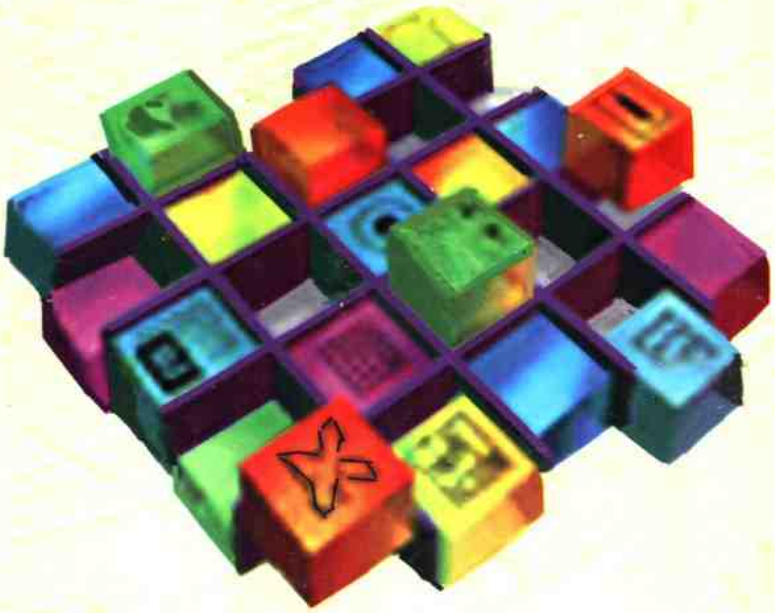
Microsoft Visual Tools 系列丛书



Visual Basic 4.0

客户机/服务器应用程序设计

Microsoft 著

科学出版社
龍門書局

Visual Basic 4.0
客户机/服务器应用程序设计

Microsoft 著
孙 义 陈先才 译
燕卫华 校

科学出版社
龍門書局
1997

内 容 简 介

本书介绍如何使用 Visual Basic 4.0 企业版构造客户机/服务器应用程序，共分三个部分。第一部分介绍 Visual Basic 4.0 企业版特有的工具，描述了小组模型和过程模型的开发方法。第二部分讨论使用 OLE Automation 设计客户机/服务器应用程序的方法和手段，以及客户机/服务器应用程序的设计过程。第三部分介绍数据访问方法，详细说明 Microsoft Jet 数据库引擎、ODBC API、远程数据对象和 RemoteData 控制等编程模型。

本书是介绍 Visual Basic 4.0 企业版客户机/服务器应用程序设计的专著，内容详尽，可供计算机网络集成和软件开发人员参考。

需要购买本书或需要有关技术支持的读者，请与北京 8721 信箱书刊部联系（邮编：100080）电话 010-62562329、010-62541992 或传真 010-62561057。

版 权 声 明

Visual Basic 4.0 客户机/服务器应用程序设计

Microsoft 著

孙 义 陈先才 译

燕卫华 校

责任编辑 王素莲

科学出版社
龙门书局 出版

北京东黄城根北街 16 号

邮政编码：100717

双青印刷厂 印刷

新华书店北京发行所发行 各地新华书店经售

*

1997 年 4 月第 一 版 开本：787×1092 1/16

1997 年 4 月第一次印刷 印张：11 1/8

印数：1～5000 册 字数：260000

ISBN 7-03-005820-8/TP・751

定价：18.00 元

目　录

第一部分　概　述

第二部分　客户机/服务器设计指南

第三部分 数据访问方法

第一部分　概　述

第一章　导言

简要说明划分应用程序在设计灵活性方面的优点，并介绍基于组件开发的企业版功能。

第二章　服务模型

描述把应用程序看成是分成三层的服务及如何使开发者能够设计牢固的客户机/服务器方案，并建议使用服务模型映射应用表单的一种策略。

第三章　企业版中的客户机/服务器工具

给出企业版提供的帮助建造基于组件的解决方案的新工具。

第四章　开发策略

简要说明建造客户机/服务器应用中涉及的管理队伍和过程的两种新方法。

第一章 导 言

企业应用是许多业务的核心，从财务到报表到订单输入，它们管理的信息领域都是操作和策略决策的关键部分。Microsoft Visual Basic 4.0 企业版能够给企业开发人员提供建立这些应用，并且给出比以前更容易、更灵活和更多控制的实现手段。

此外，企业级解决方案要求比传统的客户机/服务器开发提供更多的功能。受不断变化的全球经济的推动，企业应用开发人员致力于在竞争性增强、价格结构转移和消费者品味发展的氛围建立系统。对于减少成本同时保持质量、性能和可用性也越来越重要。

当只按客户和服务器(即系统的物理端点)定义时，客户机/服务器计算在满足这些挑战方面的潜能就变得很有限，但是客户机/服务器潜能比一台台式计算机和一台网络服务器更广泛、更多，它是关于建立和管理分布式任务、业务逻辑和共享的可重用组件。通过分解应用，开发人员可以设计出客户机/服务器使潜能得到最充分发挥，并在业务利润方面回报所需的各种功能。

1.1 应用程序的分解

用 VB 企业版建造分解的应用程序，开发人员可以运用一个对开发过程和结果都有好处的结构。独立的组件可以由分开的项目小组开发，得到更大的生产率，然后实际在多个基于 LAN 的计算系统上布置运行。

通过应用划分，各个开发小组能够协作提高：

- 可重用性。封装到各个组件中的功能可以由许多应用共享和重用。
- 灵活性。工作可以从台式机分布到功能更强的网络服务器，这会有助于解决性能和网络带宽的要求。
- 可管理性。大型复杂的项目可以划分成更简单、更安全的组件项目。
- 可维护性。在集中服务器上部署的而不是在分散的用户台式机上安排的业务逻辑会使修改更容易，且缩短解决方案转变的时间。

应用划分的分布性质提供了开发人员突破“客户和服务器”概念所产生的限制的机会。用户目标、业务目标和面向数据的目标(以及它们之间交织的业务逻辑)可以包含在相互分开的操作的对象中。用这种方法支持的系统及企业就变得很灵活，使它们能够保持与不断发展的业务需求的一致性。

1.2 用 Visual Basic 进行客户机/服务器开发

VB 企业版用于帮助划分应用系统来把这个灵活性加入到企业计算系统中。通过以下方法可以增强应用系统基础的有效性：

- 使用一种三层的而不是两层的逻辑应用体系结构建造对业务逻辑的支持。

· 通过封装的可重用组件提供灵活性。
· 用最适合需要的数据访问方法更有效地访问数据库。

在本书描述的三层逻辑应用模型中，应用功能作为协同工作来满足一个特定需要的服务和协作组件的集合来定义和提供。通过这些集合，各个组件可以相互独立地响应用户、业务或数据服务。

例如，可以运用于管理产品、客户和财政收入数据的策略，是企业应用的中心且是不断变化的规则定义。在三层模型中，一个规则可能存在于这样一个组件中，它使用从数据库提取的信息，执行计算并把结果返回给用户的台式机。该组件占用一个特别设计的运用业务规则，并位于与处理提取数据或与用户交互的编程逻辑分开的概念层。

部署在服务器上，作为一个封装的组件可由任何客户计算机访问，它的操作与数据和用户无关，用户看不到它的工作。这个划分提供用户、业务和数据服务的成分之间的关系。如果规则变化，划分方便了维护，只是提供更改的服务的组件需要改变。

Remote Automation 是关键

在网络上分配各个组件是利用企业最大潜能所必须的，通过将单台机器上的各个组件用于通信的相同 OLE Automation 接口，企业版使机器间的组件通信成为可能。但是，当在网络上通信时，OLE Automation 服务通过 Remote Automation（远程自动化）来响应用户请求。

OLE 和 Visual Basic 的 Remote Automation 特别适合于为企业级客户机/服务器系统开发划分的应用，这有许多原因：

· 因为 Remote Automation 技术提供与本地 OLE Automation 相同的编程接口，划分一个应用比开发本地可执行的应用要求较少的技术手段，这也意味着 Remote Automation 服务器可以很容易地从任何 OLE 客户访问。
· 复杂业务规则的算法用 Visual Basic 比用 SQL 语言能更直接地表述、维护和注释。Visual Basic 还提供测试、调试和控制源代码的更丰富的功能。
· OLE Automation 服务器可以是自己建立的（实现公司特定的业务规则），或从第三方厂商购买（有更通用的服务）。
· OLE Automation 服务器部署在服务器上，它可以从台式机消除大量的计算任务。
· 因为在本地和远程执行的组件间的二进制通信接口是相同的，一个组件可以在机器之间移动，而不必重新编译该组件或它的客户。

Remote Automation 服务器给 Visual Basic 开发人员提供更大的方便性、功能和灵活性，它有一些选件（包括进程内服务器、本地进程外服务器、直接的 SQL 查询和存储过程），作为对企业级客户机/服务器系统划分应用程序的基本工具。

1.3 如何使用本书

本书是用于建造企业级客户机/服务器应用的 VB 企业版功能的指南。要得到完整的知识，应该阅读全书，但是读者在开发过程中工作的性质会使读者集中于特定的章节。

第一部分提供了概念性的指南，这不仅只是针对构造划分的应用程序，而且是对构造开发过程自身，使得开发人员和开发小组可以最好地利用提供的工具。这一部分包括对企业版特有

的工具的概述。

第二部分提供关于使用 OLE Automation 服务器建立更牢固的客户机/服务器设计应用程序的详细信息。这一部分更具体，且面向任务，它讨论设计因素，包括在客户机/服务器设计中建立和使用 OLE Automation 服务器的详细信息。

第三部分提供选择一个远程数据访问模型时要考虑的许多因素的指南。这一部分介绍这些问题，然后讨论如何通过每个编程模型解决它们：Microsoft Jet 数据库引擎、开放式数据库连接(ODBC)API 和远程数据对象(RDO)编程模型及 RemoteData 控制(RDC)。

本书的组织

第一部分：概述

第一章 简介本书所讨论的主题。

第二章 描述把应用程序看成三层的服务及如何使开发人员能够设计牢固的客户机/服务器方案，定义每个服务并建议使用这个服务模型映射应用开发的策略。

第三章 给出企业版中帮助建造基于组件的方案的新工具。

第四章 集中讨论在建立客户机/服务器应用中涉及的管理小组和过程的新方法。

第二部分：客户机/服务器设计指南

第五章 使用一个教学样板应用，讨论定义应用的需要和弄清特定需求的过程。

第六章 描述建立和部署基于组件的应用程序的“前端”逻辑和物理决策的过程。

第七章 描述 Remote Automation 基础结构，并概括说明远程 OLE 服务器的设计考虑、调试和分布，同时还详细讨论 OLE Automation 特定的工具。

第三部分：数据访问方法

第八章 讨论用于访问远程数据库的编程模型，讨论系统设计问题，并介绍与应用程序设计有关的问题。本章还包括讨论的每个数据访问方法的功能提纲表。

第九章 讨论在使用数据访问对象(DAO)和 Microsoft Jet 数据库引擎访问远程数据库时出现的设计和实现问题。

第十章 概述使用开放式数据库连接(ODBC)的远程数据库访问。

第十一章 概述远程数据对象(RDO)编程模型和 RemoteData 控制功能。

第二章 服务模型

Microsoft Visual Basic 4.0 企业版给团体开发人员提供了一整套的工具和设计策略，为业务问题提供牢固、有效的客户机/服务器解决方案。

最有效地使用这些工具和策略的关键是全面理解客户机/服务器解决方案的三层体系结构方法，这种方法稳定地获得了作为团体应用开发的主导模型的力量。这种方法把一个客户机/服务器系统的各个组件分成三"层"服务，这三层必须一起建立一个应用系统：

- 用户服务
- 业务服务
- 数据服务

这三层并不一定要对应于网络上的物理位置，而是概念层次，帮助基于组件的应用程序的设计。以这种方法设计应用程序给程序员提供了在网络上的最佳位置部署过程和数据，以获得最大的性能、安全性和维护的简易性。

本手册把这三层应用模型称为"服务模型"。

内容

- 使用组件的解决方案开发
- 什么是服务模型
- 服务模型与分层模型
- 派生服务
- 下一步的学习内容

2.1 使用组件的解决方案开发

Microsoft 自己的软件开发经验，以及 Microsoft 咨询服务的许多大型团体客户的经验，都表明客户机/服务器应用程序开发的基于组件的方法比传统的应用体系结构有更多优点。

简而言之，组件解决方案把开发小组划分成两个组，一个小组开发对许多应用程序都有用的核心组件(客户定制控制、存储过程、OLE 服务器等)，第二组通过集成由这些组件提供的服务建造业务解决方案。

这种方法比传统的独立应用开发有许多优点：

- 开发人员可以专长于最适合他们的任务，因此培训和再培训费用大大减少。
- 可重用的组件可以用 Visual Basic、C/C++、SQL 或其他工具来写，这样提供了极大的语言和厂商独立性，并允许组件建造者使用最适合特定任务要求的语言或工具。
- 组件可以分配在网络上获得最大的效率、性能、安全性和维护性。单个应用程序可以有位于集中数据库服务器、部门"业务"服务器和终端用户机上的组件，这就允许开发人员分散处理功能和调整应用程序，以适合特定的网络和基础设施限制。各个组件的位

置对终端用户是透明的。

- 可重用性通过建立可以在整个企业的多个应用中使用的通用组件来减轻开发劳动，这些组件的用户只需了解公开的接口，而不必知道它们的内部构造或使用的数据，这就会以更低的代价产生更多的更高质量的应用，同时还提高了应用程序之间高度的一致性、兼容性和业务整体性。

这些优点及其他一些优点使组件解决方案成为客户机/服务器应用开发的优秀方法，但是获得这个方法带来的好处确实要求周密的计划和设计。通常，这个设计过程包括以下阶段：

1. 弄清应用的要求。
2. 把这些要求映射到抽象的业务对象(如客户列表或记帐分类帐)和它们必须提供的服务(如产生帐单)。
3. 把业务对象及其服务映射到软件组件。
4. 决定这些组件如何分布到网络上。

这个设计过程在每一个阶段都是反复的，也就是说，在弄清应用要求中，设计者可能首先找最终用户和业务经理，建立叙述性的使用场景，然后把这个场景分发给用户进行验证反馈。这个过程可能要发生几次，每次反复都会增加细节和精练要求。接下来，陈述性的使用场景会被转换成正式的表示，显示出业务对象和服务，然后这个文档又进行不断的精练，等等。

每一步的特定设计指南在第二部分中给出，本章的其余部分从更加抽象的概念性的观点讨论服务模型。

2.2 什么是服务模型

服务模型是把应用程序看成是一组用于完成客户请求的功能或服务的一种方法，通过鼓励开发人员把应用程序模型化为离散服务的集合，它们可以被包装进行重利用、由多个应用共享和在功能上分布式地分配。

从这种观点来看，一个服务的说明就是服务的提供者与它的消费者之间的合约。按照它的接口指定服务，允许提供者封装实现细节，把使用者与实现环境隔离开来。正是这个合约使重利用成为可能，并且随着时间的推移，会具有一致的企业范围的软件工程法规的优越性。

使用一个 Microsoft 例子，字处理和电子表格应用可以共享一个数据访问组件，使得这两个应用都不必知道数据实际上是如何存放的，存放在哪里。字处理应用也可以调用电子表格服务来操作结构化的信息。使用定制控制、OLE 对象和其他可重用组件，许多服务以“就绪使用”的形式提供给应用构造者。提供这些服务的组件可以内部开发或从外部厂商购买。

服务模型并不与任务特定的产品套装，技术或厂商关联。尽管 Visual Basic 提供了许多工具和技术来支持基于服务的开发，但它可以很容易地与其他工具结合起来，这就使团体开发人员能够把基于服务的组件解决方案开发加入到机构的已存在的或正规划的企业体系中。

一个服务按正规定义就是这样的相关功能的集合，这些功能响应基于公开的接口和行为规范，并通过封装其实现的一致接口，响应对特定的活动的请求和产生结果。

正如在前面所提到的，一共有三类服务，它们有自己的属性，如下表所示：

服务类型	服务属性
用户服务	信息和功能的表示、浏览，用户界面一致性和完整性的保护
业务服务	共享的业务策略、业务信息和数据的生成、业务完整性的保护
数据服务	数据的定义、数据的存储和检索、数据完整性的保护

每种服务都有共同的特性，并提供共同的功能。这种分类还提供构成开发小组和过程的机构映像。小组和过程模型在第四章中讨论。

这些服务通过网络连接在一起，并协作运行支持一个或多个业务过程。在这个模型中，应用成为满足它支持的一个或多个业务过程需要的用户服务、业务服务和数据服务的集合。

因为服务是为通用目的设计的，并且遵循公开的接口，它们可以在多个应用中重利用和共享，如图 2.1 所示。

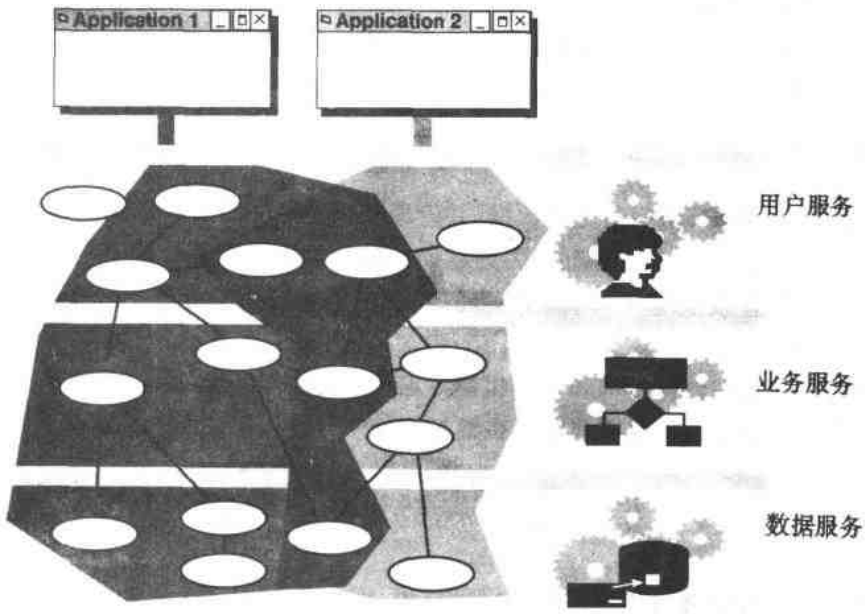


图 2.1 服务可由多个应用并发使用

2.2.1 用户服务

用户服务提供可视化的界面表示信息和收集数据，它们还保护提供要求的业务功能所需的业务服务，并把用户与应用结合起来执行一个业务过程。

用户服务通常用用户界面标识，且常常驻留于最终用户工作站上的一个可执行文件中。但是即使在这里，也有标识驻留于不同组件中的服务的机会。例如，一个演示要求可视化的网格显示数据，且这个网格可能驻留于一个 OLE 控制中。或者，提示用户输入口令获得网络访问权的登录屏幕，可能实际上作为可以连接到某个集中数据库服务器的本地自动化组件驻留。

2.2.2 业务服务

业务服务是用户服务和数据服务之间的“桥”，它们响应用户（或其他业务服务）为了执行一项业务任务而发出的请求。它们通过把正规的过程和业务规则运用于相关的数据来实现这一点。当需要的数据驻留于数据库服务器时，它们保护完成业务任务或运用业务规则所需的数据服务，这就把用户与对数据库的直接交互隔离开来。

一个业务任务是由应用的需求定义的操作，如输入购买订单或打印客户列表。业务规则是控制业务任务流程的策略。例如，为了生成客户发票，给一个库存项目运用特定的百分比加价的过程就是一个业务规则。

因为业务规则要比它们支持的特定业务任务更频繁地改变，它们是封装在与应用逻辑自身实际分开的组件中的理想候选者。

例如，如果一个应用要求的特定服务是“得到提高的标价”，那么把这个服务封装在位于集中网络服务器上且由应用程序访问的组件中就有意义。那么，如果计算提价的过程改变，它可以在单个位置中改变，而不用改动应用自身的用户部分。一旦业务服务被改变，对得到“提高标价”的所有请求得到的回答是修改过的业务服务组件的新结果。

2.2.3 数据服务

数据服务定义、维护、访问和更新数据，并管理和满足业务服务对数据的请求。

这些类型的服务是相当好理解的，因为它们比其他类型的服务分析了更长的时间。它们可以在特定的数据库管理系统（DBMS）中实现，或者由可能位于服务器多个平台及大型计算机上的异构数据库集合来实现。

分开数据服务允许数据结构和访问机制在必要时维护、修改，甚至被重构，而不影响业务服务或用户服务。数据服务的物理实现在第二部分中讨论。

2.3 服务模型与分层模型

三层体系结构有时被看作是一种“分层”模型，因为它标识三个不同的服务层次，然而，有些开发人员倾向于用单个物理层及逻辑层标识每一种服务。

这种物理分层模型意味着严格的交互结构（层到层）并不总是合适的，而且分层模型还意味着每一层对应于一个计算平台的特定开发体系结构：数据服务在数据库服务器上，业务服务在一台或多台业务服务器上，用户服务在台式工作站上。

有两种流行的分层模型：

- 两层模型鼓励划分，但受只由一个客户台式机和一个后端服务器组成的技术平台的影响。
- 三层模型正确地标识三个基础的服务类别，但是隐含着它们的关系是非常结构化的，从用户层到业务层，到数据层和支持链。

这些隐含中没有一个表现在真正的服务模型中，因为服务模型强调逻辑（概念）体系结构而不是强调物理（部署）体系结构。与分层方法相比，服务模型建议任何服务可以物理上驻留于任何地方，满足特定功能要求的任何服务可以激活任何其他服务，这提供了比其他模型倡导的

物理分层模型更高度的灵活性和可用性。

2.3.1 服务模型是逻辑的，而不是物理的

服务模型描述一个应用的概念体系结构，重点是逻辑的，而不是物理的。它表明应用是如何设计的，而不是如何部署。

在开发的实现阶段，设计的权衡很可能把服务类别之间的差别弄得模糊不清。例如，有些业务服务可能在基础数据库管理系统中作为触发器和存储过程实现。

术语"服务"用于描述提供的功能，"组件"用于描述实现中一个或多个服务的封装。

2.4 派生服务

服务模型是把应用设计从需求说明推动到基于组件的应用部署的一种方法。需求映射到服务，服务映射到组件，且组件是构成最终应用的分布在不同服务器和工作站上的实际软件。

对于几乎任何需求，把应用设计划分成几个组成部分或子系统是有意义的。这个过程可以在子系统中以相同的方式重复，直到各个子系统分解成足够小，能够作为问题某个方面的一致定义为止。这些问题往往被标识为称为业务对象的某个东西，即业务感兴趣的一个对象或实体，如客户或订单。需要多少层分解与原始问题的规模有关。

业务对象是逻辑地封装服务以提供业务环境中功能内聚的一种方法，把特定业务对象要求的所有服务组合在一起帮助处理大量服务的复杂性。

服务模型按每个业务对象所请求的功能和它所提供的功能来看待应用设计。在这个上下文中，服务是支持共同活动或产生相关信息的相关特性和功能的集合。

2.4.1 系统接口

为了有效，业务对象和服务的概念要求边界和交互的一种结构化的观点，当它们作为组件实现时，每个接口要求很好地定义与系统其余部分的接口。在逻辑层，这个接口可能不包括物理细节，但包括跨过系统边界的数据流、事件和消息的描述。组件自身被看成一个"黑盒"：标出了输入和输出，但不知道内部发生什么情况。

2.4.2 使用服务模型

以下步骤总结出服务并把它们组织成最终可以作为组件或组件组实现的业务对象的一般过程：

1. 确定业务解决方案必须提供的功能。这个处理的一种较好的初始方法是建立使用场景。
2. 标识由功能需求定义的业务对象和服务，随着后面每一步细节的加入，这个过程将继续进行。
3. 基于三个基本服务类型(用户、业务和数据)对这些服务进行分类。
4. 对于每个服务，确定要求的数据和功能，以及与其他服务的提供者之间的关系和依赖性。
5. 确定服务之间的组织，定义与其他服务通信的接口。

业务对象可以按跨过三种基本服务类型或在单个服务类型内的功能线分组，如图 2.2 和

图 2.3 所示。

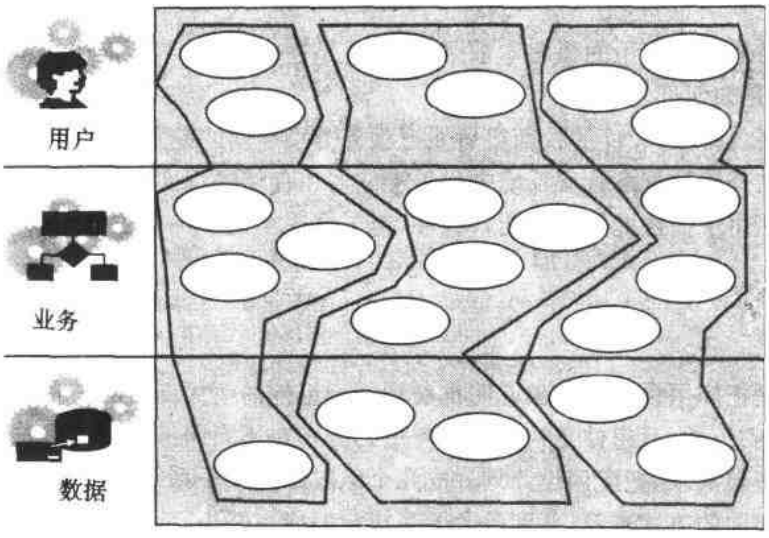


图 2.2 跨类型分组的服务

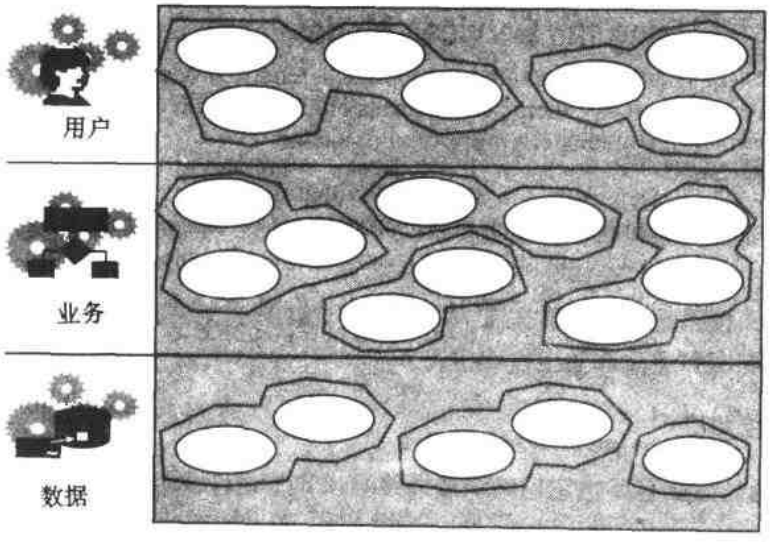


图 2.3 按类型分组的服务

把业务解决方案描述为"服务网络"有助于检查运用于整个方案及它的组成部分的各种实现。服务的描述(因为它们与业务有关)不应受任何特定技术的约束,尽管技术可能影响考虑如何描述它们的方式。但是,选择如何按组件的观点最终实现对业务问题的解决方案与技术有相当的依赖关系。

在这个阶段,要理解的关键思想是一个业务问题可以被分解成若干个服务,这些服务又变成相互作用形成应用的组件。

2.4.3 标识服务:一个例子

通过一个典型的情景为例使这个讨论更加具体。首先从可能组成一个业务问题的解决方案的高层业务对象开始。以下的讨论使用一个简单的订单处理问题作为例子。

有许多可用的方法和表示法可以用于模型化以下例子,但是在这里并不考虑使用特殊表示法。为了图示的缘故,使用一种简化的表示法表达主要思想;在实际应用中,应使用自己最喜欢的设计表示法。

首先简单地描述一个业务问题,然后描述怎样来解决它。在这个例子中,业务问题是接受和处理客户的订单。通过与系统的用户交谈勾划出一个陈述性的使用场景,在几次会谈之后,解决方案的一个描述可以是这样的:

"一个客户与我们联系并订购一个产品,我们给客户寄出产品并更改库存数量。在寄出产品后,生成一张发票并寄送给客户。"

在分析这样的一个陈述性使用场景中,常常发现名词可能直接映射到业务对象,而动词映射到服务。在这个(高度简化的)例子中,名词是

- 客户
- 产品
- 库存
- 发票

而相关的词是

- 订单
- 发货
- 更改
- 生成
- 寄送

图 2.4 表示使用这个名词—动词分析的问题/解决方案。

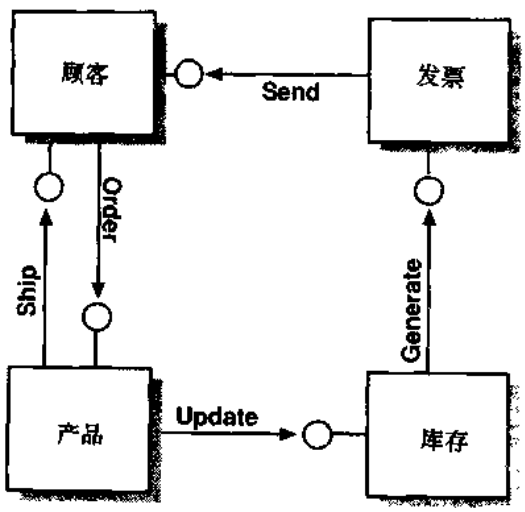


图 2.4 高层业务对象及其相互关系

接下来，开始标识每个业务对象要求什么特性和功能(作为功能说明的一部分)。从动词列表(以及背景知识)可以初步标识每个业务对象的主要功能，如图 2.5 所示。

对这些功能的每个方面(以及其他功能)有更低层的细节，但一般来说可以把这些指定为满足业务(处理订单)要求的服务。

这是开始的好地方，可以采用这些服务集并建立初始的子系统边界。注意，到目前为止主要标识了业务服务，在功能说明中其他的需求会指出什么类型的显示和输入功能需要作为用户服务提供。数据服务被派生来特别支持业务服务。一个完成后的"服务网络"和子系统如图 2.5 所示的逻辑服务模型(数据服务层被压缩，以避免混乱)。

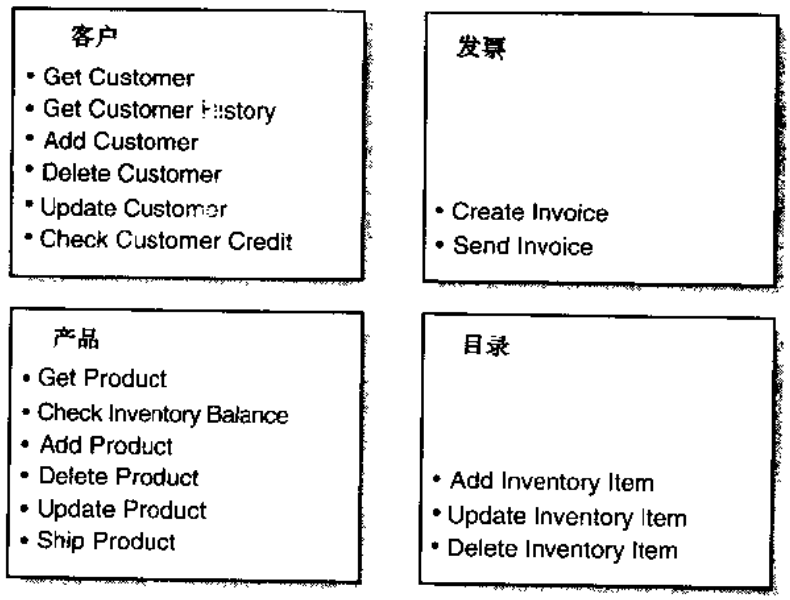


图 2.5 与业务对象关联的服务

注意这个例子产生了五个大的子系统，每个子系统含有相互作用的许多服务。这是一个功能划分，不一定意味着有五个分开的物理组件。从逻辑服务模型到具体的组件定义和实现是本书第二部分讨论的主题。

2.4.4 服务如何相互作用

如图 2.6 所示，一个服务可以调用其他服务来完成某个动作或信息请求。

使用者和提供者

正如前面所提到的，服务间相互交互的一个有用范例是使用者/提供者的概念。

服务是对其用户的提供者，而服务的用户则是使用者。一个设计得很好的服务既是它的用户的提供者，又是其他服务的使用者。

每个服务都有一个很好定义的接口，使它所支持的功能其他使用者也能使用。

重利用一个服务的机会相对于有提供者——使用者关系的任何特定服务独立地存在。一旦开发好后，封装服务的一个组件可以在 Visual Basic Component Manager 中登记，进行检查