C++ Builder 6.0 COM 程序设计

田丰 胡湘渝 吴成勇 等 编著 郝文化 审

- 多个详细的 COM 应用程序开发的实例
- 大量的开发源代码
- 详细的 COM 应用实例
- 详细的 DCOM 应用实例
- 详细阐述 COM/DCOM 应用程序开发的特点和难点

国防工业出版社
http://www.ndip.cn

C++ Builder 6.0 COM 程序设计

田丰　胡湘渝　吴成勇　等编著

郝文化　审

国防工业出版社

·北京·

内容简介

本书采用基础知识讲述、任务驱动的方法介绍了进行COM应用程序开发的理论、方法和步骤。本书共分9章，不仅重点介绍了COM模型的形成、COM理论的基础知识，还通过在C++ Builder 6.0下编写众多实用性强的实例介绍了符合COM标准应用程序的开发过程。

本书主要面向使用C++ Builder进行软件开发的中、高级用户，全书实例丰富、语言通俗、叙述深入浅出、实用性强，既适合作为软件开发人员开发COM应用程序的参考资料，也可作为C++ Builder用户的学习资料。

图书在版编目(CIP)数据

C++ Builder 6.0 COM程序设计/田丰等编著 .—北京：国防工业出版社，2003.10

ISBN 7-118-03208-5

Ⅰ.C… Ⅱ.田… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2003)第059902号

国防工业出版社出版发行

(北京市海淀区紫竹院南路23号)

(邮政编码 100044)

北京奥隆印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 19¾ 449千字

2003年10月第1版 2003年10月北京第1次印刷

印数：1—3000册 定价：27.00元

前　言

一般情况下，用户使用的软件是编译后生成的单个二进制文件（即.exe 文件）。在 COM 模型出现以前，如果要修改某个已有软件，必须先修改该软件的源代码，然后重新将这个软件进行编译，生成新的软件。通过这种方法编制大型软件时将发生许多问题，以 Windows 2000 操作系统为例，如果在命令行中运行“ver”命令，将会看到如下字样：

Microsoft Windows 2000 [Version 5.00.2195]

这里的数字 2195 表示这台计算机上使用的 Windows 2000 操作系统经过了 2195 次编译。在这样的大型软件中不可避免地要出现 bug（缺陷），如果为了弥补所有的 bug，需要修改软件的源代码并重新编译整个软件，那么这个软件的成本将迅速增加。

软件开发者为了解决这个问题提出了很多方案，其中最著名的就是组件化的程序设计方案。其核心思想是：将复杂的应用程序设计成为一些小的、功能比较单一的组件模块，这些组件模块可以运行在同一台计算机上，也可以运行在不同的计算机上，甚至可以在两台间隔很远的计算机上运行。在整个应用程序发布之后，如果需要进行修改，只需要重新编译相应的模块，然后将重新生成的模块组合到应用程序中。按照组件模型设计时提出的理想状况，尽管每台计算机具有不同的硬件和软件环境，（既可以是基于多种芯片，如 Intel 和 SRAC，也可以是安装在不同的操作系统中，如 Windows 和 Unix），组件程序一样可以稳定地运行。要想实现这样的应用软件，必须制定出一系列的规范，只有程序的每个组件都共同遵守了这些规范，软件系统才能够正常地运行。在这样的需求之下，出现了两种并行发展的标准——COM（Component Object Model，组件对象模型）和 CORBA（Common Object Request Breaker Architecture，公共对象请求中断结构），前者是由微软公司提出的，而后者则是由 OMG（Object Management Group，对象管理组织）提出的。目前，CORBA 的运用一般是在 Unix 操作系统之上，而 COM 则运用在微软的 Windows 系列操作系统之上。

微软提出的组件对象模型标准完全符合组件化程序设计思想，并且采用了面向对象的程序设计方法加以实现。面向对象技术经过上个世纪的不断发展已经相当成熟了。从上个世纪末的面向对象语言到现在的面向对象软件设计方法，面向对象的思想已经广泛渗透到计算机软件科学的各个领域。由于近年来网络技术的飞速发展，使得软件的应用环境出现了翻天覆地的变化，应用环境的复杂化对应用软件提出了更高的要求，从而使得软件设计的难度相对增大。在这种情况之下，面向对象的思想已经很难适应这种分布式的软件模型，组件化的程序设计思想的出现为程序设计思想注入了新的生命力，这就是近年来程序设计组件化得以快速发展的原因。本书从介绍 COM 的基本原理和基础知识入手，详细讨论如何建立符合 COM 规范的应用程序。读者在学习本书的过程中，应主要学其面向对象式组件化开发程序的设计思想。

主要内容

本书详细阐述了 COM 规则的实现机制，以及利用 C++ Builder 6.0 编制 COM 应用程序的步骤和方法，并通过典型实例进行了详细剖析。此外，本书在最后的几章着重介绍了操作系统中的一些新增 COM+标准和开发 COM+应用程序的方法。

本书特点

全书从 COM 技术的基础知识出发逐渐深入，紧紧围绕如何利用 C++ Builder 6.0 进行符合 COM 标准的应用程序开发这一核心问题，通过丰富的程序实例，向读者展示了如何在 Windows 操作系统下进行 COM 应用程序开发的方法和步骤，并对 COM 模型的机理和特性进行了深入而全面的介绍。

另外，本书图文并茂、实例众多，并且所举实例都具有很强的针对性，分析透彻，突出了本书以实例为中心的特点。相信通过阅读本书，读者会加深对 Windows 平台下利用 C++ Builder 6.0 进行编程的理解，提高编程的技巧。

适应对象

本书语言通俗易懂，内容丰富详实，突出了以实例为中心的特点，适合具有一定 C++ 基础知识和 C++ Builder 6.0 编程经验，并对 COM 模型有所了解的中、高级开发人员学习使用，同时也可作为在 Windows 系列操作系统上从事符合 COM 标准的应用程序开发的软件工程师的参考用书。如果读者是 C++ Builder 6.0 的初学者，请先学习一些关于 C++ Builder 6.0 编程的基础知识。

编写分工

本书由田丰、胡湘渝、吴成勇编写。在全书的编写过程中，得到了广大同仁的大力支持和帮助：

刘宇提供了许多关于 COM 的第一手资料，并对本书的内容做了详细的检查，对本书的代码部分进行了优化；胡湘蜀博士审阅了本书的样稿，并提出了许多宝贵的意见；胡晓兵教授、李双跃、罗晓宾、成尔京、龙洪能、赵莉香、舒彬和高伟等博士为本书的编写提出了建设性的意见和建议；罗伟为本书的编写提供了大量的参考意见；吴庆阳博士为本书提供了部分资料，并校阅了本书；另外参加本书编排的还有：史伟、季立江、杨大磊、李永胜、江书勇、方辉、蒋捷峰、余良忠、陈嘟、王膺宇、孙海滨、赵翔、廖正军、郭成超、赵龙、蒋强、钱峰、王海田等。在此向所有参加本书编写和关心本书编写进程的同仁表示衷心的感谢。

由于编写时间仓促，编者水平有限，书中疏漏之处在所难免，欢迎广大读者和同行批评指正。

如果读者愿意参加“C++ Builder 6.0 COM 程序设计”的学习培训，或是在学习过程中发现问题，或有更好的建议，欢迎致函。同时，我们也非常愿意随时同网络高手保持联系，E-mail：hwhpc@163.com。我们将认真、负责地对待每位读者的来信。

编著者

目　录

第 1 章　COM 概述 …… 1
1.1　COM 基本概念和基本术语 …… 1
1.2　COM 的起源 …… 2
1.2.1　OLE 和组件技术 …… 2
1.2.2　COM——面向对象组件模型 …… 4
1.2.3　COM 的发展 …… 5
1.3　组件技术和接口 …… 7
1.3.1　组件的特点 …… 7
1.3.2　组件的规则 …… 9
1.3.3　组件的分类 …… 9
1.3.4　接口的特性 …… 13
1.3.5　接口规则 …… 14
1.3.6　接口的设计和实现 …… 16
1.4　COM 的特性 …… 18
1.4.1　可复用性 …… 19
1.4.2　与语言无关性 …… 19
1.4.3　对进程的透明性 …… 20
1.5　COM 和面向对象技术 …… 21
1.6　COM 的利与弊 …… 22
1.6.1　COM 的优点 …… 22
1.6.2　COM 的局限性 …… 23
1.7　程序实例 …… 24
1.7.1　程序 1:DDE 技术 …… 24
1.7.2　程序 2:模拟接口 …… 29
1.8　本章小结 …… 33
第 2 章　COM 的实现规则 …… 34
2.1　COM 编程基础 …… 34
2.1.1　组件对象模型基础 …… 34
2.1.2　对象复用 …… 39
2.2　如何管理 COM 对象 …… 40
2.2.1　COM 组件在注册表中的信息 …… 40

2.2.2 注册 COM 组件的操作 …… 45
2.3 类工厂 …… 46
2.3.1 类工厂简介 …… 47
2.3.2 类工厂和 COM 库 …… 48
2.3.3 类工厂对组件生存期的控制 …… 50
2.4 COM 库 …… 51
2.4.1 COM 库的初始化 …… 51
2.4.2 COM 库的内存管理 …… 52
2.4.3 组件程序的装载和卸载 …… 55
2.5 本章小结 …… 57
第 3 章 开发 COM 客户应用程序 …… 58
3.1 COM 客户程序概述 …… 58
3.2 引入类型库信息 …… 60
3.2.1 使用 Import Type Library 引入类型库信息 …… 61
3.2.2 使用 Import ActiveX 引入类型库信息 …… 63
3.2.3 分析引入类型库信息后生成的代码 …… 64
3.3 控制引入的对象 …… 65
3.3.1 使用组件包装器 …… 65
3.3.2 创建包含数据感知组件的程序 …… 67
3.3.3 程序 1:建立包含数据感知控件的应用程序 …… 67
3.3.4 使用数据感知的 ActiveX 控件 ……
3.3.5 程序 2:创建包含 ActvieX 控件的应用程序 …… 71
3.3.6 如何编写基于类型库定义的代码 …… 74
3.3.7 为没有类型库的服务器创建客户程序 …… 77
3.4 本章小结 …… 78
第 4 章 COM 服务器对象 …… 79
4.1 COM 的服务器程序和客户端的联系 …… 79
4.2 创建 COM 服务器对象 …… 80
4.3 创建 COM 对象概述 …… 81
4.4 设计 COM 对象 …… 81
4.5 使用 COM 对象生成向导 …… 82
4.5.1 COM 对象生成向导使用指南 …… 82
4.5.2 为 COM 对象制定不同的线程模式 …… 87
4.6 指定 ATL 选项 …… 85
4.7 定义 COM 对象接口 …… 87
4.7.1 给对象接口添加属性 …… 87
4.7.2 给对象接口添加方法 …… 88
4.7.3 提供客户事件 …… 88
4.7.4 管理自动化对象的事件 …… 89

4.8 自动化接口 …… 89
4.8.1 双向接口 …… 90
4.8.2 派发接口 …… 90
4.8.3 自定义接口 …… 90
4.9 调度数据 …… 91
4.9.1 自动化兼容的类型 …… 91
4.9.2 自动化调度的类型限制 …… 91
4.9.3 自定义调度 …… 92
4.10 程序实例 …… 92
4.10.1 程序1:Inproc服务器 …… 92
4.10.2 程序2:COM服务器程序 …… 100
4.11 本章小结 …… 109
第5章 COM应用技术背景分析 …… 110
5.1 可连接对象 …… 110
5.1.1 客户与可连接对象的关系 …… 111
5.1.2 可连接对象的基本结构 …… 112
5.1.3 客户方的基本结构 …… 113
5.2 可连接对象的实现 …… 114
5.2.1 枚举器 …… 114
5.2.2 源对象和IConnectionPointContainer接口 …… 115
5.2.3 链接点和IConnectionPoint接口 …… 116
5.2.4 建立链接过程 …… 117
5.3 客户、源对象和接收器的协作过程 …… 118
5.3.1 接收器的实现 …… 118
5.3.2 事件的激发和处理 …… 120
5.3.3 与出向接口有关的类型信息 …… 121
5.4 结构化存储 …… 121
5.4.1 结构化存储基础 …… 122
5.4.2 存储对象和流操作 …… 124
5.5 结构化存储的特性 …… 127
5.5.1 访问模式 …… 128
5.5.2 事务机制 …… 130
5.5.3 命名规则 …… 131
5.5.4 增量访问 …… 132
5.6 命名和绑定技术 …… 133
5.6.1 名字技术基础 …… 133
5.6.2 IMoniker接口 …… 135
5.6.3 复合名字对象 …… 141
5.7 统一数据传输 …… 146

5.7.1 统一数据传输基础知识 …… 146
5.7.2 数据传输机制 …… 149
5.8 本章小结 …… 155
第6章 DCOM …… 156
6.1 DCOM的基本结构 …… 157
6.1.1 从COM转向DCOM …… 157
6.1.2 DCOM对象的定位 …… 158
6.1.3 列集 …… 159
6.1.4 DCOM特性 …… 160
6.2 对象激活 …… 161
6.2.1 创建DCOM组件 …… 161
6.2.2 远程创建进程内组件:代理进程 …… 163
6.3 连接管理 …… 164
6.3.1 有效地控制远程对象的生存期 …… 164
6.3.2 pinging机制 …… 165
6.3.3 连接点管理 …… 165
6.3.4 连接传递 …… 166
6.4 并发管理 …… 166
6.4.1 线程模型 …… 166
6.4.2 消息过滤器 …… 169
6.5 DCOM安全模型 …… 170
6.5.1 安全性策略 …… 170
6.5.2 安全性配置 …… 172
6.6 程序实例 …… 173
6.6.1 程序1:使用DCOM …… 173
6.6.2 程序2:DCOM实例 …… 176
6.7 本章小结 …… 185
第7章 自动化对象 …… 186
7.1 自动化概述 …… 186
7.1.1 自动化的产生和发展 …… 187
7.1.2 属性和方法 …… 187
7.1.3 类型库和ODL …… 188
7.1.4 IDispatch接口 …… 188
7.1.5 自动化兼容的数据类型 …… 190
7.1.6 参数顺序、可选参数和命名参数 …… 192
7.1.7 IDispatchEx接口 …… 194
7.2 自动化对象的实现 …… 195
7.2.1 类型库支持 …… 196
7.2.2 Invoke函数实现 …… 196

7.3 异常处理 …… 197
7.4 程序实例 …… 199
7.4.1 程序1:Variant 变量 …… 199
7.4.2 程序2:创建 OLE 自动化服务器 …… 202
7.4.3 建立 Office Word 自动化应用程序 …… 208
7.5 本章小结 …… 213
第8章 ActiveX 技术 …… 215
8.1 概 述 …… 215
8.2 ActiveX 控件结构 …… 217
8.2.1 ActiveX 控件中的成员 …… 217
8.2.2 ActiveX 控件的许可证机制 …… 219
8.2.3 用 ActiveX 控件构建块 …… 220
8.2.4 属性和方法 …… 221
8.2.5 ActiveX 控件注册信息 …… 224
8.3 属性页 …… 225
8.3.1 属性页和属性表 …… 225
8.3.2 通过 IPropertyNotifySink 实现数据绑定 …… 228
8.4 容 器 …… 229
8.4.1 必需的接口 …… 229
8.4.2 可选方法 …… 230
8.5 分发 ID …… 230
8.5.1 事件处理 …… 231
8.5.2 永久特性 …… 231
8.6 ActiveX 技术中的接口 …… 231
8.6.1 IOleObject 和 IOleClientSite 接口 …… 231
8.6.2 IOleControl 和 IOleControlSite 接口 …… 233
8.6.3 自定义 ActiveX 控件的接口 …… 237
8.7 ActiveX 和 Internet …… 237
8.8 ActiveX 控件实例 …… 238
8.8.1 编写 ActiveX 控件 …… 238
8.8.2 在程序中调用测试控件 …… 249
8.8.3 将控件发布到 Internet …… 251
8.8.4 创建一个数据感知的 ActiveForm …… 255
8.8.5 总结和提高 …… 257
8.9 本章小结 …… 257
第9章 COM+ …… 258
9.1 COM+概述 …… 258
9.1.1 COM+应用程序的概念 …… 259
9.1.2 COM+应用程序的类型 …… 259

9.2 开发 COM+程序需要完成的任务 …… 260
9.2.1 设计 COM 组件 …… 260
9.2.2 创建 COM+应用程序 …… 260
9.2.3 管理 COM+应用程序 …… 261
9.3 Windows DNA …… 261
9.4 组件服务配置 …… 262
9.5 设计 COM+应用程序 …… 263
9.5.1 COM+设计理念和原则 …… 263
9.5.2 设计 COM+应用程序的基本要素 …… 265
9.6 COM+提供的服务 …… 265
9.6.1 上下文 …… 265
9.6.2 事务处理 …… 267
9.7 Windows 2000 中 COM 的新特性 …… 273
9.7.1 同步机制 …… 273
9.7.2 异步 COM …… 275
9.7.3 管道 …… 276
9.8 程序实例 …… 283
9.8.1 程序 1:无名管道 …… 283
9.8.2 程序 2:有名管道 …… 284
9.8.3 程序 3:COM+应用程序 …… 291
9.9 本章小结 …… 302
参考文献 …… 303

第 1 章 COM 概述

内容点击

- COM 的基本概念
- COM 的起源和发展
- 组件技术
- 接口规则及其实现
- COM 的特性
- COM 和面向对象技术
- COM 的优劣

本章导读

本章的重点在于介绍 COM 的基础知识和基本的概念，从介绍 COM 定义和相关术语开始，详细讲述了 COM 的起源及发展。对于 COM 技术中的一些重要的背景知识和相关的支持技术作了基本讨论并给出了一些实例。最后，详细阐述了 COM 的特性及在利用高级语言实现 COM 应用程序时必须遵循的规则。

1.1 COM 基本概念和基本术语

COM（Component Object Model）是一个外来词汇，在当前的软件行业当中通常被译为“组件对象模型”，它是一种以软件组件为单元的软件开发模型。建立这种模型的目的是可以以组件的方式进行软件的开发和应用，并且各个组件之间通过统一的方式进行相互调用。COM 提供了目前惟一一种可行的和可复用的服务器端应用程序与客户端应用程序的组件标准，并包含了极其丰富的集成服务。COM 规则的出现不仅为组件之间的信息交互制定了规范，同时也是实现这种交互的技术支持。COM 是与二进制相兼容的组件

结构模型，它独立于任何语言之外，其本质作用是提供组件之间的交互规范和准则，不受特定编程语言的影响，因此 COM 还可以作为跨语言平台协同开发的标准。

COM 技术是一种比较新颖的软件技术，其中有很多新的术语和概念，本章一开始将重点介绍 COM 规则的几个核心的术语和概念，在以后的章节当中，还将有很多地方涉及 COM 规则具体应用的细节问题，并将详细讲述如何利用 C++ Builder 6.0 进行应用程序开发，在讲述的过程中还会继续介绍其他与 COM 相关的术语。

(1) 类（Class） 就是将相关的一组数据和一些特定的功能组合在一起的一个特殊函数。COM 中类的概念和面向对象技术中类的概念既有区别又有联系。关于 COM 和面向对象技术之间的细节问题请参阅 1.5 节“COM 和面向对象技术”。

(2) 对象（Object） 与 C++中对象的概念类似，对象是某一个特定类的一个实例。

(3) 接口（Interface） 一组在逻辑上相关的函数。接口的功能是使任何程序能够访问这些函数，是组件向程序和其他组件以及应用程序提供的一组公用的功能定义。接口的函数也称作接口成员函数，接口技术的细节问题请参阅 1.3 节“组件技术和接口”。

(4) 全局惟一标志符（GUID，Globally Unique Identifier） 一个 128 位的整数，每生成一个组件的接口，就会产生一个惟一标志该接口的 GUID，并向操作系统和其他软件标识这个组件接口。组件或接口有任何改变时，都会产生新的 GUID。

(5) 类标志符（CLSID，Class Identifier） 又叫做类 ID，也是一个 128 位的 GUID。客户程序使用 CLSID 来标志 COM 对象。使用 CLSID 标志对象在全局范围内具有惟一性。

(6) 组件代码（Component Code） 是组件进行的实际工作，建立组件基础结构之后，进行组件代码的编制以实现组件功能。

(7) 二进制兼容性（Binary Compatibility） COM 完全符合二进制标准，所以，不论是用什么语言编制的 COM 组件，都可以和其他的 COM 组件兼容，并可供其他软件使用。例如，某进程间通信组件由 VC++开发，而接口组件由 C++ Builder 6.0 开发，两者之间可以相互使用。

1.2 COM 的起源

随着微软 Windows 操作系统平台的推广，越来越多的程序员转入到 Windows 下的编程环境。随着软件行业的迅速发展，为了满足用户在短时间内拥有符合自身特殊要求的应用软件的需要，许多软件公司开始提供 Windows 系统下的快速应用程序开发（RAD，Rapid Application Development）工具。这些工具提供了以事件驱动进行编程的简单程序开发模型，程序员不再需要学习大量复杂的编程语言和编写大量的代码，就可以完成 Windows 应用程序的开发，应用程序的建立时间比起以前可以大大缩短。

1.2.1 OLE 和组件技术

COM 技术是从 OLE（Object Linking and Embedding，对象链接和嵌入）技术中发展起来的。要介绍 COM 的起源，就不得不先从 OLE 技术谈起。OLE 使用的是复合文档（Compound Document）的概念——现在微软 Word 中的文档文件（.doc 文件）仍然使用这种文件格式。这种格式的文件可以加入文字、波形声音、图像剪辑等元素。在 OLE 的

第一个版本 OLE1 中，组件程序和客户程序之间进行通信并没有使用 COM 规范，而是使用一种动态数据交换（DDE，Dynamic Data Exchange）机制（本章在程序实例中将介绍如何使用 C++ Builder 6.0 通过 DDE 机制进行编程）。DDE 建立在 Windows 操作系统消息机制基础之上，所以 DDE 程序存在着应用程序执行效率低、程序系统健壮性差、程序代码编制阅读困难等缺点。DDE 程序存在的严重缺陷在一定程度上限制了 OLE 的发展，微软在 OLE 的后续版本 OLE2 中，重新构建了其底层代码。新代码没有继续使用 DDE 机制，而是引入了 COM 模型，至此产生了第一个用 COM 框架构建的软件系统——OLE2。OLE2 与 OLE1 相比，在代码的执行效率、程序的稳定性等方面都有了显著的提高。

由于其底层结构采用了 COM 模型，并使用 COM 接口作为程序与程序之间的通信标准，OLE2 标准下的软件模块的定制和扩充变得非常容易。这使得 OLE 技术的发展不再局限于对象链接和嵌入，也不再局限于复合文档，而是扩充到了系统中程序间进行通信的一个统称。COM 的模型也随着 OLE2 技术的发展变得越来越成熟。

从系统工程的角度来看，在早期的计算机软件中，每个应用系统都是一个单独的应用程序。应用系统越复杂，程序就越庞大，系统开发的难度就越大。某个特定的系统一旦开发完成，应用程序将不再改变，直到该系统的后续版本出现。如果程序很庞大，系统开发周期就很长，应用系统在这两个版本之间既要满足操作系统软件的更新，又要满足计算机硬件环境的变化，在代码的编写上将出现很多困难。很显然，这种单独的应用程序很难赶上日新月异变化的计算机软硬件环境。

从系统工程和软件模型的角度来考虑，解决这个矛盾的一种方法是将应用系统分解成多个较小的模块，每一个模块都完成一定的功能，各个模块之间保持相互独立。程序运行时，如果各个模块之间需要协同工作，则通过相互之间的接口完成消息的传递和工作的协调等任务。每一个这样的单独的模块就是一个组件。从系统工程的观点出发，在设计一个应用系统时，就要将其拆分为多个组件，每个组件都能够单独开发、编译，设计运行要求比较高的应用系统，甚至要求每个组件可以单独进行调试和模拟运行环境进行测试。在完成了所有组件的编写之后，将它们组合即可以得到完整的应用系统。按照组件模型开发完成的应用系统，可扩展性良好，例如，当系统运行的软硬件环境有任何变化时，并不需要像以前一样重新设计整个应用系统，只需要对受到软硬件变化影响的部分进行修改，然后重新编译链接到应用系统当中，这也是现在广泛采用的一种软件升级的方法。图 1-1 就是这种升级方法的示意图。

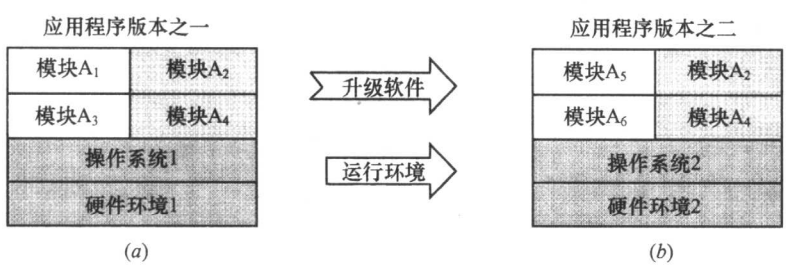

图 1-1　组件化应用程序升级示意图

如图 1-1 所示，如果按照从图(*a*)到图(*b*)的应用程序的运行环境发生了变化——可能是硬件环境或软件环境，或者两者兼而有之。应用程序的软件模块 A_1 和软件模块 A_3 受

到运行环境变化的影响。如图 1-1 所示，升级软件后，应用程序中受到影响的组件软件模块 A_1 和 A_3 分别由新的软件模块 A_5 和 A_6 所替代。由新软件模块和其他未受影响的组件组合得到新的应用程序版本之二，在新的运行环境之下可以正常工作。这样，在没有完全修改整个系统编码的情况之下，得到了一个新的应用程序系统。

注意：在真实的软件系统当中，每个模块之间还要通过接口进行通信。图 1-1 是利用组件思想编制的应用程序升级时的示意图，该图已将模块与模块之间通信的接口省略。使用新模块替代旧模块时，要特别注意使新模块保持与原来接口的兼容性，以便和应用系统中其他模块通信，保证系统顺利和稳定地运行。

在设计组件化的应用系统时需要引起重视的是：设计和实现软件的组件化模型是一件比较困难的事情，如果一个应用系统过于庞大和复杂，要切分成一系列单独的组件并不容易。通常情况下，除了要满足系统运行的业务要求外，还要尽可能地符合应用逻辑上的要求。这就要求程序设计者从系统论的观点出发，着眼于应用系统的全局，综合用户的需求和应用程序组件化设计的规则来进行程序设计。传统的结构化程序的设计思想和现在应用很广泛的面向对象程序设计思想对组件模型的设计都有很好的辅助作用，尤其是单个组件模块的设计，更是采用了许多结构化程序设计和面向对象的程序设计技术。本书着重讨论的是面向对象的组件模块设计技术，对于结构化的设计思想和技术，并没有过多的涉及，有兴趣的读者可以参考相关的书籍。

1.2.2 COM——面向对象组件模型

在 COM 技术当中，融合了时下广泛流行的面向对象的思想。在 COM 标准中，也有对象存在，COM 中的对象通常叫 COM 对象。对象是 COM 标准中一个要素，COM 对象的活动空间就是组件模块，COM 对象提供服务则必须要通过接口的方式。组件软件最关键的部分就是组件之间的接口，接口是组件之间进行通信的基础。从通信原理的角度来说，软件的模块之间如果要进行通信，必须有统一的标准来约束，即使是单个软件的组件之间要进行相互通信，也必须使用相同的通信标准。通信标准的选择，则可以根据软件开发时的具体情况而定。例如，如果仅在软件内部的组件模块中进行通信，那么可以采用自己定制的接口标准，但是如果要考虑和其他软件接口通信，就要先了解其他软件采用何种接口标准，或者是利用其他软件提供的公用接口标准。在 Windows 平台上，使用最为广泛，开发者一致推荐的标准是 COM 组件标准。图 1-2 说明了 COM 组件、COM 对象和 COM 接口三者之间的关系。COM 组件的接口为 COM 对象提供了服务，被称为 COM 接口。

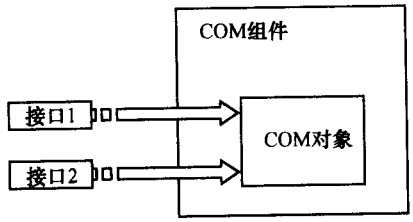

图 1-2 COM 组件、COM 对象和接口的关系

在 Windows 操作系统中，COM 组件的形式可以是动态链接库（DLL，Danymic Linking Library），可以是后缀名为.exe 的可执行文件。不论其具体形式是什么，一个 COM 组件可以包含多个 COM 对象，每一个 COM 对象又可以实现多个 COM 接口。当其他的组件或者是应用程序（这时应用程序被称为组件的客户程序）调用该 COM 组件时，首先必须创建一个 COM 对象或者是通过其他方法获得 COM 对象，然后才能通过该对象所提供的 COM 接口调用它的服务。所有的服务结束以后，如果后续工作中不再需要这个对象，必须释放该对象所占用的资源，同时还必须释放对象自身。

1.2.3 COM 的发展

在 OLE2 及其以前的工作中，技术的重点放在了解决运行环境当中应用程序之间的交互问题，包括了应用程序之间的数据交互和界面的交互，例如：当用户将 Excel 制成的一张表格插入到 Word 编辑的文档时，首先可以看到其中的数据被粘贴到了文档里；其次，Word 的工具栏中也多出了对 Excel 表格进行编辑的工具栏。前者是典型的数据交互，后者则是界面的交互。现在，由于有了 COM 这一利器，OLE2 技术已经轻松实现了程序之间交互的目标，COM 技术也呈现出了新的发展。

1. COM 与 Windows

Windows 操作系统早已完成了从 16 位向 32 位的变迁。在 16 位的 Windows 3.x 操作系统上，已经引入了对 OLE 的支持，并且也已经按照动态链接库（具体讲是使用 DLL）的组件模块的结构建立起来了。但是，由于技术尚未成熟，组件模块之间绝大多数并没有采用 COM 接口，这种状态之下，组件模型的优势是没有办法充分发挥出来的。Windows 系统向 32 位转变之后，许多的部件已经完全采用了 COM 组件的形式，例如，在 Windows 9x/Me 和 Windows NT 当中，不仅考虑了和以往版本的 SDK 的兼容性，在新增的组件中也都提供了 COM 接口。这样既使直接利用多种开发工具直接调用操作系统提供的功能成为可能，也有利于组件某些部分在需要时单独进行升级。这些 COM 形式提供的组件模块在 Windows 系统上丰富了操作系统的功能，也使得操作系统的功能扩展变得相对灵活，例如：

(1) DirectX 多媒体软件包 该软件包是以 COM 接口的形式为 Windows 平台提供强大的多媒体功能，在多媒体软件开发领域有着广泛的应用。

(2) RDO（Remote Data Object，远程数据对象）和 DAO（Data Access Object，数据访问对象）它们是以 COM 自动化对象形式为数据库应用提供的操作方法。在一些高级语言（如 Visual Basic）当中相当实用，现在仍然有许多初学者在利用 RDO 和 ADO 进行编程。更先进一些的 OLE DB 和 ADO（Active Data Object，活动数据对象）则充分发挥了 COM 接口的优点。

(3) Internet Client SDK 这是一组 COM 库文件，为应用系统增加网络特性提供了用户透明的底层操作的一致性。

其余的组件还包括 MAPI（Messaging API，消息应用编程接口），ADSI（Active Directory Service Interface，活动目录服务接口）等，这些组件都提供了标准的接口、高效的服务。当今的 Windows 系统中，COM 已经成为了系统的基本软件模型，COM 的高效性和灵活性已经贯穿于 Windows 操作系统当中，应用程序的开发和系统功能的扩展也

变得更加透明和快捷。从长远关系来看，Windows 操作系统的后续版本将继续保持和 COM、其更高级版本 COM+以及它们的后续版本的紧密联系。甚至可以这样断言：COM 已经成为 Windows 平台的组件标准模型。

2. COM 与数据库

在 Windows 操作系统下编写数据库应用程序最为常用的就是使用 ODBC（Open DataBase Connectivity，开放型数据库连接标准）编程接口。ODBC 已经成为编程的一种标准，用于在应用程序当中连接各种数据源。数据源不同，底层的 ODBC 驱动程序就不同。各个数据库软件厂商都提供了自己数据库的 ODBC 驱动，商业上通行的数据库都可以通过 ODBC 进行访问。

ODBC 是老一代的技术，微软在 ODBC 之后还推出了 RDO 和 DAO。从名称就可以看出，RDO 用于创建和维护远程 ODBC 数据库系统组件，它包括了一个自动化对象库，和 Basic 语言结合紧密，在 Visual Basic 中可以方便地使用，因此，用 Visual Basic 开发的数据库系统开始多采用 RDO。DAO 提供了两种完全不同的数据库访问方法：一种是通过微软的 Microsoft Jet 数据库引擎对微软 Microsoft Jet 数据库（后缀名为 mdb）和 ISAM（Indexed Sequential Access Method，索引顺序存取方法）数据进行高效快速的访问；另一种则是通过 ODBC Direct（在 DAO3.5 中首次引进）直接访问 ODBC 数据源。虽然通过 Jet 数据引擎也可以访问 ODBC 数据源，但是通过 ODBC Direct 访问数据源的速度更快、效率更高，这些优点在访问远程数据库时尤其明显。

微软的第 3 代技术是对数据进行一致访问的 OLE DB 和 ADO。OLE DB 是完全基于 COM 思想开发的，它的发展趋势是成为 ODBC 的替代品，此外，其应用范围不再仅局限于关系型数据库，而是适用于所有的线性数据库；ADO 是建立于 OLE DB 上层的自动化对象库，它的优点在于可以使用于各种脚本语言中，为脚本代码访问数据库提供了极大的方便。OLE DB 和 ADO 包含数据库访问的 3 个层次：数据提供者（Data Provider）、服务组件（Service Component）和客户（Consumer）。由于其接口是开放的 COM 接口，可以很方便地增加数据源支持，数据提供者的任务只是提供一些基本的服务，在应用层上的数据客户就可以获得各种服务组件提供的服务。

3. COM 与网络应用

COM 的应用已经扩展到了局域网和广域网的应用。现在局域网的应用不再是由一个厂家提供所有的产品，通常是一些专业的厂商提供其专业的产品，然后进行系统集成。从软件行业的发展来看，系统集成占有的比重越来越大，系统集成中最关键的问题就是软件之间的接口，只要所有的软件都遵从同样的接口标准，不同的软件产品之间就可以进行交互。COM 就是这样一个标准，并且正在实际开发中发挥着作用。在局域网系统中，企业信息资料库是数据库服务器，局域网中各种应用软件都通过数据库进行数据共享，但是并非所有的资料都属于数据库信息，例如年度计划、总结等。这些资料也要进行管理，而且管理和利用这些资料的软件也不尽相同，例如在总规划设计室里通过文件服务或者电子邮件获得某生产线上的生产计划。为了实现资料信息的共享，实现共享资料数据的组件必须符合开放、一致的接口标准。在广域网（Internet）的应用软件的发展中，最能体现 COM 的优势，Internet 软件对开放性的要求很高，开放性对软件的要求就是遵从标准。微软提出的 ActiveX 技术包含了所有基于 COM 的 Internet 相关的软件技术，从