到本地计算机上相应的数据库中，并能够做到实时更新(同步)。两地数据库上的操作，相当于本地计算机上两个不同数据库的操作，至此，远程计算机数据层的问题已经解决。

5.4 其他用到的一些关键技术

5.4.1 用户权限管理分配模板的建立

首先在数据库表中 template 中根据模板号分配其对应的各项子菜单；然后把所有主菜单的各项参数在 DataSet 中列为一表，对每个主菜单下的所有子菜单按照在此权限模板的有无情况进行分类，以 0，1 表示；接着，按照主菜单号，过滤视图为每个主菜单号对应的所有子菜单号；并设计网页后台代码，对所有主菜单只显示一次，对子菜单采用循环显示。具体设计流程如图 5－12 所示：

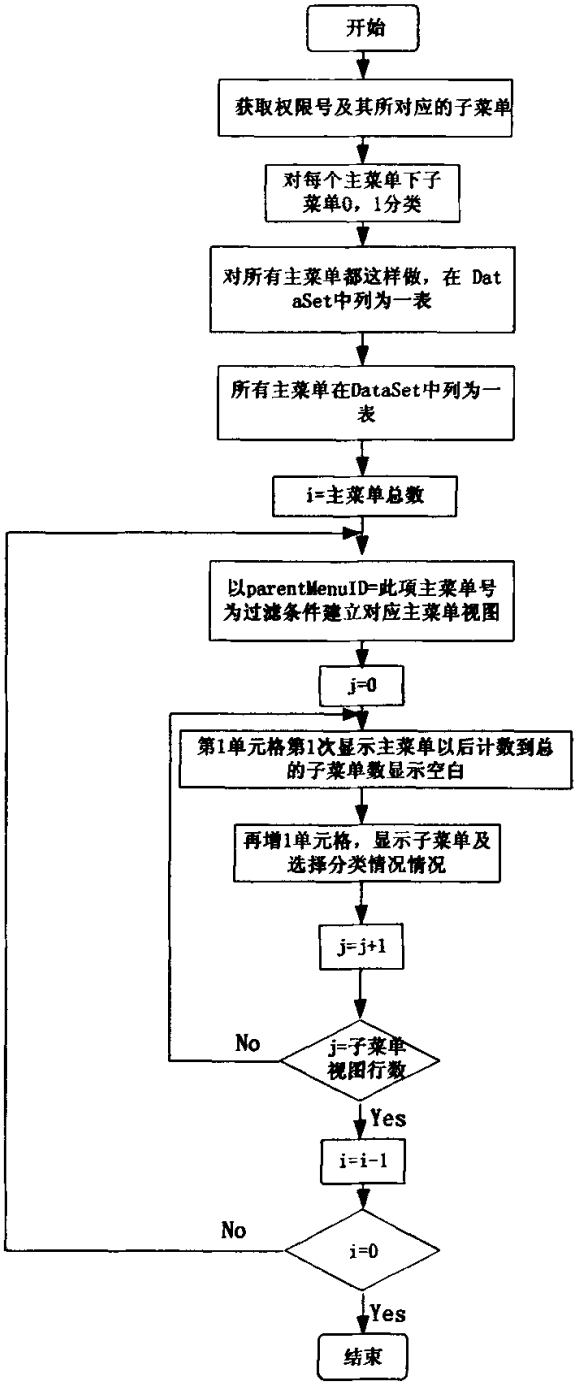

图 5－12 权限模板建立流程图

5.4.2 分页技术

构造页面变更函数，建立页面选择按钮和实际显示页面的映射联系。获得实际页码号后，传给 ShowData2Html(CurrPage)页面显示函数。页面变更函数设计：首先获得当

前页面数，针对向前、向后、第一页、最后一页、下一页、跳转六种情况，对 CurrPage 变量重新赋值，在调用页面显示函数进行实际显示。

页面显示函数 ShowData2Html(CurrPage)用传入的 CurrPage 号替换函数中 showPage 变量，计算最大页面数，计算每页显示的起始行和结束行，并对应到从数据库取到 DataSet 中内存表的实际行，予以显示。

```
this.Total_page.Text=Convert.ToInt32(ds.Tables["maintable"].RoWS.Count  /10.0+0.495).ToString();
//计算总的页面数
                    if (showPage>int.Parse(this.Total_page.Text))
showPage=int.Parse(this.Total_page.Text); //防止显示页面大于最大页面
                    this.Current_page.Text=showPage.ToString();
                    int startrow,endrow;
                    startrow=(showPage-1)*10;
      endrow=(showPage*10)>ds.Tables["maintable"].RoWS.Count?ds.Tables["maintable"].RoWS.C
ount-1:showPage*10-1;//>则取 ds.Tables["maintable"].RoWS.Count-1，否则，则取 showPage*10-1
                    for(i=startrow;i<=endrow;i++)
                   {
                    ...../实际显示的用户代码
                     }
```

5.4.3 菜单控件的应用

（1） 下载安装 MSPlus.Web.UI.WebControls 制作控件，添加引用到项目，菜单所需的各项数据有 MenuItem ID="MI234 ， Label=" 添加 Web 服务 " LeftIcon="icon_fileSearch"

MenuItem Url="alert('Client Event MACK')"等。

（2）在 popedomtemplate 表中写 templateID,menuID 是关键,接着在 menu 表中填写菜"MenuID,LeftIcon,Url,LeftIconOver 等项，在显示页面前台添加控件，后台添加对控件的使用。

（3）编写 DyniMenu 类，结合登录权限，动态产生菜单，其中 Url 一项为点菜单时，菜单所指的实际页面，DyniMenu 类设计流程图表达如图 5－13 所示：

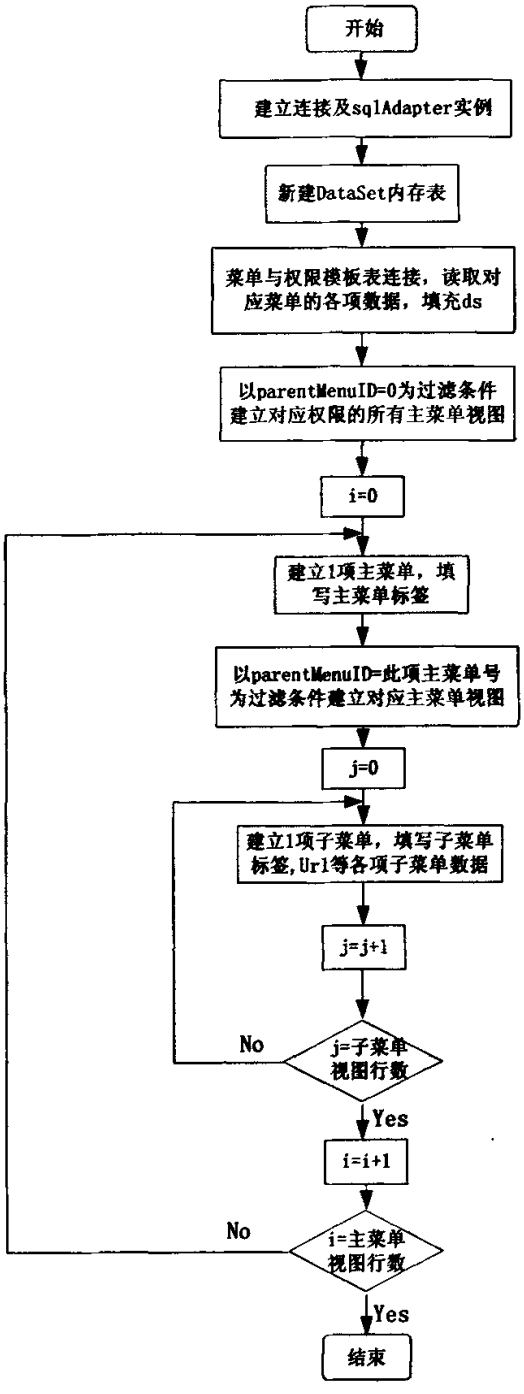

图 5－13 动态菜单类设计流程图

4. 在数据库表 TableList 中填写 ListID、ListName、ListSQL、ListUrl、EditUrl 等与页面显示内容相关的各项参数；在 LIST 表中填写 ListID，Listname 并结合页面控制后台代码以及 CSS 等以形成实际页面。其中每一页都有对 DyniMenu 类的动态调用从而

生成所需的菜单界面。

5.5 本章小结

本章结合具体的应用实例—中石油股份公司基金项目管理系统和专家评审系统，分析了如何在实际的企业应用中利用 Web Service 技术实现 SOA。本章中介绍了应用的背景、分析了企业的具体需求和业务流程。根据第四章提出的基于 SOA 的企业应用模型，在.NET 下实践了 SOA。

使用效果主要界面

说明：

对项目管理的系统人员登录界面增加单一登录模块接口；增加项目入库模块；增加专家系统附件信息集成模块；对项目管理人员增加调用专家意见接口；增加专家系统网上信息平台模块；开发数据访问组件 DBConn，开发权限模板模块；开发项目评审情况四个 Web 服务。

图 5—14 项目管理人员登录界面（增加单一登录模块接口）

图 5—15 项目入库界面

图 5—16 立项管理界面（增加调用专家意见接口）

图 5—17 单一登录模块界面

图 5—18 附件下载界面

图 5—19 短信功能主要界面

图 5—20 权限模板主要界面

图 5－21 项目的所有专家评分结论情况服务

图 5－22 特定专家对项目的详细评审结论服务

图 5—23 活动目录图

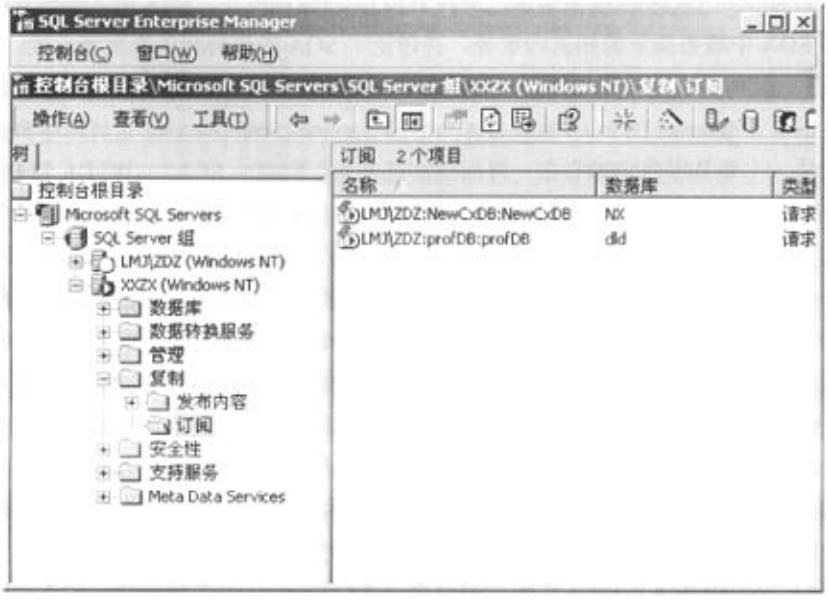

图 5—24 复制订阅图

第六章 总结与展望

6.1 论文研究总结

分布式计算将网络上分布的各种软件资源看作是服务，随着分布式计算在技术上(如 XML, Web Service 等)的成熟，面向服务的 SOA 体系架构成为解决应用集成、提高软件重用和加快应用实施的一种必然思路。

SOA 为企业应用开发提供了一种松耦合的、互操作性强，并且具有良好可扩展性的架构思想。通过良好设计的服务，可以迅速构建出来灵活而强大的基于面向服务体系结构的应用。借助于这种 SOA 架构设计思想，设计和集成企业应用系统变得更加简单，它将成为未来企业应用系统设计的主宰思想。Web Service 是 SOA 在 Internet 环境下的实现技术，解决了传统的分布式计算中存在的许多问题。本文的工作主要体现在以下几个方面：

（1）系统分析了目前构建企业应用系统中存在的一些问题，比如：迅速增长的企业需求与目前软件开发水平之间的矛盾，不同时期采用不同技术开发的新的应用系统独立使用给企业带来的许多不便，已经存在企业应用的可扩展性和互操作性差的问题等等。对企业应用集成 EAI 传统的实现技术进行分析研究，分析了面向服务的软件体系结构出现的必然性，及其特点和优势。

（2）介绍了 SOA 的理论知识，分析了 SOA 的编程模型、SOA 中各个角色的职责功能和 SOA 中服务质量的相关因素等，并讨论了 SOA 的优势和未来发展前景。

（3）SOA 是一种设计思想，具体实现要依靠实际的技术。本文深入研究了目前比较流行的基于 Internet 的 SOA 实现技术－Web Service。Web Service 之所以得到业界的广泛关注，主要是因为它建立在一组标准协议之上：XML、SOAP、WSDL 和 UDDI。并将 Web Service 技术与 SOA 现有的其他实现技术如 CORBA，DCOM，等进行了分析比较。更加突出 Web Service 的与平台无关、易于扩展、易于集成等特点。

（4）本文总结了 SOA 的设计原则，并在此基础上，结合了经典的面向对象的分析与设计方法和传统的多层软件体系结构思想，综合运用 DCOM，Web Service 等技术，研究提出了基于 SOA 的企业应用集成框架的一般模型。该模型 B/S，C/S 模式相结合，将 SOA 架构分为客户访问层、企业应用层、企业应用集成层、消息传输层、服务提供与封装层、数据访问层。并对每层的作用和实现技术进行了详细的分析论述。另外对框架中的 Web Service 互操作问题进行了较深入的研究讨论，给出了对应的解决办法；对框架中的旧有系统 Web Service 转化问题，给出了 COM 方式的解决方案。

（5）本文的 Web Service 实践主要是在. NET Framework 下进行的，结合实际的案例－中石油股份公司基金项目管理系统和专家评审系统，实现了. NETFramework 下基于 Web Service 技术的 SOA。在案例分析中详细介绍了企业背景，分析了企业的应用需求，并提出了相应的解决方案，按照本文提出的基于 SOA 的企业应用集成框架模型，进行服务设计、实现、发布管理、调用，设计了单一登录等模块，从而实现了整个系统。

并对 Web 服务的安全问题，提出了利用 SOAP 头条目进行验证的解决办法。对于企业私有 UDDI 注册中心的组建，提出了实用，快捷的实现方法。

6.2 进一步的工作

虽然 SOA 被称为在面向对象和基于构件的软件开发之后的下一代技术，它本身还处在不断的发展之中，目前还没有以上两种开发方法成熟，且目前的应用还不广泛。然而，由于 SOA 自身具有的优势符合了未来软件发展的趋势，并随着各大软件供应商在此展开的深入研究和激烈竞争，它必然会成为成熟的下一代主流开发技术而得到广泛支持和应用。

进一步的工作研究重点主要有：

（1）由于实际案例的企业需求及个人水平所限，本文对 SOA 架构中业务流程的事务管理，使用 BPEL 对服务进行组装和编排实现工作流程方面的内容涉及较少[31]。

（2）Web Service 依然是一个正在发展的技术，有关 Web Service 的标准目前还在如 J2EE、.NET 等几大集成平台之间展开争论，有待统一，因而集成框架中的互操作问题与此密切相关。另外，有关 Web 服务的大数据量访问及安全问题需要进一步的研究。

（3）旧有系统的 Web Service 转化是企业应用集成中的一个热点问题，集成框架中提出的 COM 方式 Web Service 转化途径对 VB、VC、Delphi 等许多语言都是适用的，但并不是对所有语言都适用，例如对应 ASP 应用，需要考虑 CORBA 等其他途径进行转化。

Web Service 仍然是正在发展的技术，本文提出的基于 Web Service 的 SOA 企业应用集成框架的性能也在不断的提升当中，框架的设计还有一些不完善的地方有待于进一步改进。在今后的研究中，将继续关注新技术的动态，并将新技术的特点完善地与 SOA 设计思想进行结合，开发出更好的分布式环境下的企业应用。

致 谢

本课题和论文是在我的导师孟东升和余元华副教授的悉心指导和热情鼓励下完成的。从论文的选题、方向的把握，到系统的方案设计以及论文的内容组织，孟老师和余老师都给出重要而富有建设性的建议，使得本论文的研究工作得以顺利完成。在我攻读硕士学位的三年时间里，得到了孟老师无微不至的关怀和教导，孟老师严谨的治学态度，深厚的专业知识、认真务实的治学态度、勤奋的工作作风，给我以深刻的印象。老师在专业知识上的辛勤传授和治学修身上的谆谆教导都将使我受益终生。在此，谨向尊敬的导师致以最衷心的感激和敬意。

在此，我还要衷心感谢方明老师，在我研究生期间给予我学习和科研上的指导和关怀，以及在论文写作过程中给予我诸多有益的建议和帮助。感谢黄晓伟老师，正是在他的研究基础上，才使得我的论文在较高的起点上得以顺利完成。

感谢油气信息系统工程研究中心实验室的全体老师和各位同学，在本论文的研究工作中给我提供的良好的学习和实践环境，以及在学术上的交流和技术上的指导。感谢李言武，蒋志新，纪福全等同学给予作者的友情和无私的帮助。

同时感谢研究生部的老师和领导三年来对作者的关心和培养。感谢计算机学院领导和老师以及行政工作人员在作者学习和生活中提供的各项保障、帮助和指导。

特别感谢作者的家人和朋友们，在作者的多年的学习生涯中，在物质上和精神上给予我的无私帮助，在此感谢他们！没有他们的支持，我就不可能完成自己的学业。

最后向评阅本论文的各位老师致以深深的谢意，感谢他(她)们对作者提出的宝贵意见及进一步研究的技术方向和方法。

参考文献

[1] 杨涛，刘锦德. Web Service 技术综述——一种面向服务的分布式计算模式[J]. 计算机应用，2004，24 (8)：1-4.

[2] 郭春艳. 基于 SOA 企业应用的研究与实现[D]. 大连：大连理工大学，2005

[3] Taylor KL, O'Keefe CM,Colton Jetal. A service-oriented architecture for a health research data network. Proceedings of the 16th International Conference on Scientific and Statistical Database Management, Santorini Island, Greece, 2004:443-444.

[4] DangG, Cheng Z, Jin S et al. A service-oriented architecture fortele-immersion.Proceedings of the 2005 IEEE International. Conference on e-Technology, e-Commerce and e-Service, Hong Kong, China, 2005：646-649.

[5] Wang H, Ghoting A, Buehrer Getal. A service-oriented framework for next generation data analysis centers. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, 2005：219-219.

[6] Herr M, Bath U, Koschel A. Implementation of a service-oriented architecture at deutsche post mail. European Conference on Web Service, Erfurt, Germany, 2004：227-238.

[7] Rabhi F A, Dabous FT, Yu Hetal. A case study in developing Web Service for capital markets. Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service, Taipei, Taiwan, 2004:38-41.

[8] 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, 15 (3) ：428-442.

[9] 乔江等. 基于 Web Service 的 EAI 实现研究[J]. 计算机应用, 2003(11):100-102

[10] 尉飞新，杨德华. 基于 Web Services 的企业应用集成研究[J]. 微计算机应用，2007(3)

[11] 李锦棠. 企业 SOA 服务集成的研究与设计[D]. 广州：广东工业大学，2006

[12] 李磊. CORBA 与 EJB 集成技术的研究[J]. 计算机科学，2001(6)：27－29

[13] 蒋志青. 企业业务流程设计与管理(第二版)[M]. 北京：电子工业出版社，2004

[14] 叶军等. Web Services 在企业信息系统集成中应用架构的研究[J]. 计算机应用研究 2007(06)

[15] 朱金生. 面向服务的企业应用研究与实现[D]. 西安：西北工业大学，2005

[16] 李航. 面向服务的异构分布式计算体系互联技术研究[D]. 长沙：国防科学技术大学,2003-03

[17] 董明忠. 一种 CORBA 中间件的智能入侵检测系统的实现[J]. 计算机与信息技术 2007(4):51－53

[18] Matjaz B.Juric 等著，袁然等译. J2EE EAI 编程指南[M]. 北京:电子工业出版社，2002-09

[19] 博格利. CORBA 技术及其发展[J]. 重庆电力高等专科学校学报 2007(01):15－18

[20] 柴晓路，梁宇奇著. Web Services 技术、架构和和应用[M]. 北京:电子工业出版社. 2003

[21] 郑小平. . NET 精髓——Web 服务原理与开发[M]. 北京:人民邮电出版社. 2002

[22] 袁占亭. 基于 BPEL 和 SOA 的 Web 服务开发研究[J]. 微计算机信息，2006 年第 22 卷：233－235

[23] 刘迎春，兰雨晴，于乐乐. ESB 中的数据交换技术[J]. 计算机系统应用,2005-10

[24] 周晖. 基于 WEB 服务的企业应用集成技术研究[D]. 杭州：浙江大学，2004

[25] 孙永强. WEB 服务深入编程[M]. 北京：清华大学出版社，2002 年第 1 版

[26] 陈传波，张道杰，李涛. 基于 WEB 服务的企业应用集成模型研究[J]. 计算机工程与科学，2004 (12)：35-38

[27] 黄双喜，范玉顺等. 基于 WEB 服务的企业应用集成[J]. 计算机集成制造系统,2003(10)：85-90

[28] 郭东恩，杜传宇. SQL Server 实用基础教程[M] . 北京：航空工业出版社，2005

[29] 赵松涛，吴维元. SQL Server 2000 系统管理实录[M] . 北京：电子工业出版社，2006

[30] 郑阿奇，刘启芬. SQL Server 实用教程[M] . 北京：电子工业出版社，2006 年第 2 版

[31] 冯波. 面向服务的工作流管理系统[D] . 青岛：中国海洋大学，2005

附 录

攻读硕士学位期间发表论文

1．吕鸣剑，孟东升 基于 SOA 实现企业应用集成[J]. 福建电脑 ，2006(9)

2．吕鸣剑，孟东升 基于 J2EE 实现 CMT 事务管理[J]. 现代电子技术，2007(22)，已录用。