申请同济大学工学硕士学位论文

基于“和欣”嵌入式操作系统的 WEB SERVICE 组件的构建

培养单位：电子与信息工程学院

一级学科：计算机科学与技术

二级学科：计算机应用

研 究 生：陆益

指导教师：顾伟楠 教授

二〇〇五年二月

A dissertation submitted to
Tongji University in conformity with the requirements for
the degree of Master of Philosophy

School/Department:
Discipline:
Major:
Candidate: Yi Lu
Supervisor: Prof. Wei-Nan Gu

February, 2005

基于“和欣”嵌入式操作系统的WEB SERVICE组件构建的

陆益

同济大学

学位论文版权使用授权书

本人完全了解同济大学关于收集、保存、使用学位论文的规定，同意如下各项内容：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并采用影印、缩印、扫描、数字化或其它手段保存论文；学校有权提供目录检索以及提供本学位论文全文或者部分的阅览服务；学校有权按有关规定向国家有关部门或者机构送交论文的复印件和电子版；在不以赢利为目的的前提下，学校可以适当复制论文的部分或全部内容用于学术活动。

学位论文作者签名：

年　　月　　日

经指导教师同意，本学位论文属于保密，在　　年解密后适用本授权书。

指导教师签名：　　学位论文作者签名：

年　　月　　日　　年　　月　　日

同济大学学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。本学位论文原创性声明的法律责任由本人承担。

签名：

年　　月　　日

摘　要

随着计算机网络技术和信息技术的发展，计算机软件也经历了单机软件、C/S 模式软件系统、B/S 模式软件系统等各个阶段，软件模型也经历了面向过程、面向对象，以及最近提出的面向方向等各个阶段。其中每一个概念的提出，都是计算机软件史上的一次飞跃，给软件设计、编写、集成、维护以及对客户需求的满足都带来了极大的方便。

但是在家电高度智能化的今天，仅仅对传统的计算机进行软件集成是不够的，软件集成必须考虑到各种嵌入式设备。如何让各种手机、PDA 使用原先只有传统计算机才能实现的服务，如何将电视机、微波炉等家电无缝集成到分布式软件中来而不再需要驱动，这些都对计算机软件体系提出了新的挑战。

本文介绍了笔者在上篇论文中提出的面向服务的软件模型，和国家 863 重点软件项目“和欣”嵌入式操作系统及其 EZCOM 的主要思想和现在的软件集成技术，分析了 COM、CORBA 和 WEB SERVICE 技术的优劣，得出了 WEB SERVICE 将成为下一代软件系统集成的主要技术的观点。然后详细介绍了 WEB SERVICE 的 UDDI、WSDL 和 SOAP 协议的主要内容。并且在该 863 项目的支持下，利用“和欣”嵌入式操作系统提供的一系列 API 及其 EZCOM 技术实现了一个在“和欣”嵌入式操作系统上部署和提供 WEB SERVICE 的组件，并且仿照一些 OPEN SOURCE 的项目中有关 WEB SERVICE 的 API 提供了一系列在“和欣”嵌入式操作系统上处理 WEB SERVICE 的 API，从而利用“和欣”嵌入式操作系统让嵌入式设备有了直接提供 WEB SERVICE 的能力，再利用面向服务的软件模型，在一定程度上解决了将嵌入式设备用最小的代价动态集成到分布式软件中来的问题。最后展望了“和欣”嵌入式操作系统及其 WEB SERVICE 组件在我国今后的生产生活中将起到的重大作用。

关键词：“和欣”嵌入式操作系统；面向服务；WEB SERVICE

ABSTRACT

随着计算机网络技术和信息技术的发展，计算机软件也经历了单机软件、C/S 模式软件系统、B/S 模式软件系统等各个阶段，软件模型也经历了面向过程、面向对象，以及最近提出的面向方向等各个阶段。其中每一个概念的提出，都是计算机软件史上的一次飞跃，给软件设计、编写、集成、维护以及对客户需求的满足都带来了极大的方便。

但是在家电高度智能化的今天，仅仅对传统的计算机进行软件集成是不够的，软件集成必须考虑到各种嵌入式设备。如何让各种手机、PDA 使用原先只有传统计算机才能实现的服务，如何将电视机、微波炉等家电无缝集成到分布式软件中来而不再需要驱动，这些都对计算机软件体系提出了新的挑战。

本文介绍了笔者在上篇论文中提出的面向服务的软件模型，和国家 863 重点软件项目“和欣”嵌入式操作系统及其 EZCOM 的主要思想和现在的软件集成技术，分析了 COM、CORBA 和 WEB SERVICE 技术的优劣，得出了 WEB SERVICE 将成为下一代软件系统集成的主要技术的观点。然后详细介绍了 WEB SERVICE 的 UDDI、WSDL 和 SOAP 协议的主要内容。并且在该 863 项目的支持下，利用“和欣”嵌入式操作系统提供的一系列 API 及其 EZCOM 技术实现了一个在“和欣”嵌入式操作系统上部署和提供 WEB SERVICE 的组件，并且仿照一些 OPEN SOURCE 的项目中有关 WEB SERVICE 的 API 提供了一系列在“和欣”嵌入式操作系统上处理 WEB SERVICE 的 API，从而利用“和欣”嵌入式操作系统让嵌入式设备有了直接提供 WEB SERVICE 的能力，再利用面向服务的软件模型，在一定程度上解决了将嵌入式设备用最小的代价动态集成到分布式软件中来的问题。最后展望了“和欣”嵌入式操作系统及其 WEB SERVICE 组件在我国今后的生产生活中将起到的重大作用。

关键词：“和欣”嵌入式操作系统；面向服务；WEB SERVICE

目录

第一章　引言

1．1 计算机软件模型变迁

1．1．1 面向过程开发方法

面向过程(Procedure-Oriented)的开发方法是由 Yourdon 和 Constantine 提出的，即所谓的 SASD 方法，也叫做面向功能的软件开发方法或面向数据流的软件开发方法。面向过程开发方法是一种“功能性程序设计”的开发方法，针对特定问题，根据所需功能，制定特定方法进行的结构化的软件开发方法。Yourdon 方法是 80 年代使用最广泛的软件开发方法。它首先用结构化分析(SA)对软件进行需求分析，然后用结构化设计(SD)方法进行总体设计，最后是结构化编程(SP).

面向过程开发方法接近于计算机世界的物理实现，将程序分离为数据结构和相应的算法，以变量、函数和过程等单元来对客观世界进行描述，实现了一定程度上的模块化，关注点分离程度不高。

1．1．2 面向对象开发方法

面向对象(Object-Oriented)开发方法是随着 OOP(面向对象编程)向 OOD(面向对象设计)和 OOA(面向对象分析)的发展而形成的。面向对象对面向过程进行了进一步的抽象提高，采用了类和对象的概念，把变量以及对变量进行操作的函数和过程封装在一起，用这种更高一级的抽象来表达客观世界。

面向对象方法学的出发点和基本原则，是尽可能模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界解决问题的方法与过程，也就是使描述问题的问题空间与实现解法的解空间在结构上尽可能一致，向着语义断层的方向迈了一大步。面向对象开发方法正是由于符合了人类的思维习惯，逐渐成为当今主流的软件开发方法。

在面向对象的观点中，客观世界是由对象组成的，任何事物都是对象;并且所有对象都划分成各种对象类(简称为类)，每个类定义了一组数据和方法:按照子

类与父类的关系把若干个类组成一个层次结构的系统(继承);对象彼此之间只能通过传递消息互相联系。综上所述，面向对象的方法可以用上述的四个方面来概括:面向对象=对象+类+继承+消息通信。面向对象由于采用类和对象对数据和行为进行了封装，达到了功能和数据的高度统一，封装、继承、内聚等概念进一步丰富了系统的模块化。获得了较好的功能关注点分离能力。

1．1．3 面向方面模型

面向方面编程 (AOP) 是施乐公司帕洛阿尔托研究中心 (Xerox PARC) 在上世纪 90 年代发明的一种编程范式，它使开发人员可以更好地将本不该彼此纠缠在一起的任务（例如数学运算和异常处理）分离开来。

AOP 为开发者提供了一种描述横切关注点的机制，通过划分方面代码来提取横切关注点，并能自动将横切关注点织入到软件系统中，从而实现对横切关注点的模块化。

AOP 方法有很多优点。首先，由于操作更为简洁，所以改进了性能。其次，它使程序员可以花费更少的时间重写相同的代码。总之，AOP 能够为不同过程提供更好的封装性，提高未来的互操作性。

1．2 计算机软件体系结构变迁

1．2．1 单机结构

在网络尚未出现或者还未普及之前计算机软件一般都是部署在单机上的。软件的全部功能和数据都放在同一台计算机上，如图 1.1 所示。和这个时期的软件如果有数据库，则相对应的一般是文件性的数据库，比如 dBase 等。而且当时相对而言应用程序还很少有异地交换数据的需求，所以相应的软件功能也较简单。

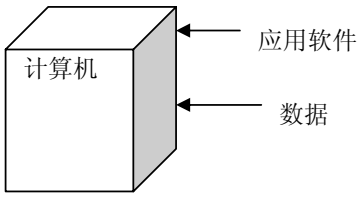


图 1.1 单机结构

1．2．2 主机－终端结构

最初运行在网络上的软件是一种基于主机-终端模式的计算模型，系统中几乎所有的计算都由大型的主机来完成，终端只是单纯作为一种输入输出设备。这种结构从本质上讲和第一种单机结构并无区别。如图 1.2 所示。

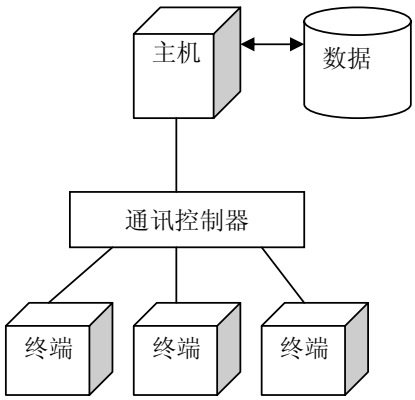


图 1.2 主机－终端结构

1．2．3 客户机－服务器结构

八十年代末，针对主机-终端模式的信息系统的问题和不足，人们提出了如图 1.3 所示的客户-服务器（Client/Server，简称 C/S）结构，由于这种结构比较适于局域网运行环境，所以逐渐得到了广泛的应用。在这种体系结构中，将软件系统分为两大部分：一部分是由多个用户共享的信息与功能，这部分称为服务器部分；另一部分是为每个用户所专有，称为客户部分。客户部分负责执行

前台功能，如管理用户接口、数据处理和报告请求等。而服务器部分执行后台服务，如管理共享外设、控制对共享数据库的操纵、接受并应答客户机的请求等。这种体系结构将一个应用系统分成两大部分，由多台计算机分别执行，使它们有机的结合在一起，协同完成整个系统的应用，从而达到系统中软、硬件资源最大限度的利用。

任何一个应用系统都可分成三部分：显示逻辑部分（表示层），事务处理逻辑部分（功能层）和数据处理逻辑部分（数据层）。由于 C/S 结构被设计成两层模式，显示逻辑和事务处理逻辑部分均被放在客户端，数据处理逻辑和数据库放在服务器端，从而使客户端变得很“胖”。但随着应用系统的大型化以及用户对系统性能要求的不断提高，C/S 结构越来越满足不了需求，主要体现在：

（1）客户机负载过重，成本增加。随着应用系统功能的日益复杂化，客户端应用程序也变得越来越庞大，客户机不堪重负，加重了投资成本。

（2）程序开发量大。由于存在许多不同的客户端都要访问数据库，而且通常用户接口和应用程序都集中在客户端，所以对不同客户端要开发不同的程序，极大的增加了编程的工作量。

（3）系统维护困难。一旦系统程序要进行修改和升级，则需要更新所有客户端的程序，使系统的正常维护升级工作很难进行。

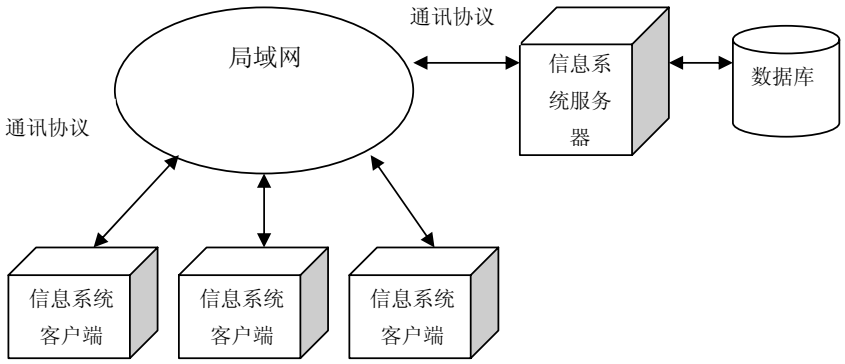


图 1.3 客户－服务器结构

1．2．4 浏览器－服务器结构

现在，由于目前 WWW 上的商业活动激增，同时随着 Internet 的大规模使用和普及，应用程序正在转向利用 Internet 的有关技术。与大型主机规模化的潮流相比，选用 Internet 有较小的破坏性和较大的建设性。换言之，客户机－服务器结构将要被基于 Internet 的体系结构所取代。

基于 Internet 的信息系统的主要结构是以 Web 浏览器、Web 服务器和基于网络的协议（如 TCP/IP 和 HTTP 协议）为基础的，该结构也称为 B/S（Browser/Server）结构（如图 1.4 所示）。和传统的 C/S 结构相比，它具有以下的特点：

客户端较“瘦”。C/S 结构下的客户端具有显示与处理数据的功能，需要在客户端运行庞大的应用程序，因此，客户端很“肥”；而在 B/S 结构中，客户端只运行 Web 浏览器和操作系统。Web 服务器上已完成大多数处理，客户端仅仅是个表现工具。

开放的标准。C/S 所采用的标准往往是内部统一的、专用的，很难扩展且很难与其他的系统交互。而 B/S 结构所采用的标准都是开放的、非专用的，是经过标准化组织所确定而非单一厂商所制定，保证了应用的通用性和跨平台性。

较低的开发和维护成本。对于 C/S 模型，无论是安装、配置还是升级都需要在所有的客户机上实施，极大的浪费了人力和物力。B/S 体系结构能使信息系统实现分布式的面向对象的应用，只需在客户端装通用的浏览器，应用可以由 Web 服务器集中维护和运行，并透明地分布到不同类型的客户端。

统一友好的界面。C/S 应用的用户界面是由客户端软件决定的，其使用方法和界面各不相同，每个系统都要求用户从头学起，难于使用。B/S 应用的界面都统一在浏览器上，易于使用，界面友好。

统一的连接方式。C/S 结构可能需要混合多种传输协议；而 B/S 体系中，所有系统都使用 TCP/IP 和 HTTP 进行通信。

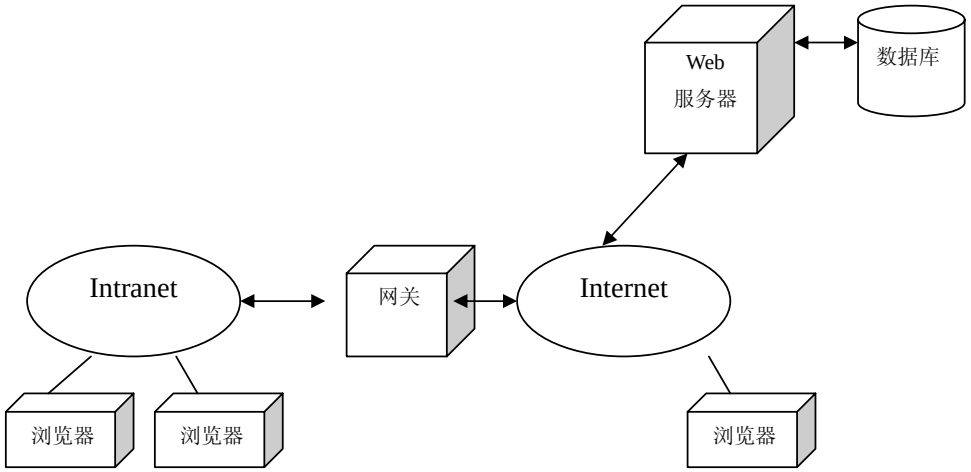


图 1.4 浏览器－服务器结构

1．3 当前无线网络和家电智能化对软件提出的需求

但是在家电高度智能化的今天，仅仅对传统的计算机进行软件集成是不够的，软件集成必须考虑到各种嵌入式设备。如何让各种手机、PDA 使用原先只有传统计算机才能实现的服务，如何将电视机、微波炉等家电无缝集成到分布式软件中来而不再需要驱动，这些都对计算机软件体系提出了新的挑战。

笔者曾经在论文《基于和欣操作系统的面向服务软件模型研究》一文中提出一个面向服务软件模型。

面向服务软件模型主要观点是，软件的模块将不再象传统的面向对象模型那样由提供源代码的类构成，而是由一些遵循一定规范的，向用户界面或其他模块提供服务的二进制组件构成。其中将和用户界面在同一台计算机上执行的服务组件模块，由提供本地进程间服务的组件提供；将和用户界面在同一个局域网内执行的服务组件模块，由提供远程进程服务的组件提供；将和用户界面仅通过公网通讯的模块，则通过 XML Web Service 提供。因为服务组件模块仅通

过一定的规范对用户界面提供服务，这样就可以做到二进制封装、语言无关性，并且可以随时升级、替换，只要能遵循同样的规范，将不影响用户界面的使用。

面向服务模型也是一种和面向过程、面向对象、面向方面不矛盾的软件模型，它的提出更好的解决了软件复用，使得软件工厂化生产能够更好的进行，也使得各种异构的系统能够更好的集成到软件中来。

利用上述的面向服务软件模型，我们更可以构造一个面向服务的软件结构，此结构对 C/S 软件和 B/S 软件进行了一些综合和改进，可以更好的在新的形式下进行软件的整合。

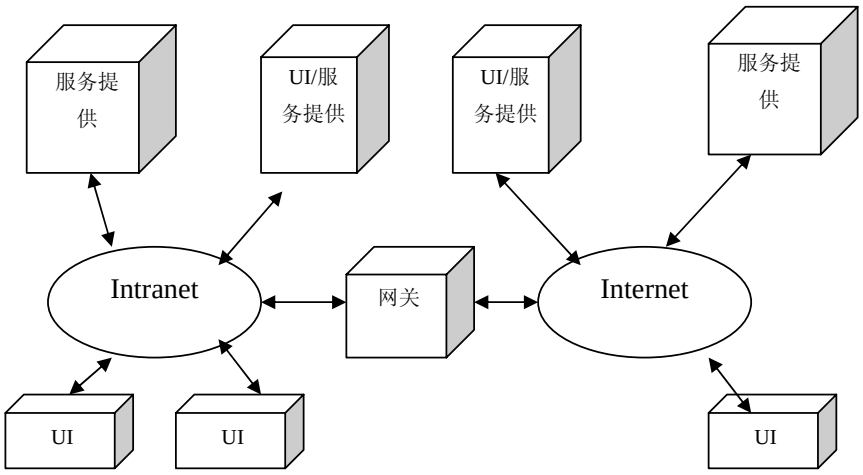


图 1.5 面向服务软件结构

在此模型中，没有严格的区分客户端和服务器，网络上的任何一台计算机或嵌入式设备都可以成为服务的提供者或服务的接受者，并且同过一个 UI（根据需要可能是一个客户端软件，或者是一个标准浏览器），把接受的统一格式的服务转化成人可以接受的界面，进行人机交互。并且无论是嵌入式设备或者传统 PC 还是服务器，只要能够处理统一的服务信息，就可以成为此软件结构中的服务器或者客户端，甚至两者都是。

嵌入式设备大多异构，并且处理能力强弱不同，如何快速的开发、重用在嵌入式设备中的服务处理软件，只能依赖于嵌入式操作系统，只有在嵌入式操作系统上，才能实现快速的软件跨平台移植，甚至“一次编写，到处编译、运行”（提出“一次编写，到处运行”的 JAVA 语言依赖于 JAVA 虚拟机，而 JAVA 虚拟机也必须要跑在操作系统之上）。

第二章 “和欣”嵌入式操作系统

“和欣”嵌入式操作系统是一个完全我国自主研发，完全自主知识产权的国际先进的嵌入式操作系统，在国内是最适合嵌入式设备搭建面向服务软件结构基础的嵌入式操作系统。以下详细介绍了“和欣”嵌入式操作系统的特征、基础、技术要点。

2．1 32 位嵌入式操作系统

“和欣”是 32 位嵌入式操作系统。该操作系统可以从多个侧面进行描述：

32 位嵌入式操作系统。操作系统基于微内核，具有多进程、多线程、抢占式、基于线程的多优先级任务调度等特性。提供 FAT 兼容的文件系统，可以从软盘、硬盘、FLASH ROM 启动，也可以通过网络启动。和欣操作系统体积小，速度快，适合网络时代的绝大部分嵌入式信息设备。

完全面向构件技术的操作系统。操作系统提供的功能模块全部基于 ezCOM 构件技术，因此是可拆卸的构件，应用系统可以按照需要剪裁组装，或在运行时动态加载必要的构件。

从传统的操作系统体系结构的角度来看，和欣操作系统可以看成是由微内核、构件支持模块、系统服务器组成的。

- 微内核：主要可分为 4 大部分：硬件抽象层（对硬件的抽象描述，为该层之上的软件模块提供统一的接口）；内存管理（规范化的内存管理接口，虚拟内存管理）；任务管理（进程管理的基本支持，支持多进程，多线程）；进程间通信（实现进程间通信的机制，是构件技术的基础设施）。
- 构件支持模块：提供了对 ezCOM 构件的支持，实现了构件运行环境。构件支持模块并不是独立于微内核单独存在的，微内核中的进程间通讯部分为其提供了必要的支持功能。
- 系统服务器：在微内核体系结构的操作系统中，文件系统、设备驱动、网络支持等系统服务是由系统服务器提供的。在和欣操作系统中，系统服务器

都是以动态链接库的形式存在。

2．2 ezCOM 构件技术

2．2．1 ezCOM 构件技术概要

ezCOM 构件技术是面向构件编程的编程模型，它规定了一组构件间相互调用的标准，使得二进制构件能够自描述，能够在运行时动态链接。

ezCOM 兼容微软的 COM。但是和微软 COM 相比，ezCOM 删除了 COM 中过时的约定，禁止用户定义 COM 的非自描述接口；完备了构件及其接口的自描述功能，实现了对 COM 的扩展；对 COM 的用户界面进行了简化包装，易学易用。

从上面的定义中，我们可以说 ezCOM 是微软 COM 的一个子集。ezCOM 很大程度地借鉴了 COM 技术，保持了和 COM 的兼容性，同时对 COM 进行了重要的扩展。在和欣 SDK 工具的支持下，使得高深难懂的构件编程技术很容易被 C/C++程序员理解并掌握。ezCOM 中的“ez”源自与英文单词“easy”，恰如其分地反映了这一特点。

ezCOM 的编程思想是“和欣”技术的精髓，它贯穿于整个技术体系的实现中。

为了在资源有限的嵌入式系统中实现面向中间件编程技术，同时又能得到 C/C++的运行效率，ezCOM 没有使用 JAVA 和.NET 的基于中间代码—虚拟机的机制，而是采用了用 C++编程，用和欣 SDK 提供的工具直接生成运行于和欣构件运行平台的二进制代码的机制。用 C++编程实现构件技术，使得更多的程序员能够充分运用自己熟悉的编程语言知识和开发经验，很容易掌握面向构件、中间件编程的技术。在不同操作系统上实现的和欣构件运行平台，使得 ezCOM 构件的二进制代码可以实现跨操作系统平台兼容。

2．2．2　ezCOM 技术的意义

对于面向 WEB 服务的应用软件开发，以及开发操作系统这样的大型系统软件而言，采用 ezCOM 构件技术具有以下的意义：

- 不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。
- 实现在对某一个构件进行升级时不会影响到系统中的其它构件。
- 不同的编程语言实现的构件之间可以实现互操作。
- 提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。同时提供系统运行的安全、保护机制。
- ezCOM 构件与微软的 COM 构件二进制兼容，但是 ezCOM 的开发工具自动实现构件的封装，简化了构件编程的复杂性，有利于构件化编程技术的推广普及；
- ezCOM 构件技术是一个实现软件工厂化生产的先进技术，可以大大提升企业的软件开发技术水平，提高软件生产效率和软件产品质量；
- 软件工厂化生产需要有零件的标准，ezCOM 构件技术为建立软件标准提供了参考，有利于建立企业、行业的软件标准和构件库。

2．2．3　如何用 ezCOM 技术编程

ezCOM 是一个面向构件的编程模型，也可以说是一种编程思想，它表现为一组编程规范，包括构件、类、对象、接口等定义与访问构件对象的规定。

ezCOM 的实现，是由一个配套的开发环境和运行环境完成的。

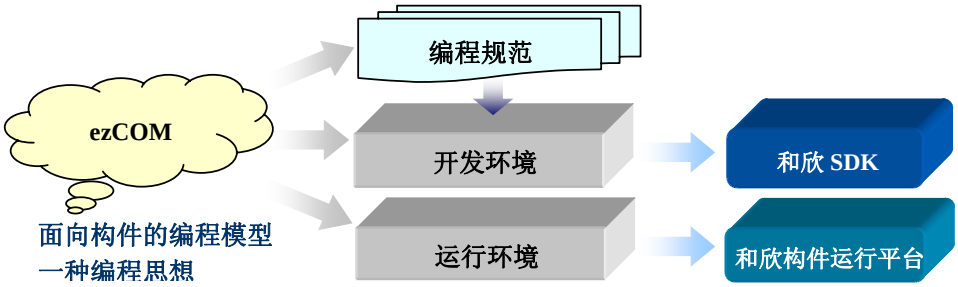


图 2.1 ezCom 运行结构

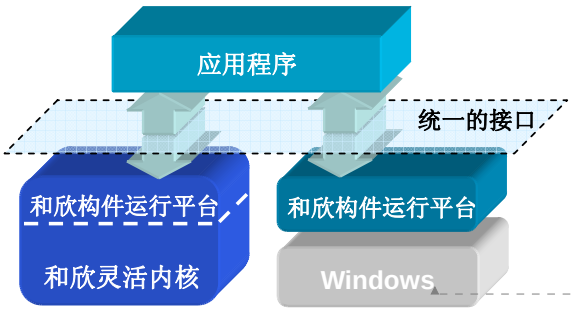
2．3“和欣”构件运行平台

2．3．1 和欣构件运行平台简介

和欣构件运行平台提供了一套符合 ezCOM 规范的系统服务构件及支持构件相关编程的 API 函数，实现并支持系统构件及用户构件相互调用的机制，为 ezCOM 构件提供了编程运行环境。和欣运行平台有在不同操作系统上的实现，符合 ezCOM 编程规范的应用程序通过该平台实现二进制级跨操作系统平台兼容。

在和欣操作系统中，和欣构件运行平台与“和欣灵活内核”共同构成了完整的操作系统。

在 Windows 2000、WinCE、Linux 等其它操作系统上，和欣构件运行平台屏蔽了底层传统操作系统的具体特征，实现了一个构件化的虚拟操作系统。在和欣



带格式的

图 2. 2 和欣构件运行平台架构

构件运行平台上开发的应用程序，可以不经修改、不损失太多效率、以相同的二进制代码形式，运行于传统操作系统之上。

图 2.2 显示了和欣构件运行平台在 Windows 2000/XP、和欣操作系统中的位置。

2．3．2 和欣构件运行平台的功能

从和欣构件运行平台的定义，知道该平台为 ezCOM 提供了运行环境。从这个意义上，这里说的 ezCOM 技术也可以理解为在运行环境中对 ezCOM 规范提

供支持的程序集合。

从编程的角度看，和欣构件运行平台提供了一套系统服务构件及系统 API（应用程序编程接口），这些是在该平台上开发应用程序的基础。

和欣操作系统提供的其它构件库也是通过这些系统服务构件及系统 API 实现的。系统提供的这些构件库为应用编程开发提供了方便：

图形系统构件库；

设备驱动构件库；

文件系统构件库；

网络系统构件库。

从和欣构件运行平台来看，这些构件和应用程序的构件是处于同样的地位。用户可以开发性能更好或者更符合需求的文件系统、网络系统等构件库，替换这些构件库，也可以开发并建立自己的应用程序构件库。

从支持 ezCOM 构件的运行环境的角度看，和欣构件运行平台提供了以下功能：

根据二进制构件的自描述信息自动生成构件的运行环境，动态加载构件；

提供构件之间的自动通信机制，构件间通信可以跨进程甚至跨网络；

构件的运行状态监控，错误报告等；

提供可干预构件运行状态的机制，如负载均衡、线程同步、访问顺序控制、安全（容错）性控制、软件使用权的控制等；

构件的生命周期管理，如进程延续（Persistence）控制、事务元（Transaction）控制等；

总之，构件运行平台为 ezCOM 构件提供了对程序员完全透明的运行环境，构件可以运行在不同地址空间，不同环境，甚至跨网络。构件运行平台自动为构件运行提供支持，配置必要的网络协议、针对不同的输入输出设备的协议。程序员不必过多地去关心诸如网络协议转换及构件运行控制等与其它构件互操作时的协调问题，只需专注于自己需要解决的程序算法的实现。从而可以从繁杂庞大的应用环境体系中解放出来，大大提高编程的效率。

和欣构件运行平台直接运行二进制构件，而不是像 JAVA 和.NET 那样通过虚拟机在运行程序时解释执行中间代码。因此，与其它面向构件编程的系统相比，具有资源消耗小，运行效率高的优点。

2．3．3 和欣构件运行平台的技术优势

作为总结，和欣构件运行平台的主要技术优势列举如下：

开发跨操作系统平台的应用软件；

对程序员透明的 ezCOM 构件运行环境，提高编程的效率；

直接运行二进制构件代码，实现软件运行的高效率；

构件可替换，用户可建立自己的构件库。

需要说明的是，和欣构件运行平台实现的应用软件跨操作系统平台兼容是以具有同样的硬件体系结构为前提的。目前，和欣构件运行平台还不能支持不同指令系统的 CPU 间的“跨平台”兼容。

2．3．4 利用和欣构件运行平台编程

对程序员来说，编写运行于和欣构件运行平台上的程序，运用 ezCOM 技术和跨平台技术的具体方法，体现在对构件库的接口方法、通用 API 函数的调用上。应用程序运行所需要的动态链接库，则是在程序运行时由和欣构件运行平台自动加载的。

下图简明地表示了编写运行于和欣构件运行平台上的应用程序所需的相关要素之间的关系的示意。

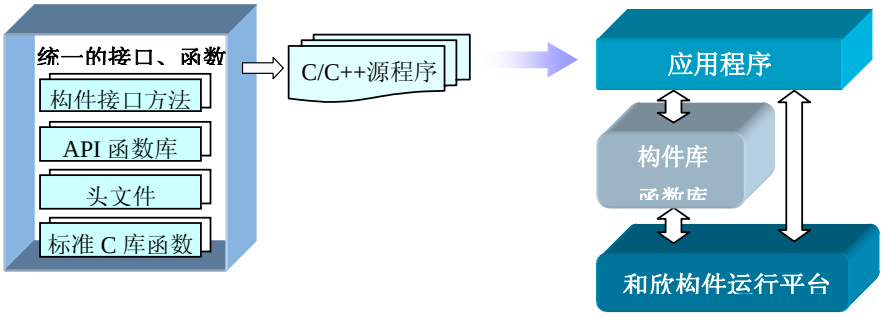


图 2．3 和欣构件运行平台上的应用程序结构

2．4“和欣”灵活内核

2．4．1 和欣灵活内核简介

和欣操作系统的实现全面贯穿了 ezCOM 思想，ezCOM 构件可以运行于不同地址空间或不同的运行环境。可以把操作系统的内核地址区看成是一段特殊的地址空间，用户可以根据运行时的需求，自主选择将操作系统的某些系统服务构件、文件系统、图形系统、设备驱动构件等运行于内核地址空间或用户地址空间。与传统的操作系统的“大内核”、“微内核”体系结构相比，和欣操作系统内核里提供的系统服务，完全可以由用户依据系统自身的需求动态决定。因此称和欣操作系统内核为“灵活内核” （Agile Kernel）。

和欣灵活内核的体系结构，利用构件和中间件技术解决了长期以来困扰操作系统体系结构设计者的大内核和微内核在性能、效率与稳定性、安全性之间不能两全其美的矛盾。

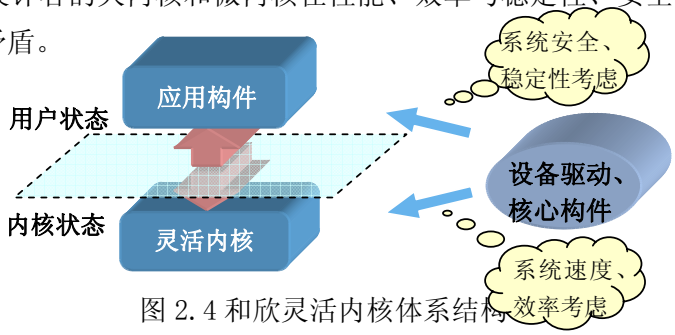


图 2.4 和欣灵活内核体系结构

2．4．2ezCOM 构件技术在和欣操作系统中的作用

ezCOM 技术由操作系统内核来实现，可以充分利用内核中的线程调度、跨进程通讯、软件装卸、服务定位等设施对 ezCOM 构件提供高效、可靠的服务。同时内核本身的程序实现也可因利用 ezCOM 技术而变得更加模块化，从而加强对内核的软件工程管理。

和欣操作系统正是基于这样的思路实现的。“和欣”中的操作系统内核、和欣构件运行平台提供的构件库，都是用 ezCOM 技术实现的。内核与 ezCOM 技术运行环境的紧密结合，为和欣操作系统的“灵活内核”体系结构提供有力的支持，高效率地实现了全面面向构件技术的新一代操作系统。

虽然 ezCOM 技术会增加内核代码量，但脱开应用一味强调内核大小并没有意义，ezCOM 技术引入内核将会大大减少各种应用软件与操作系统的总体资源开销。

在和欣构件运行平台上直接运行二进制构件，这也符合对运行效率、实时性有严格要求的嵌入式系统的工业要求。二进制代码就是实际的 CPU 指令流，其所需的执行时间是可计算的，因此，系统运行时间是可预知的（predictable），这是目前存在的其它虚拟机系统所不能及的。

2．5“和欣”SDK

2．5．1 和欣 SDK 简介

和欣 SDK 在 Windows 2000/XP 上为软件开发技术人员提供了一个面向构件化编程的应用软件集成开发环境。和欣 SDK 帮助技术人员充分运用“和欣”技术体系所包括的 ezCOM 构件技术、和欣构件运行平台技术，开发和欣构件运行平台上的应用软件。

ezCOM 构件技术的思想，充分体现在利用和欣 SDK 所提供的开发工具，利用和欣构件运行平台提供的接口、构件库等进行应用开发的过程中。利用和欣 SDK，技术人员在不用了解很多关于构件技术细节的情况下，就可以开发出面向构件的应用软件。

2．5．2 在和欣 SDK 上开发软件

和欣 SDK 可以帮助技术人员方便地进行以下开发工作：

在 Windows 操作系统上开发与微软的 COM 技术完全兼容的构件化应用软件；

开发和欣操作系统的应用软件；

调试运行在目标系统上的和欣操作系统的应用软件。

和欣 SDK 可使以下两类用户受益：

对于在 Windows 操作系统上开发应用软件的用户，可以开发与微软的 COM

技术完全兼容的构件化应用程序。在保持 ezCOM 与 COM 兼容的同时，和欣 SDK 提供了构件定义语言 CDL 和自动化功能，简化了不必要的繁琐，可以让熟悉 C/C++的程序员很容易掌握面向构件的编程技术。Windows 应用软件的编程人员完全可以把和欣 SDK 作为构件化软件的开发平台来使用。

对于开发和欣操作系统的应用软件的用户，和欣 SDK 是必不可少的开发平台。和欣 SDK 除了上述的 ezCOM 构件化软件开发环境以外，还提供了与目标机（用户开发的嵌入式系统）通信、下载、远程调试的工具。

用和欣 SDK 开发构件化软件的方法和技巧，对开发 Windows 应用软件、“和欣”应用软件都是通用的。

2．5．3 和欣 SDK 的功能

和欣 SDK 提供的功能如下：

开发环境，分类存放管理源文件和目标文件等；

zmake 自动编译环境，包括构件定义语言 CDL 及其编译器，用于生成基于 ezCOM 技术的可动态连接的二进制构件；

系统提供的函数库；

系统提供的构件库；

编写应用程序所需的头文件；

编辑工具、常用开发工具、嵌入式系统的文件传输工具、调试工具；

全面配套的技术文档，大量的程序范例。

2．5．4 和欣 SDK 的特点

完整的面向构件的应用软件开发平台；

为 Windows 2000 应用程序、“和欣”操作系统应用程序的开发提供了统一的开发环境；

构件定义语言 CDL、zmake 自动编译环境，提供了 ezCOM 编程的自动化机制，屏蔽了许多关于构件技术的复杂概念和繁琐的编程描述，大大简化了构件编程；

在和欣 SDK 上开发的应用软件实现了跨操作系统平台的二进制兼容；

作为小结，将和欣 SDK 各主要要素，以及与编程的关系图示如下：

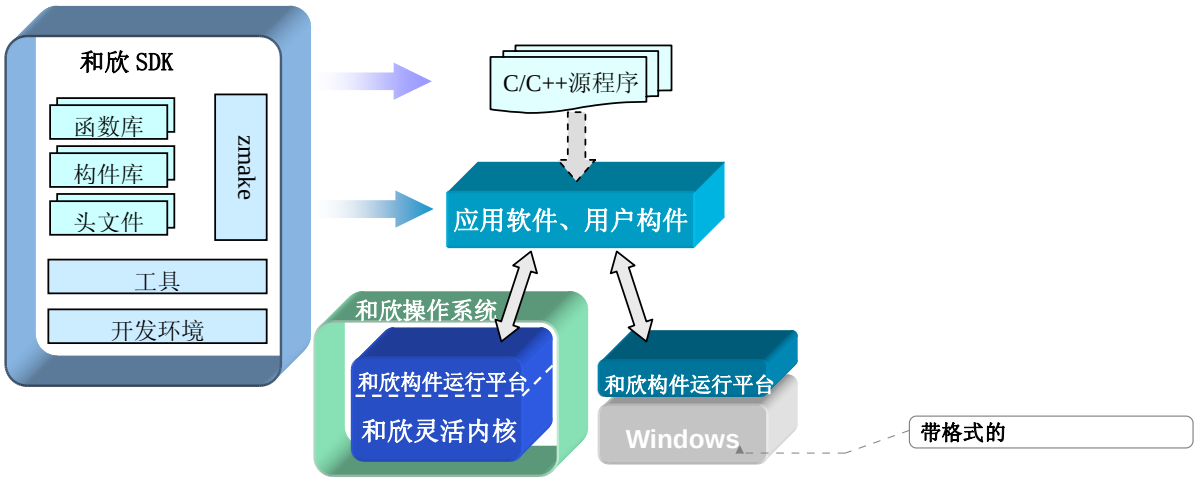


图 2.5 和欣 SDK 架构

第三章　分布式软件技术

分布式软件技术起源于 70 年代，早期研究主要集中于分布式操作系统。进入 80 年代以后，面向对象技术的应用成为发展的主流。而 90 年代软件开发技术的一个重要进展就是组件化，而开发和使用可复用的组件是从面向对象技术发展而来的一项重要技术。

组件是指在软件系统设计中能够重复使用的模块，它包装了一系列互相关联的操作和服务，作为一个在整个分布式软件系统中可以即插即用的独立对象能够有效的嵌入其他开发商开发的组件中。为了获得更高的效率厂和更好质量的产品，把软件工程师从繁琐的细节问题中解放出来，计算机软件工业的生产模式也和生产硬件一样，软件的开发最终将向硬件一样成为对数个组件的组装。

按照分布式对象的分状方式，在分布式软件系统中，完成系统功能的一部分的一个分布式对象就是一个组件。上述分布式对象计算框架使得组件具有自治性和协作性。初步的实践应用已经证明了分布式对象技术的可行性和先进性。目前比较流行的分布式软件足见对象标准主要有以下几种：OMG 的 CORBA、MICROSIFT 的 DCOM 和 SUN 公司的 RMO，而就对应的服务器而言，有 Component Object Model(COM)，Common Object Request Broker Architecture (CORBA) Beans,Enterprise JavaBeans(EJB)等等。

而最近流行的 XML BASED WEB SERVICE 技术则使用 XML 和 SOAP 协议，相对使用更简单，更容易集成，相对以上分布式技术有较大改变。下文想详细阐述他们的概念和区别。

3．1 微软 COM 技术

COM 组件技术是由 Microsoft 公司与 DEC 公司于 1995 年提出的。COM 代表 Component Object Model(组件对象模型)，而 DCOM 代表 Distributed ComponentObject Model(分布式组件对象模型)。COM 模型实际是一种通信标准用于异质网络和操作系统中创建、使用和加入可重复使用的组件对象。理论上，这些组件对象可以互相通讯与交互，而与它们的语言、分布及原始平台无关。

COM 规程包括一套标准 API、一个标准的接口集以及 COM 用于支持分布式计算的网络协议。

而 DCOM 模型则是一套用于分布式环境中的 COM 对象，在 DCOM 环境中位于

一个网络上的 COM 对象能与位于另一个网络上的 COM 对象进行通信。为了便

于理解，可以把 COM 看作某种(软件)打包技术，即把它看作是使软件的不同部分按照一定的面向对象的形式，组合成可以交互的过程和一组支持库。COM 对象可以用 C++, Java 和 VB 等任意一种语言编写，并可以 DLL 或作为不同过程工作的执行文件的形式来实现。使用 COM 对象的客户端，无需关心对象是用什么语言写的，也无需关心它是以 DLL、还是以另外的过程来执行的。

Windows NT 从 4.0 版开始支持 DCOM, DCOM 增强了 COM 的分布处理性能，支持多种通信协议，加强组件通信的安全保障，把基于认证的 Internet 安全机制同基于 Windows NT 的 C2 级安全机制集成在一起。它使得局域网上不同计算机间的 COM 可以进行通信，这种通信基于 RPC 协议。MTS(Microsoft TransactionServer)是为 Windows NT Serve:开发的一组软件，,MTS 不仅可以实现事务监控，它还为 COM 提供了一个全新的运行期环境，增加了许多 COM 不支持的功能加:分布式事务、安全模型、线程缓冲池，并增强了配置和管理能力。MTS 还提出了基于属性编程的概念，NITS 平台提供的服务可以通过设置属性来改变，程序设计者可以在设计期间决定属性的设置，也可以在应用期间改变属性的设置。

虽然 MTS 对 COM 的功能进行了一些改进和扩充，但它们并不能彻底融合成

为一个整体，,COM 是 Windows NT 直接提供的，而 MTS 是可选的，二者拥有不同的编程模型，运行时所处的层次也不同。随着 Windows 2000 的发布，微软把 COM,DCOM 和 MTS 集合在一起称为 Cow ,它是基于 MTS 和 COM 扩展的一组服务，提供了改良的线程和安全性、事务管理、对象池、排队组件以及应用程序管理及打包。COM++倡导一种新的设计概念，把 COM 组件提升到应用层，把底层细节留给操作系统，使得 COM 十与操作系统的结合更加紧密。COM+的底层结构仍然以 COM 为基础，但在应用方式上则更多地继承了 MTS 的处理机制，包括 MTS 的对象环境、安全模型、配置管理等。

COM+是一系列为基于 COM 应用设计的服务，要使用这些服务也很简单。首

先需要创建一个组件对象，创建组件对象所要做的事情就是使它支持 IUnknown 接口。然后创建一个 COM 十应用程序，再把要用的组件对象加入到这个 COM 十应用程序中。最后要做的就是通过属性的设置指出需要哪种服务，也就是为每一个组件建立一个环境，指出组件是否需要事务处理、负载平衡、队列服务等等，所有这些信息就叫做环境，这种方法称作面向属性编程。以后当客户调用一个组件对象时，一个拦截器拦截了调用，并在调用到达组件对象之前检查当前的环境，而且指出需要什么服务。例如当客户发出一个调用请求后，拦截器首先得到调用，它将检查对象所处的环境，得出该组件需要事务支持，于是系统先进行相应事务处理操作，再将调用请求传递给组件对象，由它执行调用。当调用完成后，返回时拦截器又截取调用，搜索环境看是否需要什么清理工作。所有这些都将以环境的形式指出，而不需要在编程时考虑。这样做的好处是大大简化了编程，开发者可以专注于业务逻辑，而不是编程结构和技巧。这将导致一个非常简单和规范的编程模型，而且易于升级，易于扩展。

3．2CORBA 技术

CORBA (Common Object Request Broker Architecture，公共对象请求代理体系结构)是国际对象管理组织 OMG 提出的一种分布式对象处理体系结构，它很好地结合了面向对象和分布处理技术，而这两者的结合正是当今软件产业的发展方向。CORBA 主要目标是实现重用性、移植性和分布的、异种的环境中基于对象软件的互操作性。这些规格一致性将可能开发出一种异种的、跨所有主要硬件平台和操作系统的应用环境。

1.CORBA 的结构

从体系结构上看，CORBA 由四部分组成。

〔1)对象请求代理 ORB:它是 CORBA 的核心，它保证在分布式异构环境中，透明地向对象发送和接收请求，帮助实现应用组件之间的互操作。ORB 是支持 CORBA 构件相互作用的“软总线”(Software Bus)，服务性构件可以向它登

记注册，当客户性对象需要某种服务时，可以向它发出请求，ORB 负责搜索已在其“注册登记”了的服务性对象，找到后启动该服务，传送请求给该服

务性对象，并将结果传送回给客户性对象。

(2)公共对象服务(Common Object Services):这是一些最有可能被用来支持分布式对象环境下构造应用的标准化组件，目前已通过的公共对象服务包括

对象命名服务、事件服务、对象生存期服务、永久对象服务、对象关系服务等。

(3)公共设施(Common Facilities):它是比公共对象服务粒度更大的可重用的构件块，它主要用来帮助构造跨多个应用域的应用程序，典型的公共设施包括用户接口、信息管理、系统管理和任务管理等。

(4)应用对象(Application Objects):它是指供应商或用户借助于 ORB.公共对象服务及公共设施而开发的特定产品，它不在 CORBA 体系结构中实行标

准化。

2.CORBA 的组件及接口

在 CORBA 中除核心 ORB 外，还有许多组件与接口，它们与 ORB 共同组成 CORBA 体系结构。这些组件及接口主要包括:(1)修口定义语言 IDL 产生的客户的桩 Stub 以及服务器端的骨架 Skeleton, (2)动态调用接口 DII, (3)接口池(4)对象适配器(Object Adapter)ICORBA 的服务

命名服务(Naming Service):它支持用逻辑名定位服务对象，支持名字上下文的结构。

安全服务〔Security Service〕:提供了分布对象安全的完整框架，支持身份认证、访问控制、信息加密等。

事件服务(Event Service):允许用户动态注册和注销自己感兴趣的事件，还包括主动服务 Push 和 Pull 技术等。

时钟服务(Time Service):提供分布式对象环境下的时钟服务，也可定义和管理事件的触发。

(5)生命周期服务(Life Cycle Service):支持对象的创建、复制、迁移和删除操作。

3.CORBA 的特点

由上所述不难看出，采用 CORBA 的特点非常突出:

(1)它采用软构件及软总线的概念，实现了系统异构平台之间的统一，客户端与服务器完全可以在不同的平台上，用不同的语言来编写应用，具有很好的开放性和灵活性。

(2)它通过对象引用技术来唯一确定分布式环境下的对象实例，实现了分布式对象处理功能。

(3)一个对象实现可为多个客户端应用调用，也可调用其他的对象实现，实现了对象的可重用性和互操作性。

(4) CORBA 提供的公共对象服务功能很强，其中基于事件服务的主动服务 Push 和 Pull 技术，是处理实时系统的一种很好的技术。

3．3XML BASED WEB SERVICE

3．3．1 xml 概述

XML（eXtensible Markup Language，意为可扩展的标记语言）是国际组织 W3C 定义的一种元标记语言。W3C 对 XML 的描述是：“XML 是标准通用标记语言（Standard Generic Markup Language,SGML[ISO 8879]）针对应用的一个子集，或者说是 SGML 的一种受限形式”。与 HTML 一样，也是源自 SGML（Standard Generalize Markup Language），它保留了 SGML80%的功能，但复杂程度却降低了 20%。同时，XML 是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识。它也是元标记语言，即定义了用于定义其他与特定领域有关的、语义的、结构化的标记语言的句法语言。

XML 描述的是结构和语义，而不是格式化。XML 标记描述的是文档的结构和意义，而不是页面元素的格式化。可以用样式单为文档增加格式化信息。文档本身只说明文档包括什么标记，而不是说明文档看起来是什么样的。相比之下，HTML 文档包括了格式化、结构和语义的标记。

XML 和 HTML 相比主要有以下几个优点：

1 良好的可扩展性。XML 允许不同机构根据自己独特的需要来创建自己的一套标记。这种开放的解决方法更有助于置标语言的发展。

2 内容与形式的分离。XML 把信息的显示方式从信息本身中抽取出来。这样做便于信息表现方式的修改，便于数据的搜索，也使得 XML 具有良好的自描述性，能够描述信息本身的含义以及它们之间的关系。

3 遵循严格的语法要求。XML 不但要求标记配对，嵌套，而且还要求严格遵守 DTD 的规定，增强了文档的可读性和可维护性。

4 便于不同系统之间信息的传输。不同企业，不同部门中往往存在不同的系统，XML 可以作为不同系统之间的交流媒介，是一种非常理想的网际语言。

XML 的核心信息单元是 XML 文档。与其他任何复杂的软件一样，XML 文档包含有物理部分与逻辑结构。XML 建议书在 XML 文档的物理内容与逻辑结构之间制定了差别。尽管这种区别非常明显，但在实际中，很难在不提及物理实体的情况下讨论逻辑结构。

从物理上讲，一个 XML 文档可以由文本字符构成。这些字符包含在称为实体的“存储单元”中。实体被划分为多种类别，比如字符实体，命名实体，文档实体以及外部 DTD 子集。从文法上讲，一个 XML 文档可定义为：

document::=prolog element Misc*

这说明 XML 文档含有一个序言（Prolog），接着是根元素、可选择的各种内容。

在逻辑上，XML 文档由以下五种成分组成：

组件类型	被……定界
DTD 中的声明	<!DOCTYPE[…]>
	<!ELEMENT…>
	<!ATTLIST…>
	<!ENTITY…>
	<!NOTATION…>
文档内容部分的元素	<tag…>…</tag>或<tag…/>
CDATA 部分	<![CDATA[…]]>
注释	<!-- -->
处理指令(PI)	<?…?>

元素构成了 XML 文档的基本结构，一个元素声明由元素的名称与一个内容说明组成：

elementdecl::=’<!ELEMENT’ S QName S contentspec S? ‘>’

Qname 代表“Qualified Name”，表示一个名称由命名空间前缀限定。“contentspec”一共有四种，由四个规范说明：

contentspec::= ‘EMPTY’ | ‘ANY’ | Mixed | children

<!ELEMENT br EMPTY>

<!ELEMENT container ANY>

<!ELEMENT p (#PCDATA|a|ul|b|I|em)*> <!—Mixed content -->

EMPTY 是自解释的，没有内容。ANY 是一种混合内容，允许分析性的字符数据与所有可能类型的子元素按任意顺序组合。

对于比较复杂的 XML 文档，我们需要定义子元素。我们引入内容粒子（cp）的概念。最简单的内容粒子就是一个前面提到的 QName，这是基本情况。一个内容粒子具有高度递归性：一个 cp 还可以是一个 cp 的选择（用竖线符分隔 cp）或一个 cp 的序列（用逗号分割 cp）。最后，一个 cp 可以用可选择的重复因子字符‘+’、‘*’和‘？’结尾：

cp::= (QName | choice | seq) (‘?’ | ‘*’ | ‘+’)?

choice::=’(‘ cp (‘|’ cp)* ‘)’

seq::= ‘(‘ cp (‘,’ cp)* ‘)’

子内容说明自己也是一个选择或一个系列，后面接着一个可选择的重复因子字符。所有这些都归结为，我们可以构建任意深度的真正复杂的元素声明，这一点，对于我们用 XML 来描述组件关系是很有利的。

3．3．2XML 文挡的解析

XML 格式的数据在使用前要对其进行解析。目前有两种流行的解析方法：DOM（文档对象模型）和 SAX（Simple API for XML）。

3．3．2．1 DOM

DOM 文档 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中导航仪寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而 DOM 被认为是基于树或基于对象的。

DOM 基础

DOM 事实上就是采用树状对象集合的方式访问给定文档内容的抽象规范。

DOM 规范与 Web 世界的其他标准一样受到 W3C 组织的管理，在其控制下为多重平台和语言使用 DOM 提供一致的 API，W3C 把 DOM 定义为一套抽象的类而非正式实现 DOM。因此，正是独立的开发商实际上提供具体平台和开发语言下规范接口的实现。DOM 的接口定义是采用对象管理组织的 接口定义语言（IDL）创建的。即便你对 IDL 并不太了解也没有多大关系，因为 IDL 的自我

描述非常得当，你直接查阅这些定义不会受到理解上的困难。在本文中我给每一个谈到的接口都设置了对应的 IDL 定义的链接，这样你就可以在必要的情况下引用它阅读相应的文档。

DOM 的功能分为 3 个级别：

级别 1 只提供了解析 XML 文档所必需的最基本支持。

级别 2 扩展了级别 1 以支持 XML 名称空间。这是目前推荐的功能级别，而且我在这篇文章中引用的都是 DOM 接口的级别 2 版本。

级别 3 这一级别增加了对 Xpath 查询、装载和保存文档的支持。

由于 W3c 的规范只是最低要求的建议，产品供应商在大多数情况下还会对 DOM 规范进行专有性的技术扩展。比方说，这就是为什么许多可用 DOM 的实现已经内建 XPath 支持的原因。在使用这些扩展的时候你可得小心谨慎，尤其是那些采用第 3 级别的功能。那些对象的接口还在经常性的变动之中，最终，正式版本可能与你根据工作版本编写的代码不兼容。

DOM 的对象模型

DOM 把文档表示为节点（Node）对象树。 “树”这种结构定义为一套互相联系的对象的集合，或者说节点，其中一个节点作为树结构的根（root）。节点被冠以相应的名称以对应它们在树里相对其他节点的位置。例如，某一节点的父节点就是树层次内比它高一级别的节点（更靠近根元素），而其子节点则比它低一级别；兄弟节点显然就是树结构中与它同级的节点了——不在它的左边就在它的右边。图 3.1 给这些术语提供了图形化的解释，

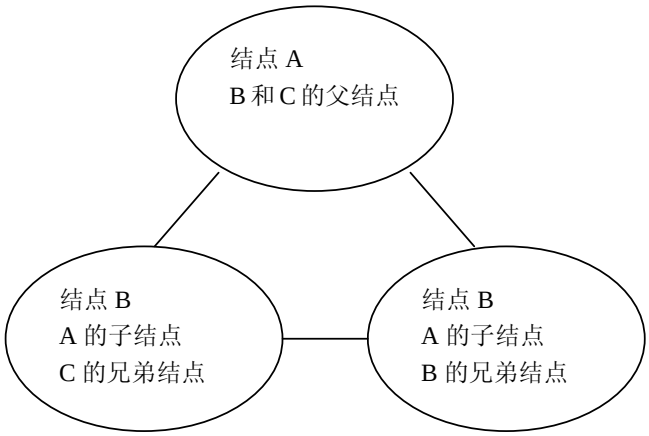


图 3.1 XML 节点关系的图形

节点对象不但表示了文档中的XML元素而且代表了在一个文档之内的其他所有内容，从最顶端的文档元素自身到单独的内容要素，比如属性、注释以及数据等等都包括在内。每一个节点都有其专门的接口，这些接口对应于节点所代表的 XML 内容，但这些接口其实在本质上也是节点。面向对象的支持者会说所有的 DOM 对象都继承于节点。而节点（node）接口则是用来导航文档树、增加新节点以修改一个文档结构的主要方法。

节点概述

节点暴露了一些导航元素供你在文档树上移动。parentNode 方法返回当前节点的父节点，而 nextSibling 和 previousSibling 方法则返回位于当前节点右边或者左边的兄弟节点。检查 hasChildNodes 属性还可以让你知道给定的节点是否存在子节点。

假设某一节点存在子节点，这些子节点就可以用 ChildNodes 属性检索。ChildNodes 用 NodeList 结构返回当前节点所有的直属(低 1 级)子节点。NodeLists 是代表一组节点的有序列表（按索引号获取），而它们的同类 NamedNodeMaps 则把这些返回的节点表示为字典（按名称获取）。这两种对象都是“活”的对象，意思是说对列表的任何修改都会立即受到对象树的反应。

节点对象还暴露了一组增加并且删除子节点的方法。 insertBefore 方法在子节点列表内指定节点之前（在左边）直接插入一个新节点，而 appendNode 方法则在当前节点的子节点列表的末尾（最右边）附加新节点。replaceNode 方法直接用其他节点代替指定子节点，而 removeNode 则从子节点组中删除给定的节点。

专用节点接口

节点接口提供了导航文档以及修改文档树的便捷方法，但为了能真正完成大部分有意义的工作，你首先要钻研那些抽象的 DOM 接口。在本文章的余下部分，我就引领读者考察部门此类接口。

文档为根

Document 接口对 Node 接口进行了扩展以代表完整的 XML 文档，同时提供了文档树的根元素（<XML>元素）。你在装载一个 XML 文档时大多数实现都会传递给你一个文 Document 对象。Document 是一种影响文档整体而并不真正适用于其他方面的产物。其大多数方法主要作为创建其他 DOM 对象的工厂方法。这些形如“createX”的方法可以为那些不支持传统构造器语言实现创建 Element、

DocumentFragment、TextNode、CDATASection、ProcessingInstruction、属性（Attr）、EntityReference 以及各类名称空间节点。

Document 还包括两种在文档内移动到特定位置的有用方法：

getElementsByTagName 返回给定标签名字内所有元素的 NodeList。其列表内的排列顺序按照在文档中遇到这些元素的顺序确定。这是一种在文档中检索特定元素所有实例的简便方法，而且因为元素作为 Node 被返回，所以围绕文档的导航成为可能。

getElementByID 返回其类型 ID 属性匹配指定 ID 的元素。该方法用来快速定位文档中的单个元素。

Document 接口还有一个很有意思的地方，其 Node 接口暴露了一个 ownerDocument 性质，通过它可以返回该节点的父 Document 对象。

树的元素

element 接口主要处理属性（也可以由根 node 接口所用），其 13 个方法提供了访问多种属性的形式。在这其中，你可能最常用到 getAttribute/setAttribute 和 getAttributeNode/setAttributeNode 方法。前者允许你在提供了属性名称的前提下直接读写属性值。后者则允许你操作代表属性的实际 Attr 对象。Element 接口的过人之处是任何检索数据的方法都关联某一元素。这是因为给定元素的数据被当作该元素的子节点，而子节点则可以通过根 Node 接口的 ChildNodes 属性获取。如果子节点只包含了字符数据，那么元素的数据节点就只实现 Text 接口。然而，在复杂数据的情况下，实现适当的 Element、Attr 和/或 Text 接口的一组子节点，按照数据的类型即可代表元素的子节点。

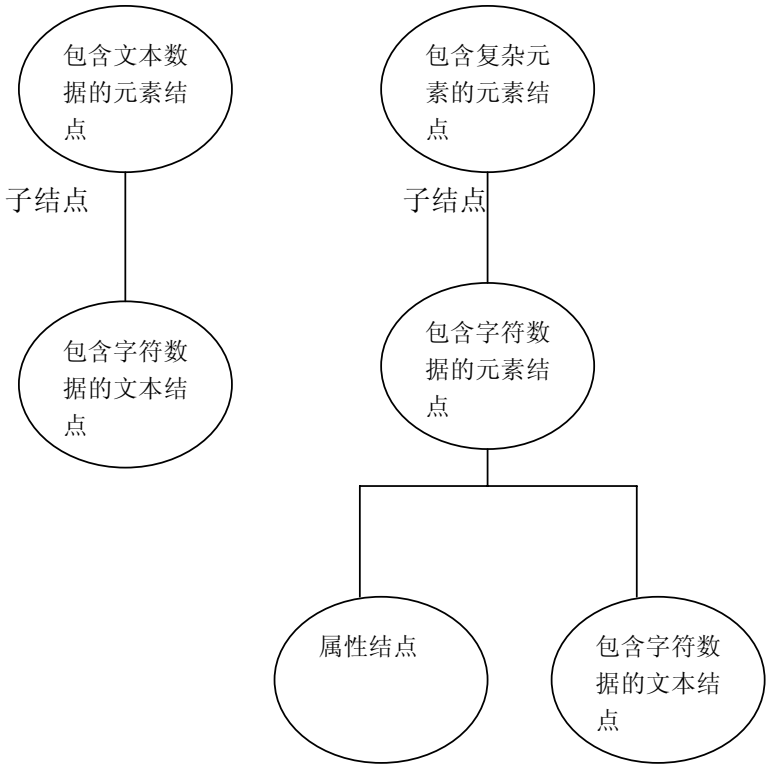


图 3.2 元素及其数据节点之间关系

关联数据节点的两个元素

文档分段

在处理 XML 时，有个常见的任务就是创建新的元素并把它们附加到现有的文档。DocumentFragment 接口以修改插入方法（insertBefore、appendNode 和 replaceNode）行为的方式略微扩展了 node 以便在 DocumentFragment 插入文档的情况下，只有其子节点被插入，而 DocumentFragment 节点自身则不被插入。这样就促使 DocumentFragment 成为 XML 树内新节点的一种理想的临时附着点。

3．3．2．2SAX

在用 DOM 耗费较长时间解析 XML 文件以后，你可能注意到在用 DOM 处理大型文件时其性能下降的非常厉害。这个问题是由 DOM 的树结构所造成的：这种结构占用的内存较多，而且 DOM 必须在解析文件之前把整个文档装入内

存。在采用 DOM 之后性能受到严重影响的情况下，你不妨考虑使用 Simple API for XML（SAX）。

DOM 提供了基于对象的接口，它解析 XML 文档后会在内存中创建一棵对象树，然后应用程序根据这棵树做进一步操作，如根据它创建业务层对象等等。

SAX 则提供了基于事件的接口，它解析 XML 文件时，会产生一系列事件，并将事件发送给应用程序中的事件处理器，然后事件处理器对相应的事件做出相应的处理，如创建业务层对象等等。

SAX 不是官方的标准，它是 XML 社区中的标准，是对 W3C 组织发布的 DOM 的一种补充。

不要被 SAX 名称中的 simple 所迷惑，如果你用 SAX，实际要做的编码工作往往要比使用 DOM 还多，但对于体积较大的 XML 文件来说，它的确是个更好的选择。

SAX 分析 XML 文件的时候，会产生四种事件：Content，Error，DTD，Entity，并对每个事件定义了接口：

(1)ContentHandler：定义与文档内容关联的事件（如开始和结束标记），这是最常用的接口。

(2)ErrorHandler：定义错误事件。也是必不可少的接口。

(3)DTDHandler：定义与 DTD 关联的事件。

(4)EntityResolver：定义装入实体关联的事件。比较少用到。

每种事件发生后，语法解析器都会调用事件处理器（注意，不是事件解析器调用语法解析器），由事件处理器根据不同的事件做出不同的响应。

事件处理器

前面说了 SAX 使用了四个接口来定义解析时产生的事件，也就是说我们的事件处理器要具备这些事件接口并负责处理事件，而 SAX 语法解析器当然也具备这些接口，不过它只负责调用事件处理器，我们现在简单说明一下这些接口的定义说明。

(1)ContentHandler 接口：SAX 最常用的接口，它定义了下列事件：

startDocument;　文档开始。是解析器发出的第一个事件。

endDocument;　文档结束，是解析器发出的最后一个事件。发生在到达文档末尾或者解析时出现无法逾越的错误。

startElement;　节点(element)开始。在 endElement 事件之前，节点的内容会

被解析器按顺序地，作为 Attributes 参数传给事件处理器。短格式的节点（如）也会触发 startElement , endElement 这两个事件。

endElement; 节点结束。

startPrefixMapping; 开始进入 namespace 的区域。一般来说你不用关注这个事件，因为当你将 XMLReader 的 Feature 属性设为 True 后，它会自动去做替换前缀的工作。

endPrefixMapping; namespace 区域的结束。

characters; 发现了文本(包括空格或者回车换行)。解析器会传给应用程序一个连续的字符串。

ignorableWhitespace; 发现了 XML 标准定义的可忽略空格。同样，也是一个连续的字符串。

processingInstruction; 将处理指令通知应用程序。也就是 XML 中<?target data?>样式的标签，不过 XML 文件第一行的<?xml version="1.0" encoding="..." ...?>不会触发这个事件。 ※skippedEntity; parser 告诉应用程序它已经跳过了一个实体(当 parser 未在 DTD/schema 中发现实体声明时)。

setDocumentLocator; 解析器通过这个函数将 Locator 对象发给应用程序。Locator 提供了目前在文档中的位置，因为 SAX 没有强制要求 Parser 要负责提供 Locator(它只是推荐)，所以如果 Parser 提供了 Locator，那它必须通过这个函数告知应用程序，而且这将是第一个事件。

(2)ErrorHandler 接口：SAX 异常事件的基本接口，定义了三个级别的错误。

应用程序必须实现对错误信息的处理，因为 Parser 不会抛出异常，抛出异常是事件处理器的事。 ※warning; 警示，不违反 XML 规范的错误。例如，文档中没有 XML 声明。

error; 错误，违反 XML 规范的错误，当 XML 文件是 well-formed 但某些值 invalid 的时候产生。

finalError; 致命错误，当 XML 文件不是 well-formed 的时候产生。

(3)DTDHandler 接口：声明了两个与 DTD 语法分析器相关的事件。

notationDecl; 通知应用程序已经声明了一个标记。

nparsedEntityDecl; 通知应用程序已经发现了一个未经过语法分析的实体声明。

(4)EntityResolver 接口：对象解析的基本接口

因为 SAX 语法分析器已经可以解析大多数 URL，所以很少应用程序实现 EntityResolver。

resolveEntity; 运行应用程序解析扩展对象。

3．3．3 web 服务概述

在 2002 年秋季以前部署的 Web 应用大多数都通过服务器发布动态内容，通过浏览器来进行人机交互。这种 Web 应用提供的服务都是由人直接调用。事实上，在大型软件系统中，单个的 Web 服务器往往不能承担所有负荷，一般都存在一个 Web 服务器群，它们之间的数据交换由于数据量巨大和时间分布不规则等原因，通常不是由人来调用而是由程序来调用。

随着 SOAP 等技术的发展与普及，以及电子商务的迅速崛起，一种新的开发基于 Web 的应用模式正在迅速发展，这就是 Web 服务（Web services）。Web 服务基于一些正在发展中的技术，如 SOAP,UDDI,WSDL。

Web 服务是一个崭新的分布式计算模型，尽管其思想早就存在，但其得到迅速发展却是最近的事情。Web 服务是一系列标准的综合，这些标准包括 XML,SOAP,UDDI,WSDL,WSFL 等。Web 服务利用这些标准提供了一个松散耦合的分布式计算环境。在 Web 服务的模型中，厂商将其服务封装成一个个相对独立的 Web 服务，每个服务提供某类功能；客户可以通过绑定到 HTTP 的 SOAP 来访问这些服务。

3．3．3．1Web 服务架构

Web 服务使用 SOA（面向服务的架构 Service Oriented Architecture）架构，该架构由三个参与者和三个基本操作构成。三个参与者是：服务提供者（Service Provider），服务请求者（Service Request），服务代理（Service Broker）。三个操作是：发布（Publish），查找（Find），绑定（Bind）。服务提供者把服务发布到代理的一个目录上，请求者到代理的目录上搜索该服务，得到如何调用该服务的信息，然后根据这些信息去调用服务提供者发布的服务。在 Web services 体系中，使用 WSDL 描述服务，UDDI 发布查找服务，SOAP 执行服务调用。SOA 模型如图 3.3 所示：

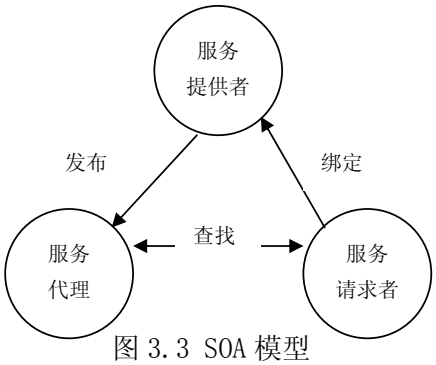


图 3.3 SOA 模型

3．3．3．2 Web 服务的特征

Web 服务是一种部署在 Web 上的对象/组件，它具备如下特征：

良好的封装性

Web 服务既然是一种部署在 Web 上的对象，自然具备对象的良好封装性，对于使用者而言，他能且仅能看到该对象提供的功能列表。

松散耦合

这一特征也是源于对象/组件技术，当一个 Web 服务的实现发生变更的时候，对于调用者来说，Web 服务的实现任何变更对他们来说都是透明的，甚至是当 Web 服务的实现平台从 J2EE 迁移到了.Net，或者是相反的迁移流程，用户都可以对此一无所知。对于松散耦合而言，尤其是在 Internet 环境下的 Web 服务而言，需要有一种适合 Internet 环境的消息交换协议。而 XML/SOAP 正是目前最适合的消息交换协议。

使用协议的规范性

首先，作为 Web 服务，对象界面提供的功能应该使用标准的描述语言来描述。其次，安全机制对于松散耦合的对象环境是很重要的，因此我们需要对诸如授权认证、数据完整性（比如签名机制）、消息源认证以及事务的不可否认性等运用规范的方法来描述、传输和交换。最后，在所有层次的处理都应当是可管理的，因此需要对管理协议运用同样的机制。

使用标准协议规范

作为 Web 服务，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，以便由任意方进行实现。

高度可集成能力

由于 Web 服务采用简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范，完全屏蔽了不同软件平台的差异，无论是 CORBA、DCOM 还是 EJB 都可以通过这一种标准的协议进行互操作，实现了很高的可集成性。

3．4 几种技术的比较

WEB SERVICE/SOAP 刚提出来的时候，许多人就提出了疑问：SOAP 与 CORBA 和 DCOM 的区别何在？

首先指出的是 SOAP 不会取代 CORBA，COM/DCOM，三者的概念有所区别。COM/DCOM 是个组件模型标准，CORBA 是分布式应用的服务标准。CORBA 和 DCOM 为分布式应用程序建立服务，服务对象来执行客户端调用的服务。而 SOAP 是基于 XML 和 HTTP 的分布式对象的通信协议，是 COM/DCOM 和 CORBA 对象进行通讯的协议。实际上，利用 SOAP 的互操作性和 CORBA 强大的执行能力，两者可以很好的结合在一起。 OMG （Object Management Group responsible for the CORBA specification）正在关注这方面的发展。

CORBA 应用程序和 DCOM 应用程序不能实现互操作，两者不能在一起协作。因为在 ORPC(Object RPC)协议中，用 ObjRef 代表了一个正在运行对象的引用；在 CORBA/IIOP(Internet Inter-Orb Protocol)中，用交换可互操作对象引用 IOR（Interoperable Object Reference）代表一个服务器的对象引用。不幸的是，IOR 与 ObjRef 不能够关联起来。然而，使用 SOAP 可以实现在垂直应用层面上 CORBA ，DCOM 技术的水平整合，能够更好的集成 CORBA，DCOM 为一个整体。

SOAP 并没有定义信息的语义，服务质量，基于 INTERNET 的事务处理。而是采用 XML 进行消息编码，正确的处理需要服务器和客户端本身来执行，理解和执行彼此使用的信息格式（ONE-TO-ONE，REQUEST/REPLY，BROADCAST，ETC），应用程序本身在语义解析中扮演着十分重要的角色。而 CORBA，DCOM 表示了传送信息的语义，对参数和返回值使用二进制编码。可对诸如参数名称或类型的任何元信息都不编码，但使中介很难处理消息。又因为每个系统使用不同的二进制编码，系统间的互操作的很难实现。

尽管 CORBA 可以在不同的平台上执行，DCOM 可以在微软的各种平台上运行，但是基于 CORBA 和 DCOM 的解决方案必须依赖于单一的应用程序。比如说，假如运行的是 DCOM 服务器程序，所有的分布式的客户端不得不运行于微软的

操作平台上。CORBA 虽然可以运行于不同的平台，但 CORBA 的互操作性并没有在更高层的服务上进行扩展，如安全性和事务处理，在这种情况下，许多提供的服务没有得到很好的优化。DCOM 和 CORBA 适合于服务器--服务器间的通讯，但是对于客户端--服务器的通讯十分脆弱，尤其当客户程序分布在 INTERNET 上更是如此。

SOAP 不象 DCOM 一样试图定义分布式系统的所有元素，SOAP 没有提供分布式类库，类型安全检查，版本控制等等，SOAP 比它处于一个更低的层次，有点类似于 IIOP 在 CORBA 的作用，DCOM 却提供了一些额外的协议功能，是 IIOP 或者 SOAP 所不具备的。然而，许多. DCOM 的额外功能只有在服务器－－服务器间通信时才会用到，对于客户端－－服务器之间的通信则是多余的。

SOAP 简单的理解，就是这样的一个开放协议 SOAP=RPC+HTTP+XML：采用 HTTP 作为底层通讯协议；RPC 作为一致性的调用途径，ＸＭＬ作为数据传送的格式，允许服务提供者和服务客户经过防火墙在 INTERNET 进行通讯交互。RPC 的描叙可能不大准确，因为 SOAP 一开始构思就是要实现平台与环境的无关性和独立性，每一个通过网络的远程调用都可以通过 SOAP 封装起来，包括 DCE（Distributed Computing Environment ） RPC CALLS，COM/DCOM CALLS, CORBA CALLS, JAVA CALLS，etc。

SOAP 使用 HTTP 传送 XML，尽管 HTTP 不是有效率的通讯协议，而且 XML 还需要额外的文件解析（parse），两者使得交易的速度大大低于其它方案。但是 XML 是一个开放、健全、有语义的讯息机制，而 HTTP 是一个广泛又能避免许多关于防火墙的问题，从而使 SOAP 得到了广泛的应用。但是如果效率对你来说很重要，那么你应该多考虑其它的方式，而不要用 SOAP。

为了更好的理解 SOAP,HTTP,XML 如何工作的，不妨先考虑一下 COM/DCOM 的运行机制，DCOM 处理网络协议的低层次的细节问题，如 PROXY/STUB 间的通讯，生命周期的管理，对象的标识。在客户端与服务器端进行交互的时候，DCOM 采用 NDR（Network Data Representation）作为数据表示，它是低层次的与平台无关的数据表现形式。

DCOM 是有效的，灵活的，但也是很复杂的。而 SOAP 的一个主要优点就在于它的简单性，SOAP 使用 HTTP 作为网络通讯协议，接受和传送数据参数时采用 XML 作为数据格式，从而代替了 DCOM 中的 NDR 格式，SOAP 和 DCOM 执行过程是类似的，如下图，但是用 XML 取代 NDR 作为编码表现形式，提供

了更高层次上的抽象，与平台和环境无关。

客户端发送请求时，不管客户端是什么平台的，首先把请求转换成 XML 格式，SOAP 网关可自动执行这个转换。为了保证传送时参数，方法名，返回值的唯一性，SOAP 协议使用了一个私有标记表，从而服务端的 SOAP 网关可以正确的解析，这有点类似于 COM/DCOM 中的桩（STUB）。转化成 XML 格式后，SOAP 终端名（远程调用方法名）及其他的一些协议标识信息被封装成 HTTP 请求，然后发送给服务器。如果应用程序要求，服务器返回一个 HTTP 应答信息给客户端。与通常对 HTML 页面的 HTTP GET 请求不同的是，此请求设置了一些 HTTP HEADER，标识着一个 SOAP 服务激发，和 HTTP 包一起传送。例如：对于一个询问股票价格的应用程序，服务器端具有组件提供某股票当前的价格，组件是 COM 或 CORBA 在服务器上建立的。客户端发送一个 SOAP 请求给服务器询问股票价格。服务器依赖于服务器上的 SOAP 网关，使用内嵌的 HTML 对象调用合适的方法，然后把得到的价格通过 SOAP 应答传给客户端。

总之和 CORBA、COM/DCOM 相比，WEB SERVICE/SOAP 的优点在于编写简单、容易集成以及平台无关性

3．5 使用嵌入式操作系统和 WEB SERVICE 解决嵌入式设备软件集成问题

由于基于 XML 的 WEB SERVICE 的以上优点和“和欣” 嵌入式操作系统的面向组件中间件的特点，我们可以利用“和欣”和 XML 的 WEB SERVICE 来构造一个模型，此模型可以相对简单的解决将嵌入式设备集成到大型分布式软件中来的问题。

利用“和欣”和 XML 的 WEB SERVICE 来解决将嵌入式设备集成到大型分布式软件中的问题的核心思想是，在“和欣”上实现一个将“和欣”ezCOM 组件包装成用 XML 和 SOAP 向外交互的 WEB SERVICE 发布的组件。因为 WEB SERVICE 的所有协议都是统一的，规范的，通用的，同样具有 WEB SERVICE 的易集成性，可以直接用 SOA 的方法来设计软件，对于 WEB SERVICE 来说传统的计算机和嵌入式设备并无不同，都是一个可以接受和发布标准 WEB SERVICE 的标准节点。而对于“和欣” 嵌入式操作系统的 ezCOM 技术来说，

所有的 WEB SERVICE 的具体函数都可以写成 ezCOM 形式，这就给动态装载带来了可能，就是通过这个机制才能把二进制代码进行简单配置后使用。如果出现了新的嵌入式设备，只要“和欣” 嵌入式操作系统经修改和编译可以支持这种新的设备，那么我们的代码也只要进行一次不需要修改源代码的交叉编译，就可以运行在新的设备上。

对于新接入网络的嵌入式设备，可以利用已有的 UDDI 来进行动态的集成入松耦合的软件系统中。例如，一台智能电视机，刚连入网络，他可以到 UDDI 注册中心去找一些他用到的服务，比如当天的节目单等等，之后使用这些服务获得相应的信息，也可以使用这些服务进行远程视频点播等。同时它也可以到 UDDI 注册他提供的服务，比如他可以通过 WEB SERVICE 发布收视率，或者是某广告的收看情况等，其他公司可以通过 UDDI 查找这些服务，之后再调用这些服务来，获得他们想要的数据。

除了“和欣” 嵌入式操作系统的先进的 ezCOM 技术以外，由于“和欣” 嵌入式操作系统是完全国产的、不开源的操作系统，在信息安全要求日益增高的今天，“和欣”也比外国公司的 WINCE、SYMBIAN 等操作系统，或者是开源的 LINUX 系统更适合作为我国家电信息化的基本操作系统。

第四章 WEB SERVICE 协议详解

4．1SOAP 协议

SOAP 的开发工作开始于 1998 年，是为了规范端点之间的 XML 数据传输。起初，这些端点被认为有一个使用其他技术难以远程访问的接口，SOAP 被当作了一种 RPC 机制。然而，人们发现 SOAP 可以用与更广泛的用途，因此在仅仅天的规范中已经不再有这样的限制。

SOAP 的发展曾由于 XML 缺少一种一致的类型系统和描述端点没有适当的元数据格式而受到了阻碍。现在的情况不是这样了——XML 模式已经牢固地建立了，WSDL 已经成为标准的端点描述格式。

SOAP 在 WEB SERVICE 中自然而然地有其一席之地，因为这两个技术的意图是相通的。SOAP 的一个重要特征是它不限制使用什么传输机制。对于 WEB SERVICE 来说，通过 HTTP 使用 SOAP 是很理想的（正如下章我们在“和欣”嵌入式操作系统上实现的那样。）。但是 SOAP 也可以通过局域网，用电子邮件消息，甚至用软盘传输（这些方法现在并没有实用于 WEB SERVICE 中）。另一个要点是，SOAP 消息设计用于单向的数据传送，但可以直接映射为 HTTP 的请求——响应机制。

SOAP 规范还详细讲述了如何使用标准格式传输 SOAP 的错误。如果提供 WEB SERVICE 的 WEB 服务器上发怔错误（可能是由于无效的输入参数，也可能是服务器上的一个必要功能没有发挥作用），那么一个响应会使用此标准格式被发送回消息的创建者。

SOAP 现在的版本是 1.2，可以爱 W3C 的站点上找到。

4．1．1SOAP 的结构

SOAP 消息是由一个信封成，这个信封定义了消息的总体结构。包含在信封中的是消息头信息和消息提信息。消息头和消息提没有在 SOAP 规范中定义，这个信息是特定于使用 SOAP 消息的系统的。错误信息如果没有采用的话，就会被包含在消息体中进行发送。

SOAP 专用的元素和属性名大部分包括在命名空间

http://schemas.xmlsoap.org/soap/envelope/中。基本元素是<Envelope>、<Header>和<Body>.

SOAP 消息的基本结构如下：

```
<?cml version=”1.0”?>
<soap:Envelope xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope/”>
    <soap:Header>
        …header information…
    </soap:Header>
    <soap:Body>
        …body information…
</soap:Body>
</soap:Envelope>
```

这里 SOAP 命名空间的前缀是 soap。

SOAP 消息头部分

SOAP 消息的创建者可以把任意元素放在这个部分，这个部分组成了 SOAP 消息的消息头消息。在处理 WEB SERVICE 时，这个部分也是完全不必要的，因为很多 WEB SERVICEW 跟不需要任何的消息头信息。但是，如果这样做对 WEB SERVICE 比较合适的话，WEB SERVICE 有可能对客户端要求消息头消息。WEB SERVICE 也可以不允许使用消息头信息，但不家强制。例如，为对不同用户开放不同功能的 WEB SERVICE 指定可选定的用户登陆信息。如果 WEB SERVICE 既可以通过 SOAP 访问，又通过 HTTP GET/POST 访问，可选消息头会非常有用，因为它们可以在 HTTP 消息头中发送数据，但不能在 SOAP 消息头中发送。通过 HTTP 向要求 SOAP 消息头的 WEB SERVICE 方法进行请求在其他情况下可能会失败，但这里可以对 WEB SERVICE 进行配置，让它接受两种消息头类型。

SOAP 规范允许通过使用应用于消息头元素的 SOAP 两个专用属性进一步配置消息头。配置包括指定消息头所瞄准的端点类型，消息头是否必须被处理。端点类型的指定（用 actor 属性指定）并不真的应用于 WEB SERVICE，因为端点一般是被瞄准的 WEB SERVICE。指定端点类型就是为了处理 SOAP 消息通过中间设备传送到目的地的情况。中间设备可以是 WEB SERVICE 和外部世界之间的一个网关，例如，它负有使用 SOAP 消息头中路由信息进行路由的责任。

另一个 属性是 mustUnderstand，如果将之设为 1（或 true），那么处理 SOAP 消息的应用程序（比如一个 WEB SERVICE）必须处理这个消息头，或者生成一个 SOAP 错误消息。

关于 SOAP 消息头最后要注意的一件事是，他们必须属于一个命名的命名空间。在 WEB SERVICE 的情况中这很简单：从 WEB SERVICE 本身使用的那个命名空间来或许消息头。

2．SOAP 消息体部分

和消息头部分一样，SOAP 消息提的内容完全取决于在被瞄准的 WEB SERVICE 语义中指定的消息定义。事实上可以说的更具体一点：内容是由和 WEB SERVICE 相关的 WSDL 文挡中定义的。

WEB SERVICE 的 SOAP 消息体可以采用两种格式：文挡或者 RPC 格式。文挡格式中，消息的每个部分直接包含在<soap:body>中。为各部分定义的类型考虑了这一点。

RPC SOAP 消息格式更想 HTTP GET/POST 格式。在这种格式中，消息的每个部分是被调用的操作的一个参数，就和上面显示的各部分一样，只是参数，而不是包含在一个更高层次元素中的参数。但是，对于 RPC 格式的消息，消息的每个部分不应该直接放在<soap:body>中，而是放在被调用的操作同名的一个元素中。

对于 WEB SERVICE，似乎偏向于使用文挡格式的消息。这是因为这样一个事实：从类型定义到 SOAP 消息格式的映射不象 RPC 格式的消息那样隐晦，因为我们对于所发生的事情可以一目了然。

4．1．2 通过 HTTP 使用 SOAP

SOAP 可以通过 HTTP 使用。为此 SOAP 消息被打包为 HTTP POST 请求（因为 SOAP 消息可以很大，因此使用 HTTP GET 不合适）。SOAP1.1 消息使用的 MIME 内容是 text/xml,但 SOAP1.2 版本指定应该使用 MIME 类型 application/soap. 另外，SOAP1.1 还使用了一个额外的 HTTP 消息头 SOAPAction 来命名它瞄准的端点,但是在 SOAP1.2 中这个头是可选的。下面是一个.net 自动打包的 HTTP POST 请求的 SOAP 示例：

```
POST /test.asmx HTTP/1.1
```

Connection: Keep-Alive

Content-Length:437

Content-Type:text/xml;charset=utf-8

Expect:100-continue

Host:localhost

User-Agent:Mozilla/4.0(compatible;MSIE 6.0;MS Web Services Client Protocol 1.0.3705.209)

SOAPAction:http://192.192.2.3/test

<?xml version=”1.0” encoding=”utf-8”?>

<soap:Envelope xmlns:soap=” http://schemas.xmlsoap.org/soap/envelope/ ”

xmlns:xsi=”http://www.w3.org/2002/XML Schema-instance”

xmlns:xsd=http://www.w3.org/2002/XML Schema

…SOAP message…

</soap:Envelope>

SOAPAction 消息头用于指定被调用的 WEB SERVICE 操作，所以这个消息头对于 Web SERVICE 调用来说是必须的。

4．1．3 多部分消息

SOAP 和 HTTP POST WEB SERVICE 请求可以被格式化为多部分消息。也就是说，请求的所有数据不是包括在一个地方，而是使用引用指向其他地方的数据， 这和电子邮件消息中的附件有些相似。在现实中，所谓的其他地方可能就在 HTTP 请求的消息体的内部，因为使用外部位置可能不太实际。

把消息分为所个部分的原因是，有时需要向 WEB SERVICE 发送二进制的数据。例如，发送使用 JPEG 格式压缩的图象，一个声音文件或加密的数据。以 SOAP XML 的形式包装这样的数据可能效率不太高，因为 XML 分析在到达进一步的标记之前，必须处理大量的数据。当然，有些二进制数据格式本身就是高度结构化的，可以使用一个更丰富的 XML 语法重新构建。但这也会引起低效率。比较高效率的做法是不处理这些数据，在别处打包数据，并且只通过对 SOAP 元素或 HTTP POST 参数数据的一个引用来使用它。

为此可以使用两种方法：MIME multipart/related 格式和 DIME（Direct Internet Message Encapsulation,直接因特网消息封装）。事实上，这些方法很相似，都是把二进制数据放在请求体的末尾，在别处编写引用代码引用它们。他们的区别是格式化附件的方法：

MIME 附件有消息头和脚注信息区分每个附件。

DIME 附件有消息头信息指定他们的总长度。所有附件数据在一个块中遵守这个长度。

4．1．4SOAP 错误

如果 SOAP 请求失败。SOAP 消息体可能包含错误信息。处理错误的框架是标准化的，SOAP 规范中也定义了典型环境中使用的一些标准的错误代码。

SOAP 错误的基本结构如下代码所示：

```
<?xml version='1.0'?>
<soap:Envelope xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/ "
    <soap:Body>
        <soap:Fault>
            <faultcode>soap:faultcode</faultcode>
            <faultstring>soap:faultstring</faultstring>
<faultactor>soap:faultactor</faultactor>
<detail>detail fault information</detail>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

一个<soap:Body>元素能包含（且只包含）一个<soap:Fault>元素。该元素中包含一个<faultcode>元素（它包括一个在 SOAP 规范中定义的错误代码），还包括一个<faultstring>元素（给出进一步的错误信息），一个可选的<faultactor>元素（标识生成错误的端点，如果省略该元素，就假设是最后的目的地），一个可选的<detail>元素（包含自定义的细节信息）。SOAP 规范中定义的错误代码如下所示：

错误代码	含义

Soap:DataEncodingUnknown	端点不识别 SOAP 消息体或消息头中的一个或多个元素使用的编码
Soap:DTDNotSupported	SOAP 消息不支持 DTD，所以在 SOAP 消息中发送 DTD 会失败，并生成这个错误代码
Soap:MustUnderstand	消息头发送时属性 mustUnderstand 被设成了 True（或 1），但端点或者不识别消息头，或者不识别消息头的内容
Soap:Receiver	端点发生错误。SOAP 消息本身是好的，但处理消息时，其他错误（如数据库访问错误）引起了未预见到的问题，这时使用这个代码
Soap:Sender	发送到端点的 SOAP 消息或者没有遵守端点要求的规范，或者由于发送者的责任处理失败（也许是身份验证的原因）
Soap:VersionMismatch	SOAP 信封使用了错误的命名空间。可能是由于 SOAP 消息使用了端点不支持的 SOAP 版本。

4．2WSDL 协议

WEB SERVICE 通过一个 URL 访问。但是只知道一个 WEB SERVICE 的 URL 还不足以让您使用它，您还需要知道如何格式化请求，以便让 WEB SERVICE 去做您想让它去做的事情，这要求您了解 WEB SERVICE 支持什么操作。可以把一个操作看成一个方法，要执行一个操作，需要了解给它提供什么参数（如果有参数的话），您会受到什么样的响应（如果有响应的话）。还需要了解（在请求或响应中）被交换的数据采用什么格式，包括要使用的数据类型和顺序。

通过 WSDL 对 WEB SERVICE 的描述，可以了解所有这些信息。

WEB SERVICE 也能以不同方式使用。对于访问 WEB SERVICE 使用什么技术没有限制，对使用的消息格式和（或）协议也没有具体的约束。事实上，如

果愿意，您可以定义自己的做事方式。但是，在 WEB SERVICE 访问方面出现了两个非常有用的定义标准，所以没有任何理由自己创造。这两个标准就是 SOAP 和 HTTP GET/POST，它们分别代表消息格式和协议。

WSDL 允许您指定访问 WEB SERVICE 的多种方式和每种访问方式使用的格式。它还允许把访问的细节和 WEB SERVICE 操作的定义分开。对于 WSDL 的扩展称为绑定扩展，用语您想使用的无论哪种技术，包括 SOAP 和 HTTP GET/POST。

4．2．1WSDL 结构

WSDL是用XML编写的，它的根元素是〈definitions〉。这个元素来自http://schemas.xmlsoap.org/wsdl/的一个命名空间，它构成的WSDL文挡的基本结构如下

<definitions xmlns=”http://schemas.xmlsoap.org/wsdl/”>

…WSDL document…

</definitions>

<definitions>元素可以包含下列元素：

Types - 数据类型定义的容器，它使用某种类型系统(一般地使用 XML Schema 中的类型系统)。

Message - 通信消息的数据结构的抽象类型化定义。使用 Types 所定义的类型来定义整个消息的数据结构。

Operation - 对服务中所支持的操作的抽象描述，一般单个 Operation 描述了一个访问入口的请求/响应消息对。

PortType - 对于某个访问入口点类型所支持的操作的抽象集合，这些操作可以由一个或多个服务访问点来支持。

Binding - 特定端口类型的具体协议和数据格式规范的绑定。

Port - 定义为协议/数据格式绑定与具体 Web 访问地址组合的单个服务访问点。

Service - 相关服务访问点的集合。

大家可以参考下图，来理解一下 WSDL 文档的结构组织：

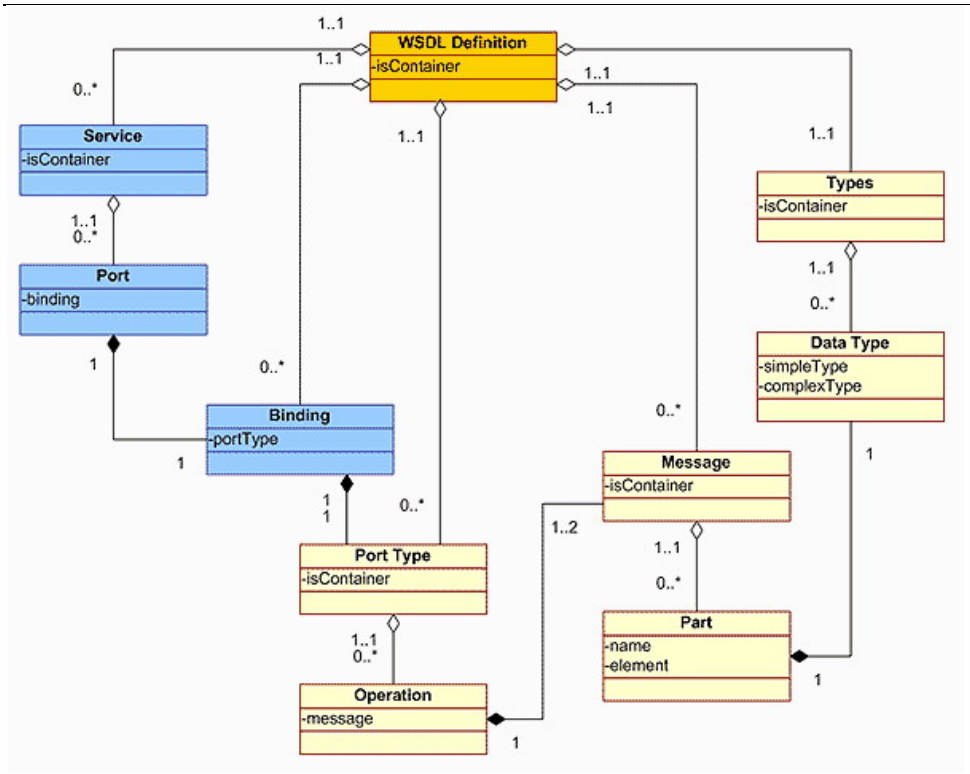


图 4.1 WSDL 元素的对象结构示意图

其中，Types 是一个数据类型定义的容器，包含了所有在消息定义中需要的 XML 元素的类型定义，我将在今后的文章中结合 XML Schema 来详细说明如何进行类型定义。

Message 具体定义了在通信中使用的消息的数据结构，Message 元素包含了一组 Part 元素，每个 Part 元素都是最终消息的一个组成部分，每个 Part 都会引用一个 DataType 来表示它的结构。Part 元素不支持嵌套(可以使用 DataType 来完成这方面的需要)，都是并列出现。

PortType 具体定义了一种服务访问入口的类型，何谓访问入口的类型呢？就是传入/传出消息的模式及其格式。一个 PortType 可以包含若干个 Operation，而一个 Operation 则是指访问入口支持的一种类型的调用。在 WSDL 里面支持四种访问入口调用的模式：单请求；单响应；请求/响应；响应/请求。

在这里请求指的是从客户端到 Web 服务端，而响应指的是从 Web 服务端到

客户端。PortType 的定义中会引用消息定义部分的一个到两个消息，作为请求或响应消息的格式。比如，一个股票查询的访问入口可能就会支持两种请求消息，一种请求消息中指明股票代码，而另一种请求消息中则会指明股票的名称，响应消息可能都是股票的价格等等。

以上三种结构描述了调用 Web 服务的抽象定义，这三部分与具体 Web 服务部署细节无关，是可复用的描述(每个层次都可以复用)。如果与一般的对象语言做比较的话，这部分可以堪称是 IDL 描述的对象，描述了对象的接口标准，但是到底对象是用哪种语言实现，遵从哪种平台的细节规范，被部署在哪台机器上则是后面的元素所描述的。

Service 描述的是一个具体的被部署的 Web 服务所提供的所有访问入口的部署细节，一个 Service 往往会包含多个服务访问入口，而每个访问入口都会使用一个 Port 元素来描述。

Port 描述的是一个服务访问入口的部署细节，包括通过哪个 Web 地址(URL)来访问，应当使用怎样的消息调用模式来访问等。其中消息调用模式则是使用 Binding 结构来表示。

Binding 结构定义了某个 PortType 与某一种具体的网络传输协议或消息传输协议相绑定，从这一层次开始，描述的内容就与具体服务的部署相关了。比如可以将 PortType 与 SOAP/HTTP 绑定，也可以将 PortType 与 MIME/SMTP 相绑定等。

在介绍了 WSDL 的主要元素之后，大家会发现，WSDL 的设计理念完全继承了以 XML 为基础的当代 Web 技术标准的一贯设计理念：开放。WSDL 允许通过扩展使用其他的类型定义语言(不光是 XML Schema)，允许使用多种网络传输协议和消息格式(不光是在规范中定义的这些：SOAP/HTTP，HTTP-GET/POST 以及 MIME 等)。同时 WSDL 也应用了当代软件工程中的复用理念，分离了抽象定义层和具体部署层，使得抽象定义层的复用性大大增加。比如我们可以先使用抽象定义层为一类 Web 服务进行抽象定义(比如 UDDI Registry，抽象定义肯定是完全一致的遵循了 UDDI 规范)，而不同的运营公司可以采用不同的具体部署层的描述结合抽象定义完成其自身的 Web 服务的描述。

4．2．2WSDL 扩展

从上面的讨论可以看出，WSDL 定义如果不和具体的协议一起使用就没那么有意义了。您可以自定义协议，但一般来说，有 SOAP 和 HTTP GET/POST 这两个现成的协议定义就足够了。

要使用这些扩展，只需要引用包含扩展名称的命名空间并在<binding>和<port>元素中使用这些扩展。

SOAP 和 HTTP GET/POST 协议都能够使用消息的 MIME 编码，MIME 使我们能够进一步配置包含传输数据的有效负载。具体来讲，它们允许数据被分成多个部分。SOAP 也训育消息使用 DIME 进行编码。DIME 和 MIME 相似，但比 MIME 功能强大。

HTTP GET/POST 扩展

通过 HTTP 使用简单的语法访问 WEB SERVICE 对于 WEB 应用程序来说非常有用。不必创建一个详尽的客户程序以构建更复杂的 SOAP 消息，可以使用 GET 或 POST 发送一个 HTTP 消息来获取 WEB SERVICE（或者也可以使用其他的 HTTP 动作，但是 GET 和 POST 在这种情况下是最实用的）。GET 更简单一点，因为使用它时只需要在 WEB 浏览器中输入一个 URL 就可以使用 WEB SERVICE 操作了。

HTTP GET/POST 扩展允许您指定端口的基地址，在 URL 中编码什么数据以及数据应该如何出现（对 HTTP GET 而言），请求的有效负载由什么信息组成（对 HTTP POST 而言）。WEB SERVICE 的响应几乎总是用 MIME 格式编码。

包含 HTTP GET/POST 扩展的命名空间是 http://schemas.xmlsoap.org/wdsl/http/.MIME 扩展元素的命名空间是 http://schemas.xmlsoap.org/wdsl/mime/,映射前缀是 mime。

以下是一些 HTTP 扩展元素：

<http:binding>

这个元素在 〈binding〉定义中用来指定为其定义绑定 HTTP 动词，verb 属性指定 HTTP 东西。

<http:address>

这个元素把 URL 和端口关联起来，构成 WEB SERVICE 请求的端点。它把一个 URL 赋给了属性 localtion.

<input>扩展

决定 HTTP GET/POST WEB SERVICE 请求的消息格式的过程涉及到了多个扩展元素的使用，这些扩展元素使数据可以用多种格式传输。这些格式分两类：数据放在 URL 中传输或者数据放在请求的有效负载中传输。

<output>扩展

对 HTTP GET 和 HTTP POST 请求的响应一般采用 MIME 编码各个部分。使用<output>扩展可以指定编码方式。

SOAP 扩展

WSDL 的 SOAP 扩展的命名空间是 http:// schemas.xmlsoap.org/wdsl/soap/，以下是一些 SOAP 扩展元素：

<soap:address>

<soap:address>元素和<http:address>元素的作用相同，都是用来定义端点地址的。和 HTTP GET/POST 相同，也有一个属性 location 用于指定一个 URL

<soap:binding>

SOAP 绑定必须包括一个<soap:binding>元素指明应该使用 SOAP 协议。该元素指定应该如何传输 SOAP 消息（使用必须的 transport 属性）和如何格式化 SOAP 消息体的内容。

<soap:operation>

通过 HTTP 使用 SOAP 访问 WEB SERVICE 和 HTTP GET/POST 相比，指定调用什么服务的方法略有不同。HTTP GET/POST 是在 WEB SERVICE 的 URL 后面追加操作来处理请求。

对于通过 HTTP 使用 SOAP，所有的请求都指向相同的 URL，这个 URL 只是 WEB SERVICE 的地址。调用哪个操作是通过 HTTP 请求的消息头 soapAction 选择的，该消息头的内容将和操作绑定中指定的操作相比较。

<soap:operation>消息头用来控制这个行为。该元素有一个属性 soapAction，它用于指定 soapAction 消息头访问元素<operation>绑定的操作必须使用的值。

<soap:body>

<input> 和<output>元素可以包含<soap:body>元素来指定和 WEB SERVICE 交换的 SOAP 格式的消息。

<input> 和<output>的 MIME 扩展

MIME 可以用于 HTTP GET/POST，也能用于 SOAP 操作。它允许指定 WEB

SERVICE 的 SOAP 响应应包括的 MIME 类型信息，如位图、声音等。消息的这些部分可以被编码并直接嵌入 SOAP 消息体中，或放在附件中使用。

<soap:fault>

此元素用于设置 SOAP 错误响应中包含的<detail>元素的格式。

<soap:header>和<soap:headfault>

这两个元素用来定义和 WEB SERVICE 通信的消息头的格式。

4．3UDDI 协议

UDDI 项目鼓励 Web 服务相互操作和相互采用。它是一种工商界居于领先地位的企业之间的伙伴关系，这种关系最早是由 IBM、Ariba 和 Microsoft 建立的。现在参加的公司已逾 300 家。UDDI 提供了一组基于标准的规范用于描述和发现服务，还提供了一组基于因特网的实现。UDDI 继续快速发展，并赢得了业界的支持。这一规范之所以发展很快，是因为有快速实现在背后提供支持，不仅证明了思想，而且为进一步完善规范提供了丰富的实践基础。

UDDI 解决了企业遇到的大量问题。首先，它能帮助拓展商家到商家（B2B）交互的范围并能简化交互的过程。对于那些需要与不同顾客建立许多种关系的厂家来说，每家都有自己的一套标准与协议，UDDI 支持一种适应性极强的服务描述，几乎可以使用任何接口。对于一家地处澳洲的花店，虽然很希望能进入世界上的所有市场，但苦于不知道怎样才能成功，UDDI 提供了一种能实现这一目标的办法。规范允许企业在注册中心中发布它所提供的服务，这样发现企业及服务就变得高效而且简单了。

对于 B2B 交易场所提供者，他们需要获得这一行业内的供应商的分类数据，以及它们与计费服务、包装商、运输商、保险公司等之间的关系，UDDI 允许动态发现相关的 Web 服务并将其集成到聚合的业务过程中。UDDI 提供一站式搜索有关企业和电子化服务的信息。在 UDDI 中发布企业与服务信息使其它企业能大范围的访问到这些信息。

UDDI 基于现成的标准，如可扩展标记语言（Extensible Markup Language，XML）和简单对象访问协议（Simple Object Access Protocol，SOAP）。UDDI 的所有兼容实现都支持 UDDI 规范。公共规范是机构成员在开放的、兼容并蓄的过程中开发出来的。目的在于先生成并实现这个规范的三个连续版本，之后再

把将来开发得到的成果的所有权移交给一个独立的标准组织。UDDI 版本 1 规范于 2000 年 9 月发布，版本 2 于 2001 年 6 月发布。版本 3 还在开发中，预计到 2002 年年中发布。版本 1 打下了注册中心的基础，版本 2 则添加了企业关系等功能，版本 3 接下去要解决正在进行的 Web 服务开发中的重要领域内的问题，如安全性、改善了的国际化、注册中心之间的互操作性以及为进一步改进工具而对 API 进行的各种改进。

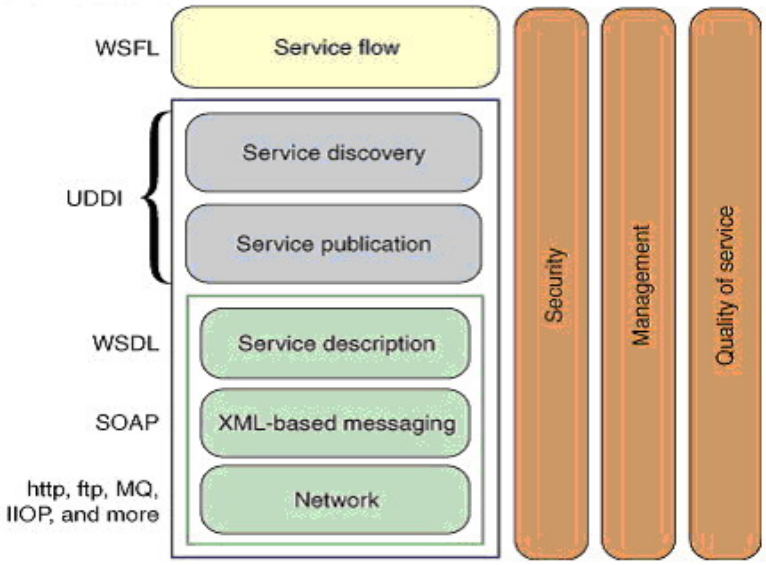


图 4.2 UDDI 的分层 Web 服务协议栈

如图 4.2 中所示，UDDI 包含于完整的 Web 服务协议栈之内，而且是协议栈基础的主要部件之一，支持创建、说明、发现和调用 Web 服务。

UDDI 构建于网络传输层和基于 SOAP 的 XML 消息传输层之上。诸如 Web 服务描述语言（Web Services Description Language，WSDL）之类的服务描述语言提供了统一的 XML 词汇（与交互式数据语言（Interactive Data Language，IDL）类似）供描述 Web 服务及其接口使用。您可以通过添加分层的功能搭起整个基础，比如使用 Web 服务流程语言（Web Services Flow Language，WSFL）的 Web 服务工作流描述、安全性、管理和服务质量功能，从而解决系统可靠性和可用性问题。

UDDI 的工作原理

UDDI 注册中心包含了通过程序手段可以访问到的对企业和企业支持的服务所做的描述。此外，还包含对 Web 服务所支持的因行业而异的规范、分类法定义（用于对于企业和服务很重要的类别）以及标识系统（用于对于企业很重要的标识）的引用。UDDI 提供了一种编程模型和模式，它定义与注册中心通信的规则。UDDI 规范中所有 API 都用 XML 来定义，包装在 SOAP 信封中，在 HTTP 上传输。

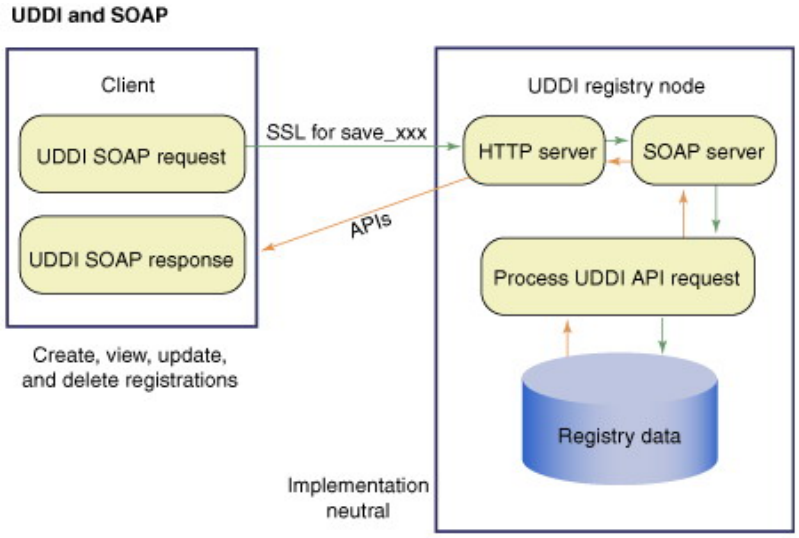


图 4.3. UDDI 消息在客户机和注册中心之间的流动

图 4.3 说明了 UDDI 消息的传输，通过 HTTP 从客户机的 SOAP 请求传到注册中心节点，然后再反向传输。注册中心服务器的 SOAP 服务器接收 UDDI SOAP 消息、进行处理，然后把 SOAP 响应返回给客户机。就注册中心条例而言，客户机发出的要修改数据的请求必须确保是安全的、经过验证的事务。

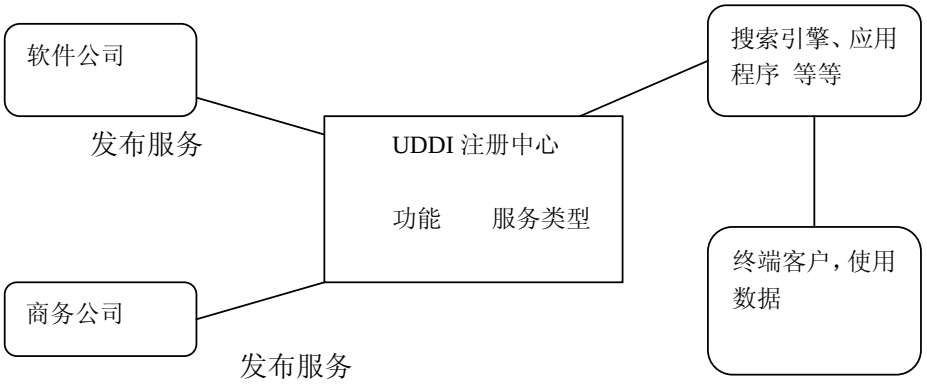


图 4.4. UDDI 工作原理

图 4.4 说明了如何往 UDDI 注册中心送入数据，顾客又如何能发现和使用这一信息。UDDI 注册中心建立在顾客提供的数据的基础之上。要使数据能在 UDDI 中物尽其用需要几个步骤。如第 1 步中所示，在软件公司和标准组织定义关于在 UDDI 中注册的行业或企业的规范时，开始向注册中心发布有用的信息。这些规范叫做技术模型或者更常见的说法是 tModel。

在第 2 步中，公司还会注册关于其业务及其提供的服务的描述。如第 3 步中所示，UDDI 注册中心会给每个实体指定一个在程序中唯一的标识符，叫做唯一通用标识符（Unique Universal Identifier，UUID）键，从而能随时了解所有这些实体的情况。UUID 键必须是唯一的，并且在一个 UDDI 注册中心中从来都不会变化。这些键看上去象格式化好的十六进制随机字符串（例如 C0B9FE13-179F-413D-8A5B-5004DB8E5BB2）。可以利用这些键来引用与之相关联的实体。在一个注册中心中创建的 UUID 键只在该注册中心的上下文中有效。

在第 4 步，诸如电子交易场所（e-Marketplace）和搜索引擎等其它类型的客户机与商业应用程序（例如，基于工作流聚合起来的 Web 服务）使用 UDDI 注册中心来发现它们感兴趣的服务。接着，另外的企业就可以调用这些服务，简便的进行动态集成，如第 5 步中所述。

UDDI 注册中心里的数据从概念上可以分为四类，每一类表示 UDDI 最上层的一种实体。每个这样的实体都指定有自己的 UUID，利用这个标识符总能在

UDDI 注册中心的上下文中找到它：

技术模型（Technical model）

企业（Business）

企业服务（Business service）

服务绑定（Service binding）

我们可以把企业与服务的注册信息分成以下三组：白页、黄页和绿页。

白页表示有关企业的基本信息，如企业名称、经营范围的描述、联系信息等等。它还包括该企业任何一种标识符，如 Dun & Bradstreet D-U-N-S® 号码。

黄页信息通过支持使用多种具有分类功能的分类法系统产生的类别划分，使您能够在更大的范围内查找在注册中心注册的企业或服务。这样的类别划分不仅可以关联企业及其服务，还可以关联 tModel。单提供白页和黄页中的一种或者这两种都提供，那么对于通过程序发现和使用服务，注册中心的条目的价值就很有限。为此，有关怎样、哪里能通过程序的方式调用服务的信息就很有必要了，而绿页就提供了这样的信息。

绿页是指与服务相关联的绑定信息，并提供了指向这些服务所实现的技术规范的引用和指向基于文件的 URL 的不同发现机制的指针。

UDDI 注册中心由 UDDI 规范的一种或多种实现组成，它们可以互操作以共享注册中心数据。有一种特殊的 UDDI 注册中心是由一组对外公开访问的叫做节点的 UDDI 实现构成。它们互操作以共享注册中心数据，合在一起就形成了 UDDI 业务注册中心。该注册中心免费向大众开放。在所有的运营商（Operator）站点上都冗余的放着 UDDI 业务注册中心的全部条目，但只有在创建条目的站点才能对条目进行更改。

第五章 在“和欣”上实现 WEB SERVICE 组件

在本章中，我们将利用 ezCOM 和“和欣”SDK 在“和欣”嵌入式操作系统上实现一个提供 WEB SERVICE 的组件。

5．1 总体架构

“和欣”WEB SERVICE 组件将主要由以下几个主要部分组成：XML 解析器（此处仅实现了一个 DOM 解析器），一个基于 XML 解析器的 SOAP 文挡解析器，一个基于 XML 解析器的 WSDL 文挡解析器和生成器，一个 CGI 作为 HTTP 服务器一个基于 XML 解析器的 WEB SERVICE 配置文挡解析器。总体的 WEB SERVICE 提供如下图所示：

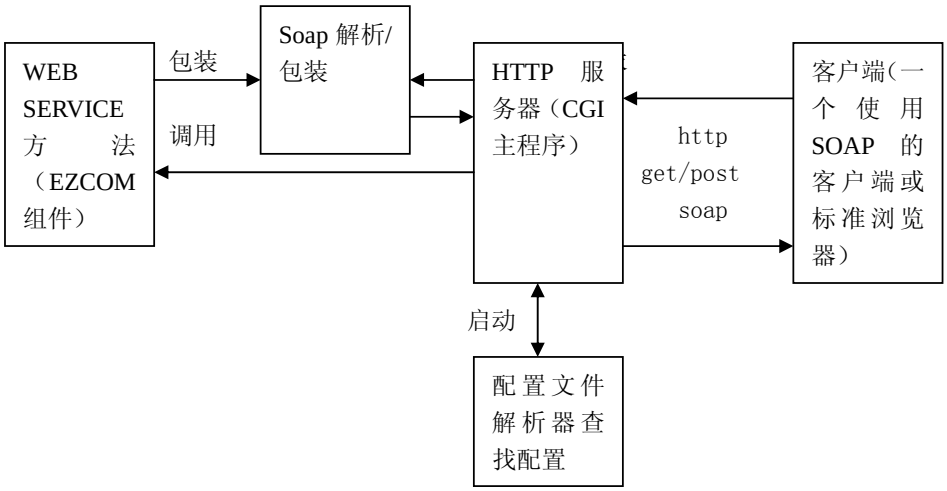


图 5.1WEB SERVICE 提供组件总体结构

整个组件的主程序就是一个 CGI 的 HTTP 服务器，在启动的时候就调用配置文件解析器。配置文件是一个 XML 文挡，用于配置和发布 WEB SERVICE 方法，里面包括了提供 HTTP 服务的端口号，系统中哪些 WEB SERVICE 方法（这些方法都将是一些 EZCOM 组件），他们的调用方式和他们的输入输出参数等。之后就把

这些内容存放在内存中，对配置的 SOCKET 端口进行侦听，等待 HTTP 请求。HTTP 服务器接受的 HTTP WEB SERVICE 请求包括 HTTP GET/POST 和 SOAP 两种，服务器接受到 SOAP 请求之后就通过 SOAP 解析器进行解析，确定调用的是哪个 WEB SERVICE 方法，找到后就调用相应的 EZCOM 组件，输入参数，并且得到结果。将得到的结果用 SOAP 解析器进行包装，包装成一个结果 SOAP 文挡，之后从原端口返回给客户。

WEB SERVICE 使用在本处只提供了简单的 SOAP 接口解析，要进行客户端编程的时候可以使用。而且“和欣”嵌入式操作系统上已经有标准浏览器，只需要改动部分代码，就可以直接使用。

以下就让我们仔细介绍各部分的实现和提供的 API。

5．2XML 解析器

此处我们选定 DOM 解析器进行实现，将暂时不提供 SAX 接口，因为在嵌入式系统上提供的 WEB SERVICE 文档通常不大，所以采用 DOM 不会出现效率问题。

“和欣”DOM API 的实现参照了 W3 上的 DOM 规范，由于是嵌入式系统的关系，对 W3 的开源代码 xercesc 接口进行了裁减和变更。

另在编写 DOM 接口之前，我们参照 MFC 的 String 类编写了一个简单的 String 类，包含最基本的字符串操作，里面的字符串全部使用 UTF-8 存储，具体函数参照 MFC 的 String 类，这里不在冗述。

“和欣”DOM API 的主要类如下所示。

Document 类

Document 对象代表了整个 XML 的文档，所有其它的 Node，都以一定的顺序包含在 Document 对象之内，排列成一个树形的结构,其主要方法如下：

Void initialization(String XMLString)	用一个 XML 文挡初始化 Document 对象.
Attr* createAttribute(String name, String value)	建立一个新的属性结点
Attr* createAttributeNS(String	建立一个带命名空间的属性结点

nspuri, String qname, String value)	
Element* createElement(String tagName)	建立一个新的元素
Element* createElementNS(String nspuri, String qname)	建立一个带命名空间的元素
NodeList* getElementsByTagName(Element *root, DOMString tagname)	在指定的 root 元素下用 Tag Name 查找他的子结点
NodeList* getElementsByTagName(Element *root, String nspuri, String local)	在指定的 root 元素下用 NameSpace 和他的本地名查找他的子结点
String toString()	把整个树转成一个 String,也就是 XML 文件本身。
Element* getRootElement()	得到根元素。

Node 类

Node 对象是 DOM 结构中最为基本的对象，代表了文档树中的一个抽象的节点。在实际使用的时候，很少会真正的用到 Node 这个对象，而是用到诸如 Element、Attr、Text 等 Node 对象的子对象来操作文档。Node 对象为这些对象提供了一个抽象的、公共的根。虽然在 Node 对象中定义了对其子节点进行存取的方法，但是有一些 Node 子对象，比如 Text 对象，它并不存在子节点，这一点是要注意的。Node 对象所包含的主要的方法有：

Node* appendChild(Node *newChild)	给 node 元素增加一个子元素
Node* getChildNode(int index)	返回一个自节点
NodeList* getChildNodes()	返回当前节点的全部子节点
Node* getFirstChild()	返回当前节点的第一个子节点
Node* getLastChild()	返回当前节点的最后一个子节点

String getNamespace()	返回当前节点的命名空间
Node* getNextSibling()	返回当前节点的下一个兄弟节点
String getName()	返回当前节点的名字，如果返回 NULL 则没有名字
String getType()	返回当前节点的属性，是何种节点
String getValue()	返回当前节点的值，如果是 NULL 则说明当前节点没有值
Node* getParentNode()	返回当前节点的父节点
Node* getPreviousSibling()	返回当前节点的上一个兄弟节点
boolean hasAttributes()	判断当前节点是不是有属性
boolean hasChildNodes()	判断当前节点是不是有子节点
Node* insertBefore(Node *newChild, Node *refChild)	在 refChild 自节点前面插入一个子节点，如果 refChild 是 NULL，则插在最后
Int numChildNodes()	计算当前节点有多少自节点
Node* removeChild(Node *Child)	删除子节点
Node *replaceChild(Node *newChild, Node *oldChild)	替换自节点
void setValue(String data)	设置一个节点的 Value

NodeList 类

NodeList 类，顾名思义，就是代表了一个包含了一个或者多个 Node 的列表。可以简单的把它看成一个 Node 的数组，纯粹是为了方便 Node 中的某些函数使用，它的主要方法如下：

Node* item(int index)	返回列表中的第 index 个 Node
Int getLength()	返回列表的长度
void free()	释放整个 NodeList

Element 类

Element 类代表的是 XML 文档中的标签元素，继承于 Node，亦是 Node 的最主要的子类。在标签中可以包含有属性，因而 Element 对象中有存取其属性的方法，而任何 Node 中定义的方法，也可以用在 Element 对象上面。

String getTagName()	返回当前元素的 TagName
String getAttribute(String name)	根据属性的名字取得他的值
Attr* setAttribute(String name, String value)	根据属性的名字设置他的值
void removeAttribute(String name)	根据属性的名字删除属性
Attr* getAttributeNode(String name)	根据属性的名字设置他的指针
boolean setAttributeNode(Attr* newAttr, Attr** oldAttr)	替换或者增加一个属性
Attr* removeAttributeNode(Attr* oldAttr)	删除一个属性
NodeList* getElementsByTagName(String name)	根据 TagName 查找元素
NodeList* getElementsByTagNameNS(String nspuri, String local)	根据命名空间查找元素
String getText()	得到标签元素的文本
Void setText(String text)	设置标签的文本

Attr 类

Attr 类代表了某个标签中的属性，继承于 Node。他的主要方法如下：

String getName()	得到属性的名字
String getValue()	得到属性的值

void setValue(String value)	设置属性的值

这是一个简化的 DOM 解析器，大大减少了代码的复杂度，由于在嵌入式设备中存储空间甚至比内存都宝贵，所以 XML 文件不会很大，因此虽然有一些在大 XML 文件中降低效率的修改，但是在实际使用中不会有影响。

5．3 SOAP 解析器

拥有了 DOM 解析器，我们使用他可以方便的构造一个 SOAP 解析器，用来解析 SOAP 文件，得到客户端的 SOAP 请求以及解析得到 SOAP 中的结果，同时也可以把结果数据，包装成 SOAP 文件，返回给服务器端。

“和欣”SOAP API 的主要类如下所示。

Soapheader 类

Soapheader 类继承自 Document 类，Document 类的方法他都可以使用，Soapheader 类主要是处理 Actor 和 MustUnderstand 两个属性。

String getActor()	获得请求的 Actor
Boolean getMustUnderstand()	获得请求的 MustUnderstand
Boolean setActor(String actor)	设置请求的 Actor
Boolean setMustUnderstand(Boolean mustUnderstand)	设置请求的 MustUnderstand

Soapbody 类

Soapbody 类继承自 Document 类， Soapbody 的处理比较复杂，暂时直接使用 Document 类的方法，只有一个产生错误 SOAP 的方法。

Soapbody* initializationFault (String faultcode,String faultinfo, String faultactor,String detail)	直接用 4 个参数初始化一个 Soapbody 类

Soapclass 类

Soapclass 类继承自 Document 类，Document 类的方法他都可以使用，另外他利用 Soapheader 和 Soapbody 类做一些额外的操作以减轻代码复杂度（Soapbody 类现在还没什么用），Soapclass 类有一些包装用的函数，以减轻编程复杂度。

void initialization(Soapheader header, Soapbody body)	用 header 和 body 初始化一个 Soapclass
Soapheader getHeader()	返回一个 Soapheader 类
Soapbody getBody()	返回一个 Soapbody 类
Void setHeader(Soapheader header)	把 Soapclass 的 header 替换掉
Void setBody(Soapbody body)	把 Soapclass 的 body 替换掉

5．4WEB SERVICE 部署文件

前面已经说过，WEB SERVICE 方法将由 ezCOM 组件的形式编写，那么如何让 HTTP 服务器找到并且动态装载这些 WEB SERVICE 方法，如何确定输入输出参数。另外我们的 HTTP 服务器将运行在什么端口之上，这些都需要人为来进行配置。所以我们定义一个 WEB SERVICE 的部署文件，用来方便在和欣上配置 WEB SERVICE。WEB SERVICE 部署文件的结构如下：

```
<?xml version='1.0' encoding='utf-8'?>
<webservieconfig>
    <domainname>www.elastos.com.cn</ domainname >
<servicesocket>80</servicesocket >
<soapaction uri=' /test/methodname ' >
<dllname>dllname</dllname>
<classname>classname</classname>
```

<function>functionname</function >

<uuid>ec3625a4-88ba-4688-bf67-d9a1f08b44dd</uuid>

<Params>

<param name=’’ type=’’ direction=’in″/>

<param name =’ ’ type=’string’ direction=’in’/>

<param name=’ ’ type=’ ’ direction=’out’/>

……

</Params>

</soapaction>

……..

<webserviceconfig>

其中 domainname 是指本机的域名，servicesocket 就是 HTTP CGI 在哪个端口进行侦听，启动服务的时候会首先检查这一项，默认为 80。

之后可以存在多个<soapaction>每一个对应一个相应的 WEB SERVICE 方法，里面的属性 uri 是相对路径，加上服务器的 IP 或者是域名可以组成整个的 URL，dllname 是指相应的 ezCOM 编译成的 DLL 的名字，这样 HTTP CGI 解析到相应的 WEB SERVICE 方法，就可以直接装载相应的 DLL，但是注意 DLL 一定要注册过，才能被程序正确加载。<classname>是如果这个函数是一个类的成员函数，那么这个类的名必须在这里指出。<function>就是要调用的 ezCOM 的方法。<uuid>是这个方法的 UUID，现在没有什么作用，以后扩展使用。后面的<Params>是，这个函数有几个参数，后面的三个参数是变量名（在实际调用中没有什么用），变量类型和变量是输入还是输出，由于 ezCOM 接口的函数返回值都是 HRESULT，所以没有返回值的设置。

配置上面信息就可以发布一个 ezCOM 作为一个 WEB SERVICE 方法，下面的章节中将有一个例子实际说明了配置方法。

5．5WSDL 组件

UDDI 和 WSDL 是 WEB SERVICE 系列协议的精华部分，正式由于这两个协议的出现，我们才可以由程序动态的使用别人发布的 WEB SERVICE 方法。但是现在 UDDI 还没有和 DNS 一样普及起来，所以我们在“和欣”WEB SERVICE 中的处理

方法是，设置一个 GETWSDL 的 WEB SERVICE 方法，这个方法的作用就是根据配置文件生成在此机器上提供的 WSDL 文件，并且返回给请求的客户端。这样客户端程序就可以根据 WSDL 来查找相应的方法。

5．6HTTP 服务器

这个部件其实是整个在“和欣”WEB SERVICE 组件中的主程序部分，启动分析配置文件、监听网络端口，分析 HTTP 请求，动态调用 ezCOM DLL 中的函数，调用 SOAP 和 DOM 组件把结果包装成 SOAP 返回客户端，都是此部件的功能。

首先我们简单介绍一下“和欣”网络编程接口（elasock API）

“和欣”提供了一组套接口函数用于进行网络编程，统称为 elasock API。与 winsock 一样，“和欣”的 elasock 也是从 BSD socket 发展而来。elasock API 可以实现基本的网络通讯功能，具体用法参见每个函数的文档。elasock 与 winsock 基本兼容。“和欣”提供的 elasock 的错误代码与 winsock 有一些不同，如果您需要使用 elasock 错误代码进行编程。

有了 elasock API 和我们上述开发的部件，利用“和欣” ezCOM 技术，我们可以构造此部件，此部件的运行流程如下：

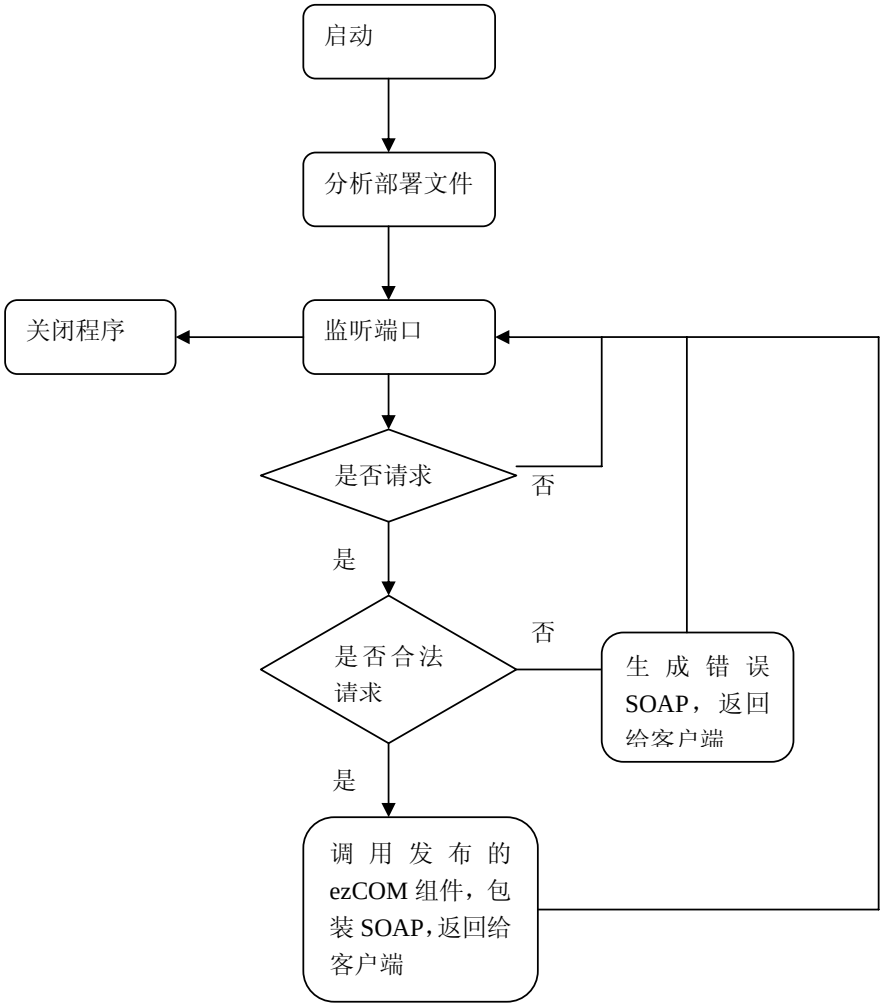


图 5.2 HTTP 主函数流程图

首先主函数启动，分析部署文件，有那些 WEB SERVICE 方法，再找出监听端口，监听 HTTP 请求。此请求包括直接的 HTTP GET/POST 方式的请求和 SOAP 方式的请求，如果检测到合法的请求，就到解析过的部署文件中去寻找相应的 WEB SERVICE 方法，如果没有找到或者出了其他问题，就返回一个 SOAP 错误，返回给客户端。如果一切正常，就根据解析过的部署文件中去找到相应的 ezCOM

方法，找到他，动态加载，运行相应的函数，得到结果，并且把结果包装成 SOAP 协议，返回给客户端。主程序在一个循环内一直监听此端口，一直到程序关闭。

5．7 本章小结

本章利用“和欣”嵌入式操作系统的 SDK 和 ezCOM 原理，构造了一个 XML 和 web service 组件，通过这个组件，我们可以开发一个 ezCOM 作为 WEB SERVICE 方法，再通过在 WEB SERVICE 部署文件中配置此 ezCOM,就可以简单的在 web service 组件中发布此方法。

“和欣”嵌入式操作系统现在可以跑在 x86，ARM 等 CPU 上，并且科泰世纪同济大学基础软件中心的其他同事已经实现了“和欣”嵌入式操作系统对数字电视和对智能手机的裁减，已经使“和欣”嵌入式操作系统顺利的跑在数字电视和手机上。所以我们可以简单的把写好的 ezCOM 作为 WEB SERVICE 方法发布在经过剪裁的“和欣”嵌入式操作系统上，就可以使得数字电视和智能手机拥有直接提供 XML based WEB SERVICE 的方法。换言之，我们可以把他们直接集成入面向服务的软件体系中，真正实现第 3．5 小节提出的使用嵌入式操作系统和 WEB SERVICE 解决嵌入式设备软件集成问题，在一定程度上解决了将嵌入式设备用最小的代价动态集成到分布式软件中来的问题。

第六章 结论与展望

6．1 研究结果

本文介绍了国家 863 重点软件项目“和欣”嵌入式操作系统和其 EZCOM 的主要思想和现在的软件集成技术，分析了 COM、CORBA 和 WEB SERVICE 技术的优劣，得出了 WEB SERVICE 将成为下一代软件系统集成的主要技术的观点。然后详细介绍了 WEB SERVICE 的 UDDI、WSDL 和 SOAP 协议的主要内容。并且利用“和欣”嵌入式操作系统提供的一系列 API 及其 EZCOM 技术实现了一个在“和欣”嵌入式操作系统上部署和提供 WEB SERVICE 的组件，并且仿照一些 OPEN SOURCE 的项目中有关 WEB SERVICE 的 API 提供了一系列在“和欣”嵌入式操作系统上处理 WEB SERVICE 的 API，从而利用“和欣”嵌入式操作系统让嵌入式设备有了直接提供 WEB SERVICE 的能力。

6．2 研究发展

今后基于因特网技术的应用发展趋势是：桌面应用正在向网络应用转移，从网上获得的不仅是一般信息，还包括程序、交互式应用。新的网络应用环境带来了新的运算模型：应用程序由多厂家产品组合而成；跨平台、跨设备的网络应用模式；浏览器成为实现这种组合并且能够相互操作的框架、平台，同时又是一个提供相同的用户体验、智能互动、统一的操作界面。同时，使用这些服务的终端和服务器也已经不再局限于传统的计算机，各种内嵌 CPU 的智能设备越来越多的参与到整个因特网中，成为一个网络结点。而 WEB SERVICE 和嵌入式操作系统的出现则为这种机制提供了便利。“和欣”作为我国 863 重点软件项目，国内先进、国际领先，将在本世纪作为我国智能电器的主要嵌入式操作系统。“和欣”已经在汽车电子、医疗器械、智能手机和工业控制等几个领域进行了使用，获得了重大的成功。“和欣”的 WEB SERVICE 服务模块将作为“和欣”和外界交互的主要方法之一，在今后的 WEB SERVICE 时代将起重大的作用。目前此模块已经接近完成，将在近期发布。并且此模块下一个版本也在设计开发

中，在不久的将来，随着我国信息化进程的加快，很快就会有大量的内嵌“和欣”，使用 WEB SERVICE 的只能电器出现。并且深入我国生产和生活的方方面面，为提高生产力，提高人民生活水平起极大的推动作用。