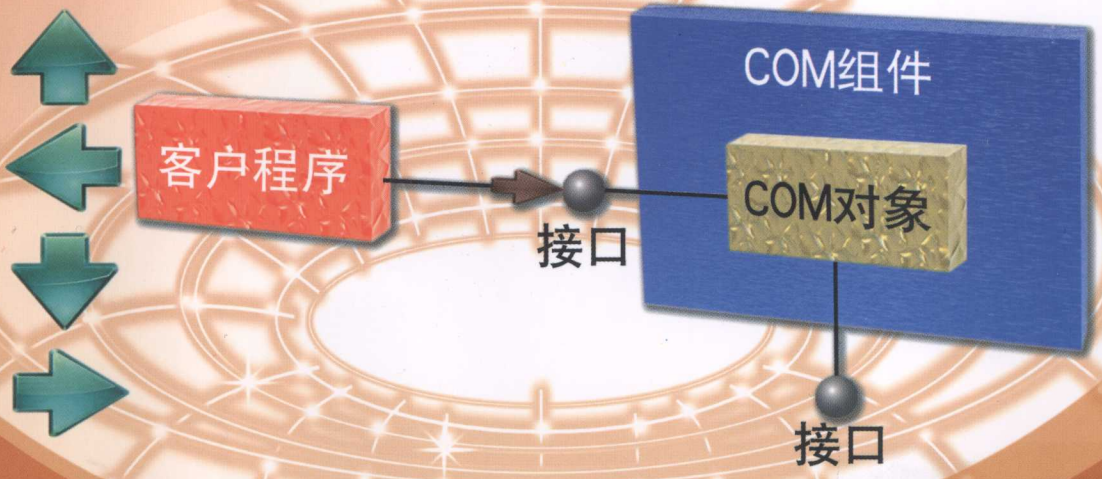
普通高等教育计算机类“十二五”规划教材

微软组件技术

主　编　赵　莉　耿军雪　杨国梁
副主编　荆　心　徐　飞　孙喁喁

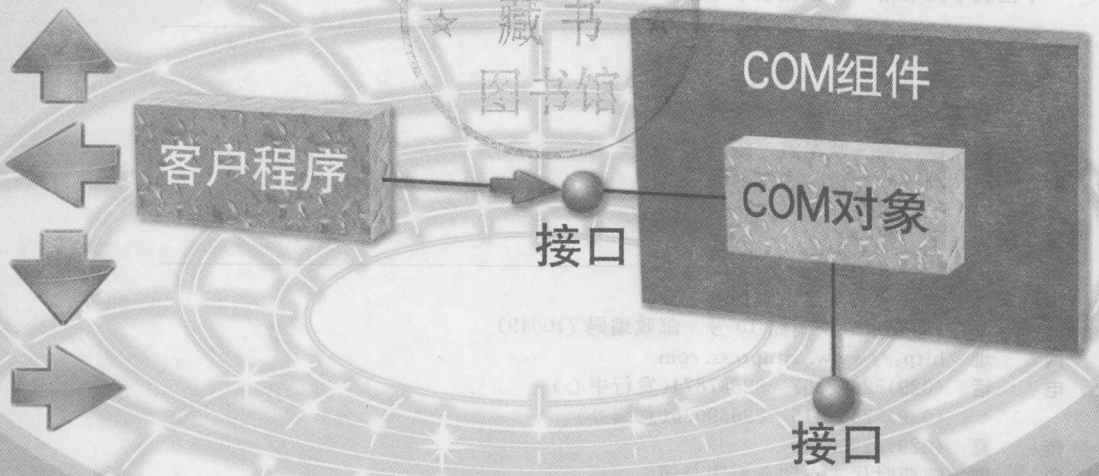


西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

普通高等教育计算机类“十二五”规划教材

微软组件技术

主　编　赵　莉　耿军雪　杨国梁
副主编　荆　心　徐　飞　孙喁喁



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

内容简介

本书全面系统地介绍了COM/DCOM/COM+等微软组件技术，并结合ATL、VC++等开发工具和OLE DB、ADO、ActiveX等技术阐述了COM的应用。内容包含了COM组件的接口、对象的概念及COM组件的实现；COM的高级特性：包容和聚合实现组件的复用、客户程序创建COM对象的进程透明性和COM的线程模型；自动化对象概念及实现；可连接对象通信机制；ActiveX控件实现；OLE DB和ADO数据库访问技术；DCOM技术；COM+应用；.NET组件技术。各章均提供了丰富的实例，便于读者巩固知识，掌握组件设计的基本方法和技巧。本书力求概念叙述准确、严谨，描述简练，语言通俗易懂，使读者易于理解和掌握。

本书适合作为高等院校计算机及相关专业组件技术课程的教材和工程技术人员学习组件的参考书，也适合于编程开发人员培训、广大计算机技术爱好者自学使用。

图书在版编目(CIP)数据

微软组件技术/赵莉等主编. —西安：西安交通大学出版社，2013.9
ISBN 978-7-5605-5709-0

Ⅰ.①微… Ⅱ.①赵… Ⅲ.①软件组件 Ⅳ.①TP311.56

中国版本图书馆CIP数据核字(2013)第218806号

书　　名　微软组件技术
主　　编　赵　莉　耿军雪　杨国梁
副 主 编　荆　心　徐　飞　孙喁喁
策划编辑　毛　帆
责任编辑　毛　帆

出版发行　西安交通大学出版社
（西安市兴庆南路10号　邮政编码710049）
网　　址　http://www.xjtupress.com
电　　话　(029)82668357　82667874(发行中心)
(029)82668315　82669096(总编办)
传　　真　(029)82668280
印　　刷　陕西时代支点印务有限公司

开　　本　787mm×1092mm　1/16　**印张**　19.125　**字数**　463千字
版次印次　2013年9月第1版　2013年9月第1次印刷
书　　号　ISBN 978-7-5605-5709-0/TP·592
定　　价　35.00元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线：(029)82665248　(029)82665249
投稿热线：(029)82668254
QQ：8377981
读者信箱：lg_book@163.com

前　言

基于组件的软件开发技术是一种强调通过可复用组件设计与构造软件系统的软件复用技术途径。组件对象模型(COM)至今仍然保持着旺盛的生命力,其原理甚至被包括.NET 体系在内的系统所采用。基于 COM 的组件、工具和服务仍然在软件领域扮演着重要角色,本书将全面深入地介绍 COM/DCOM/ COM+原理,并结合 ATL、VC++等开发工具和 OLE DB、ADO、ActiveX 等技术阐述 COM 的应用。

全书共分为 12 章。第 1 章绪论介绍了组件、设计模式的基础知识。第 2 章详细阐述了 COM 规范的实现细节,包括 COM 接口和对象,COM 的实现过程,并用 C++完成 COM 组件的实现和调用。第 3 章从重用性、跨进程性和多线程阐述 COM 的部分特性并给出了 MFC 的支撑。第 4 章讲述了自动化对象的概念、接口的实现和对象的使用。第 5 章给出了可连接对象的概念、连接过程并给出 MFC 实现。第 6 章用 ATL 开发 COM 应用的各个实例,包括进程内组件、多接口组件以及自动化组件的实现。第 7 章讲述 ActiveX 控制,详细论述了 ActiveX 控制的基本理论和 MFC 实现 ActiveX 控件实例。第 8 章数据库编程实践,给出了 OLE DB 数据库访问及 ADO 数据库访问的编程实例。第 9 章讲述了 DCOM 通信模型以及远程创建 DCOM 对象的过程。第 10 章从 COM+的基本结构、系统服务和应用开发模型阐述 COM+的应用。第 11 章简单概述了 NET 组件以及与 COM 组件的互操作。

本书力求概念叙述准确、严谨,描述简练,语言通俗易懂,使读者易于理解和掌握,丰富的实例便于读者巩固知识,掌握组件设计的基本方法和技巧。本书适合作为高等院校计算机及相关专业组件技术课程的教材和工程技术人员学习组件的参考书,也适合于编程开发人员培训、广大计算机技术爱好者自学使用。

编　者

2013 年 8 月

目　录

第1章　绪论

软件重用是软件工程的重要领域，被认为是解决软件危机、提高软件生产率和软件质量、增强软件的开放性和对外部扰动的适应性的主要途径。软件重用，是指在两次或多次不同的软件开发过程中重复使用相同或相似软件元素的过程。软件元素包括程序代码、测试用例、设计文档、设计过程、需求分析文档甚至领域知识。通常，可重用的元素也称作软构件，可重用的软构件越大，重用的粒度越大。

自从软件重用思想产生以来，计算机科学家和软件工程师就致力与软件重用的技术的研究和实践。在30多年的时间内，出现多种软件重用技术，如：库函数、面向对象、模板、设计模式、构件、构架、框架。

1. 库函数

库函数是很早的软件重用技术。很多编程语言为了增强自身的功能，都提供了大量的库函数。对于库函数的使用者，他只要知道函数的名称、返回值的类型、函数参数和函数功能就可以对其进行调用。

2. 面向对象

面向对象技术一直是学术界和工业界研究和应用的一个热点。面向对象技术通过方法、消息、类、继承、封装和实例等机制构造软件系统，并为软件重用提供强有力的支持。面向对象方法已成为当今最有效、最先进的软件开发方法。与函数库对应，很多面向对象语言为应用程序开发者提供了易于使用的类库，如VC++中的微软基础类库MFC。

3. 模板

模板相当于工业生产中所用的“模具”。有各种各样的模板（如文档模板、网页模板等），利用这些模板可以比较快速地建立对应的软件产品。模板把不变的部分封装在内部，对可能变化的部分提供了通用接口，由使用者来对这些接口进行设定或实现。

4. 设计模式

设计模式作为重用设计信息的一种技术，在面向对象设计中越来越流行。设计模式描述了在我们周围不断重复发生的问题，该问题的解决方案的核心和解决方案实施的上下文。设计模式命名一种技术并且描述它的成本和收益，共享一系列模式的开发者拥有共同的语言来描述他们的设计。

5. 构件

普通意义上的构件应从以下几个方面来理解：

① 构件应是抽象的系统特征单元，具有封装性和信息隐蔽，其功能由它的接口定义。

② 构件可以是原子的，也可以是复合的。因此它可以是函数、过程或对象类，也可以是更大规模的单元。一个子系统是包含其他构件的构件。

③ 构件是可配置和共享的，这是基于构件开发的基石，且构件之间能相互提供服务。

6. 构架

普通意义上的构架应从以下几个方面来理解：

① 构架是与设计的同义理解，是系统原型或早期的实现。

② 构架是高层次的系统整体组织。

③ 构架是关于特定技术如何合作组成一个特定系统的解释。

7. 框架

如果把软件的构建过程看成是传统的建筑过程，框架的作用相当于为我们的房屋搭建的“架子”。框架从重用意义上说，是一个介于构件和构架之间的一个概念。

构件、框架和构架三者的主要区别在于：对重用的支持程度的不同。

① 构件是基础，也是基于构件开发的最小单元。构件重用包括可重用构件的制作和利用可重用构件构造新构件或系统。

② 一个框架和构架包含多个构件。这些构件使用统一的框架（构架）接口，使得构造一个应用系统更为容易。

③ 框架重用包括代码重用和分析设计重用，一个应用系统可能需要若干个框架的支撑，从这个意义上来说，框架是一个“构件”的同时，又是一类特定领域的构架。

④ 构架重用不仅包括代码重用和分析设计重用，更重要的是抽象层次更高的系统级重用。

⑤ 框架和构架的重用层次更高，比构件更为抽象灵活。

面向过程的编程重用函数、面向对象的编程重用类、范型编程重用的是算法的源代码，而组件编程则重用特定功能完整的程序模块。

每个组件会提供一些标准且简单的应用接口，允许使用者设置和调整参数和属性。用户可以将不同来源的多个组件有机地结合在一起，快速构成一个符合实际需要（而且价格相对低廉）的复杂（大型）应用程序。

组件区别于一般软件的主要特点，是其重用性（公用/通用）、可定制性（设置参数和属性）、自包容性（模块相对独立，功能相对完整）和互操作性（多个组件可协同工作）。它可以简单方便地利用可视化工具来实现组件的集成，也是组件技术一个重要优点。

目前常用的组件框架模型，一类是与某一计算机操作系统密切相关，而另一类是跨计算机操作系统平台的。前者的典型代表是 COM 组件对象模型，以及在此基础上发展起来的 ActiveX、DCOM、COM＋、MTS 和. NET 等技术。COM 组件具有二进制一级的兼容性，基本上与计算机编程语言无关，其缺点是目前只能运行在 Windows 操作系统平台上，而不能在 Linux 和 Unix 系统中运行。COM 并不只是面向对象的组件对象模型，它既可用面向过程，也可用面向对象的语言编码，但多采用的编码语言是 VC＋＋、VB 和 Delphi，性能要求高的场合也可用 C 语言来编码。COM 已经广泛使用在 Windows 操作系统中，浏览器、邮件收发系统、Web 服务器、字处理软件中也都广泛使用 COM 组件。跨计算机操作系统平台的组件模型其典型代表是 CORBA，CORBA 主要使用在 Unix 类型的操作系统中，但它也可在 Windows 平台上运行。

从计算机语言来讲，组件模型有以 Java 语言为代表的框架和以 C 语言为基础的框架。前

者在理论上可以跨平台运行，底层平台支持是 JVM 技术；而后者则与虚拟机无关，直接在操作系统中运行，因此，速度快，运行效率高。从应用系统的角度来讲，目前市场上，主要是 J2EE 和. NET 的竞争，两者理论上没有本质的区别，都是采用虚拟机技术。但 J2EE 可以跨平台运行，而. NET 则基本不行。在企业级的应用系统中，以 Java 技术为基础的 J2EE 似乎更占优势。Java 和. NET 技术各有特长，因此，在信息系统建设中，应该允许两种技术同时存在，取长补短，协同发展，最大限度地提高系统开发的性价比和稳定性。

1.1 软件组件

软件组件技术是解决软件复用，缩短软件编写时间，降低维护成本和实现程序动态升级的最新和强有力的方案，在整个软件工业界中得到了迅速应用。

1.1.1 软件组件特点

软件组件技术将应用程序分割成小的可复用的组件，在运行时将这些组件组装起来形成所需的应用程序。每一个组件都可以在不影响其他组件的情况下被升级，这使得应用程序可以随时向前发展进化。应用组件技术还可以在应用程序定制、组件库及分布式组件等方面获益。

组件架构是可以被定制的，用户可以根据需要将某个组件替换掉。在图 1-1 中，假定某些组件是基于编辑器 Notepad 和 Word 的，用户 1 将应用程序配置成为使用 Notepad，而用户 2 更喜欢使用 Word。按照此种方式，可以加入新的组件或改变已有组件而方便地定制应用程序。

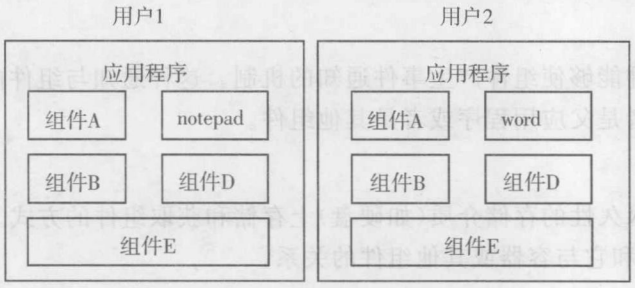


图 1-1 应用程序定制

组件架构最引人注目的优点之一是快速应用程序开发。这一优点可以使开发人员从某个组件库中取出所需要的组件并将其快速地组装到一起以构成应用程序，如图 1-2 所示。在将已有应用程序转化成分布式应用程序时，若程序是由组件组装成的，转化过程将会简单得多。首先，应用程序已经被划分成可以位于远地的各个功能部分；其次，由于任一组件均是可以被替换的，因此可以将某个组件替换成专门负责同远程机上的组件通信的组件。

在图 1-2 中，组件 C 和组件 D 被放到了网络远程机器上。在本地，它们被替换成了两个新的组件：组件 C1 和 D1。这两个新的组件的作用是将其他组件发来的请求通过网络转发给组件 C 和组件 D。本地机器上的应用程序并不需要知道实际所用到的组件到底在何处。类似地，远地组件也不需要知道它们是否位于远地。这样通过加入合适的远地组件，应用程序完全

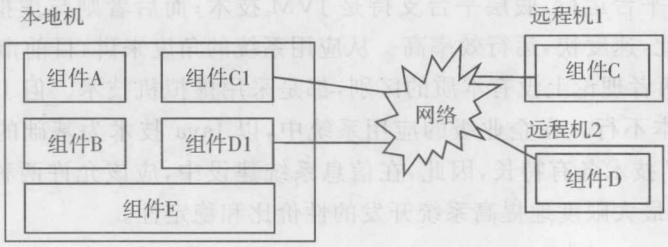


图 1-2 使用了远程组件的应用程序

不需要知道实际的组件到底在哪里。

1.1.2 软件组件模型

组件是指能够容易地组装起来，以更高的开发效率创建应用程序的可复用软件部分。组件技术的核心是组件模型，它定义了组件的结构、外部如何操作组件及组件间的相互作用。组件和容器是组件模型的两种基本元素。组件部分提供了创建实际组件的模板，是组件创建与利用的基础。容器部分定义了将组件集中起来成为有用结构的方法，提供了安排组件及与其他组件相互作用的语言环境。组件模型还负责提供各种形式的服务。功能完善的组件模型可支持下列六种主要服务。

1. 自我描述

自我描述是指向外界展示组件功能的机制。通过自我描述机制，应用程序能够对组件提出询问，找到组件的功能并且与组件相互作用。自我描述是组件模型的关键特征之一，负责决定组件对应用程序以及其他组件的外观。

2. 事件处理

事件处理是一种能够使组件产生事件通知的机制。这种通知与组件内部状态的变化相对应。通知的对象或者是父应用程序或者是其他组件。

3. 持久性

持久性是指在永久性的存储介质(如硬盘)上存储和获取组件的方式。存储和获取的信息是组件的内部状态，和它与容器或其他组件的关系。

4. 布局

布局是指可视化组件的物理布局。包括组件自己空间内的组件布局和在同一个容器内与其他组件共享空间的布局。组件的空间需求主要是给组件提供一个矩形区域，以使组件成为可视的。

5. 对应用程序建立器的支持

组件对应用程序建立器的支持，使用户能以图形方式在组件外建立复杂的应用程序。通常应用程序建立器的支持由对话框组成，使用户能以图形界面的方式来编辑组件的属性。

6. 对分布式计算的支持

随着网络技术和应用的发展，组件模型对分布式计算的支持变得越来越重要。软件组件模型可以简化分布式应用程序的开发过程。

1.1.3 软件组件实现条件

使用组件的种种优点直接来源于可以动态地将它们插入或卸出应用程序。为了实现这种功能，所有的组件必须满足以下两个互相依赖的条件。

1. 动态链接

通过动态链接，用户可以在运行时将组件替换掉，而开发人员不需要将整个程序重新编译或链接一遍后重新发行新的版本。

2. 信息封装

应用程序是由各个组件连接起来的。当用新的组件把某个组件替换掉时，需要将此组件同系统断开，然后将新的组件连上去。显然新的组件必须按同样的方式连接到系统中，否则将需要重新编写、重新编译或重新链接这些组件。对于一个应用程序或组件，如果它使用了其他组件，我们称之为一个客户，客户通过接口同其他组件进行连接。如果某个组件发生了变化，而其接口没有任何改变，那么它的客户将不需要进行任何修改。类似地，若客户发生了变化，而没有改变其接口，那么它所连接的组件也不需要任何改变。相反，如果客户或组件的变化导致了对接口的修改，那么接口的另一方也应发生相应的变化。因此为了充分发挥动态链接的功能，组件及客户都应尽可能不要改变它们的接口，即它们必须封装起来。也就是说，组件及客户的内部实现细节不能反映到接口。接口同内部实现细节的隔离程度越高，组件或客户发生变化时对接口的影响将越小。这种将客户同组件实现隔离的要求对组件加上了如下的限制：

①组件必须将实现其所用的编程语言封装起来。编程语言的暴露将会在组件和客户间引入新的依赖。

②组件必须以编译、链接好的可执行代码的形式发布。

③组件必须可以在不妨碍已有用户的情况下被升级。

④组件在网络上的位置必须可以透明地被重新分配。组件及使用它的程序应能够在同一进程中、不同进程中或不同机器上运行。客户对远程组件的处理方式应与对本地组件的处理方式一致。

1.1.4 微软组件技术

COM(Component Object Model，组件对象模型)是微软公司于1993年提出的一种组件技术，是软件对象组件之间相互通信的一种方式和规范。它是一种与平台无关、语言中立、位置透明、支持网络的中间件技术。

COM是OLE(Object Linking and Embedding，对象链接和嵌入)的发展(而OLE又是DLL(Dynamic Link Libraries，动态链接库)的发展)，DCOM(Distributed COM，分布式COM，1996年)和COM+(DCOM+管理，1999年)则是COM的发展。ActiveX控件是COM的具体应用(如VBX和DirectX都是基于ActiveX的)，ATL(Active Template Library活动模板库)是开发COM的主要工具，也可以用MFC来直接开发COM，但是非常复杂。

作为组件技术的进一步发展，微软公司于2002年推出了.NET框架，其中的核心技术就是用来代替COM组件功能的CLR(Common Language Runtime，公共语言运行库)，可采用

各种编程语言，利用托管代码来访问（例如 C#、VB、MC++），使用的是.NET 的框架类库 FCL(Framework Class Library)。微软公司的各种组件技术之间的关系与发展可以参见下图：

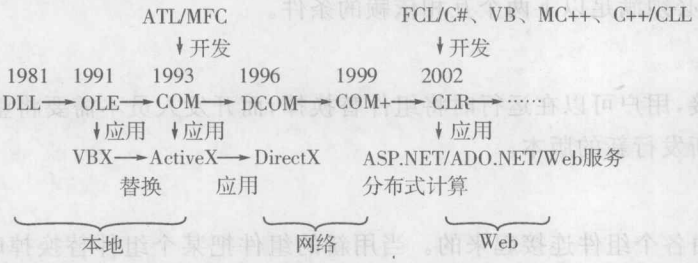


图 1-3 微软组件技术之间的关系和发展示意图

1.2 组件对象模型(COM)

组件对象模型 COM 是微软公司(Microsoft)于 1993 年创建的；是微软公司、数据设备公司(DEC)等公司所支持的一种软件组件结构标准。开发 COM 的目的是为了使应用程序更易于定制、更为灵活，最初目标是为对象链接与嵌入(OLE)提供支持。COM 提供了创建兼容对象的技术规范，以及运转它所需的 Windows 操作系统进程间通信(Inter-Process Communication，IPC)规范。

COM 规范是一套为组件架构设置标准的文档，提供了一种编写与语言无关的能够按面向对象 API 形式提供服务的组件的方法。COM 具有一个被称作 COM 库的 API，它提供了对所有客户及组件都非常有用的组件管理服务。COM 库可以保证对所有组件的大多数重要的操作都可以按相同的方式完成。COM 库中的大多数代码均可以支持分布式或网络化的组件。Windows 系统上 DCOM 的实现中提供了一些同网络上其他组件通信所需的代码。

1.2.1 COM 的特点

COM 负责设计、构建和使用软件组件，它是当前所有 Microsoft 32 位操作系统都提供的一个系统级别的技术。通过使用 COM 编程模型开发软件，程序员将获得大量内置的功能。特别值得一提的是，COM 赋予了软件模块下列一些属性：同语言的无关性（或语言环境的独立性，language independence）、版本升级的鲁棒性（即稳健性，robust veisioning）、位置的透明性（location transparency）和面向对象（object orientation）的特性。

1. 同语言的无关性：二进制的设计标准

尽管本书的重点集中在使用 C++建立基于 COM 的组件，但实际上 COM 组件可以由任何语言编写。事实上，其他一些语言，如 Visual Basic 和 Java，在开发和使用基于 COM 的组件上比 C++要更加方便。不过对于客户程序（或用户）而言，基于 COM 的组件是使用哪一种语言实现的，以及是怎样实现的并不重要，它们（他们）只关心组件本身的功能。

换句话说，基于 COM 的软件模块是和语言无关的。你可以使用 C++编写组件，并在 Visual Basic 中使用该组件；你也可以使用 Visual Basic 开发一个组件，而在 Java、C++或

Visual FoxPro 中使用它。这种混合的使用没有任何不妥之处。

COM 的一个最重要的特征在于它为你提供了一种编写面向对象的程序的新技术，使得你在编程中可以选用任何一种语言，而用户可以在另外一种语言环境里调用该组件的功能。COM 支持一种所谓的二进制设计标准：你可以把一个组件纳入到一个 DLL 或 EXE(二进制文件)文件里，组件的功能可以被 Visual Basic、Java、C++甚至 COBOL 语言调用。当然，实现这一功能的语言必须支持 COM，而在 Windows 环境里几乎所有的语言都支持 COM。

2. 版本升级的鲁棒性

COM 的另外一个重要特征是它可以支持组件版本的升级。在传递软件模块时，尤其是在多提供商应用程序的环境中共享资源时，一个困难之处是在已发布的软件模块里对功能进行升级。COM 通过使用一种具有鲁棒性的版本升级技术来解决这一问题，该技术是基于 COM 最基本的实体——组件接口(component interface)实现的。

COM 通过它对统一组件的多接口支持实现版本升级的鲁棒性。换句话说，组件的功能可以分割为细小的、独立的区域，这些区域的每一个都具有一个特定的 COM 接口。由于一个组件可以适应同一接口的微小变化，所以 COM 也就因此可以提供版本支持功能。也就是说，可以允许旧的应用程序在不进行改动的情况下运行，同时新的应用程序可以利用组件的新添特性。

3. 位置的透明性

COM 的另外一个重要的特性是位置的透明性。该特性意味着组件的用户——客户机并不需要明确了解组件所处的位置。一个客户机应用程序使用相同的 COM 服务来创建组件的实例并使用它，而无需考虑组件所在的位置。

组件可能直接位于客户机处理工作区内(一个 DLL 文件里)，也可能位于同一台计算机上的另外一个处理程序里(一个可执行文件)，还可能位于远端计算机上(一个分布式的对象)。COM 和 DCOM 提供这种位置上的透明性。

不管组件到底在什么位置上，客户机同基于 COM 的组件都将以相同的方式进行交互，而客户机接口不会改变。位置的透明性允许程序开发员建立具有可升级性、分布式、多层次的应用程序，而无需更改客户机应用程序所使用的编程模型。

4. 面向对象的特性

COM 允许软件模块以面向对象的方式传递其功能。大量旧版本软件的交互操作技术(如 DLL 专家系统和 DDE)都不提供典型的面向对象的特征，而对于 C++程序员而言，面向对象的特征是最为常用的。COM 提供三种基本的面向对象的特征，它们分别是封装(encapsulation)、继承(inheritance)和多态(polymorphism)，并且 COM 是以一种与语言无关的方式对这三种特征提供了支持。

1.2.2 COM 组件分类

根据组件与客户所处的物理空间的不同，COM 组件(也称服务器)分为以下三类：进程内服务器(in-process server)、本地服务器(local server)和远程服务器(remote server)。后两种合称进程外服务器(out-of-process server)。

1. 进程内服务器

当服务器和客户在同一进程空间中运行时，服务器称为进程内服务器，如图 1－4 所示。进程内的 COM 服务器，是以 DLL 形式封装的 COM 组件。

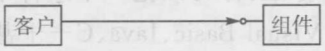


图 1－4 进程内服务器

因为组件与客户在同一个进程内，因此客户可以直接使用组件返回的接口指针，而不需要进行跨进程的调用，因而这种服务器的速度最快。

2. 本地服务器

当服务器与客户位于同一计算机上，但它们分别运行在独立的进程空间时，该服务器称为本地服务器。如图 1－5 所示。本地服务器常以 EXE 的形式封装。

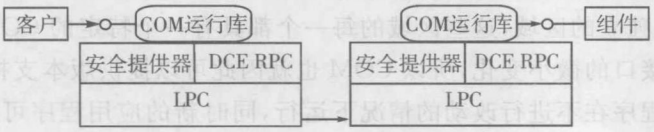


图 1－5 本地服务器

因为每个进程都有其自己的进程空间，不同进程空间中的相同的逻辑地址所对应的物理地址将是不同的。因此在不同进程间传递特定于进程的指针是没有意义的。在本地服务器的情况下，客户是不能直接通过服务器提供的接口指针访问服务器组件进程的地址空间。因此对于这种跨越进程边界的接口，需要考虑以下一些条件：

①一个进程需要能够调用另外一个进程中的函数；

②一个进程需要能够将数据传递给另外一个进程；

③客户无需关心它所访问的服务器是进程内还是进程外服务器。

COM 通过本地过程调用(Local Process Call，LPC)实现了不同进程间的通信。LPC 是基于远程过程调用(Remote Process Call，RPC)的用于单机上进程间通信的专利技术。RPC 标准是在开放软件基金会(Open Software Foundation，OSF)分布式计算环境(Distributed Compute Environment，DCE)RPC 规范中定义的，它使得不同机器上的进程可以使用各种网络传输技术进行通信。

调用进程外的函数只是第一步，还需要一种方法将函数调用的参数从一个进程的地址空间传到另外一个进程的地址空间中，这种方法称作“调整”。若两个进程都在同一台机器上，则调整过程将是相当直接的：只需要将参数数据从一个进程的地址空间中传到另一个进程的地址空间就可以了。若参与参数传递的两个进程在不同的地址空间中(如下面要讲的远程服务器的情况)，那么考虑到不同机器在数据表示方面的不同，如整数字节顺序可能会不一样，必须将参数数据转换成标准的格式。LPC 技术可以将数据从一个进程复制到另外一个进程中。为对组件进行调整，可以实现一个名为 IMarshal 的接口。在 COM 创建组件的过程中，它将查询组件的 IMarshal 接口，然后它将调用 IMarshal 的成员函数以在调用函数的前后调整和反调整有关的参数。COM 库中实现了一个可供大多数接口使用的 IMarshal 的标准版本。在

需要对性能优化时，可以对 IMarshal 进行定制。

组件在网络上的位置必须可以被透明地重新分配，因此客户应该可以按照相同的方式与进程内、本地及远程组件进行通信。为达到这一目标，COM 使用了一种非常简单的方法。

对于 Windows 系统，当应用程序调用 Win32 函数时，系统实现上将调用一个 DLL 中的函数，而此函数将通过 LPC 调用 Windows 中的实际代码。这种结构可以将用户进程同 Windows 代码隔离开，COM 使用的结构与此类似。客户将同一个模仿组件的 DLL 进行通信，这个 DLL 可以为客户完成参数的调整及 LPC 调用。在 COM 中，此 DLL（也是一个组件）被称作是一个代理（Proxy）。在 COM 中，一个代理就是同另外一个组件行为相同的组件。代理必须是 DLL 形式的，因为它们需要访问客户进程的地址空间以便对传给接口函数的数据进行调整。对数据的调整只完成了任务的一半，组件还需要一个被称作是存根（stub）的 DLL，以对从客户传来的数据进行反调整。存根也将对传回的数据进行调整，如图 1-6 所示。

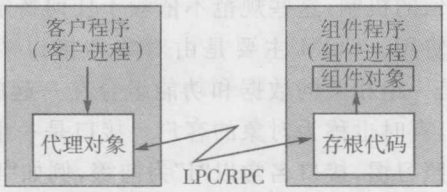


图 1-6　代理和存根

使用 LPC 和代理、残根需要编写大量的代码。借助一种名为 IDL（Interface Define Language）语言，可以编写接口的一个描述，然后用 MIDL 编译器即可以生成代理和残根 DLL。关于 IDL 和 MIDL 后面将详述。

3. 远程服务器

当服务器与客户位于不同的计算机上时，该进程外服务器称为远程服务器。如图 1-7 所示。远程服务器可以以 EXE 或 DLL 的形式封装。当以 DLL 形式封装时，在远程服务器的计算机上需要一个代理进程。

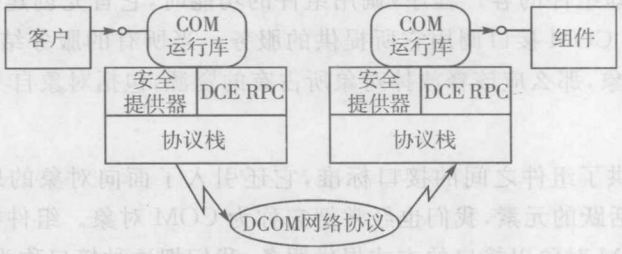


图 1-7　需要跨网路通信的远程服务器

与进程内服务器和本地服务器相比，进程外服务器的速度明显要慢，但在功能和可扩展性方面的收益将是具有革命性的，后面在介绍 DCOM 时将详细说明。

1.2.3　COM 的结构

COM 是由 Microsoft 提出的组件标准，它不仅定义了组件程序之间进行交互的标准，并

且也提供了组件程序运行所需的环境。在 COM 标准中，一个组件程序也被称为一个模块，它可以是一个动态链接库，被称为进程内组件(in-process component)；也可以是一个可执行程序(即 EXE 程序)，被称作进程外组件(out-of-process component)。一个组件程序可以包含一个或多个组件对象，因为 COM 是以对象为基本单元的模型，所以在程序与程序之间进行通信时，通信的双方应该是组件对象，也叫做 COM 对象，而组件程序(或称作 COM 程序)是提供 COM 对象的代码载体。

COM 标准为组件软件和应用程序之间的通信提供了统一的标准，包括规范和实现两部分，规范部分规定了组件间的通信机制。由于 COM 技术的语言无关性，在实现时不需要特定的语言和操作系统，只要按照 COM 规范开发即可。然而由于特定的原因，目前 COM 技术仍然是以 Windows 操作系统为主，COM 为组件和应用程序之间进行通信提供了统一的标准，它为组件程序提供了一个面向对象的活动环境。COM 标准包括规范和实现两大部分，规范部分定义了组件和组件之间通信的机制，这些规范不依赖于任何特定的语言和操作系统，只要按照该规范，任何语言都可以使用。COM 主要是由对象和接口两部分组成。对象是某个类(class)的一个实例；而类则是一组相关的数据和功能组合在一起的一个定义。使用对象的应用(或另一个对象)称为客户，有时也称为对象的客户。接口是一组逻辑上相关的函数集合，其函数称为接口成员函数。按照习惯，接口名常以“I”为前缀，例如“IUnknown”。对象通过接口和成员函数为客户提供各种形式的服务。图 1-8 可说明 COM 组件、COM 对象和 COM 接口三者之间的关系。

图 1-8 COM 组件、COM 对象和 COM 接口三者之间的关系

一个组件程序可以包含多个 COM 对象，而且每个 COM 对象可以实现多个接口。当另外的组件或普通程序(即组件的客户程序)调用组件的功能时，它首先创建一个 COM 对象或者通过该对象所实现的 COM 接口调用它所提供的服务。当所有的服务结束后，如果客户程序不再需要该 COM 对象，那么应该释放掉对象所占有的资源，包括对象自身。

1. COM 对象

COM 不仅仅提供了组件之间的接口标准，它还引入了面向对象的思想。在 COM 标准中，对象是一个非常活跃的元素，我们也经常把它称为 COM 对象。组件模块为 COM 对象提供了活动的空间，COM 对象以接口的方式提供服务，我们把这种接口称为 COM 接口。

在 COM 规范中，并没有对 COM 对象进行严格的定义，但 COM 提供的是面向对象的组件模型，COM 组件提供给客户的是以对象形式封装起来的实体。客户程序与 COM 组件程序进行交互的实体是 COM 对象，它并不关心组件模块的名称和位置(即位置透明性)，但它必须知道自己在与哪个 COM 对象进行交互，如图 1-9 所示。

类似于 C++语言中类(class)的概念，COM 对象也包括属性(也包括状态)和方法(也称为操作)，对象的状态反映了对象的存在，也是区别于其他对象的要素；而对象所提供的方法就是对象提供给外界的接口，客户必须通过接口才能获得对象的服务。对于 COM 对象来说，接