软件开发文集

第一辑

《软件开发文集》编委会 编

- 部件对象模型
- OLE 及其开发工具
- 管理一个 FoxPro 开发小组
- Visual Basic 4.0 全面发挥 Windows 95 的性能

科学出版社

软件开发文集

第一辑

《软件开发文集》编委会　编

科学出版社

1996

(京)新登字 092 号

内 容 简 介

本书是为软件开发人员和计算机用户编写的《软件开发文集》系列丛书的第一辑，围绕专题内容，向读者提供了软件开发经验和技巧、应用设计策略、开发技术规范、开发管理、开发平台等方面的技术资料。

本书的重点文章是：部件对象模型、OLE 及其开发工具、管理一个 FoxPro 开发小组，以及 Visual Basic 4.0 全面发挥 Windows 95 的性能、用 Visual C++和 MFC 开发数据库应用程序等专题。

本书适用于软件开发人员和广大计算机用户。

图书在版编目（CIP）数据

软件开发文集　第一辑/《软件开发文集》编委会编.—北京：科学出版社，1996.5

ISBN 7-03-005406-7

Ⅰ.软…　Ⅱ.软…　Ⅲ.软件开发-文集　Ⅳ.TP311.52-53

中国版本图书馆 CIP 数据核字（96）第 07626 号

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

新世纪印刷厂印刷

新华书店北京发行所发行　各地新华书店经售

*

1996 年 5 月第　一　版　　开本:787 × 1092　1/16

1996 年 5 月第一次印刷　　印张：6

印数：1 — 7 000　　字数：120 000

定价：9.00 元

编者的话

近年来，我国的计算机产业发展迅速，PC 机销量已居世界第六位。同时，我国有一批人数众多、技术水平相当不错的软件开发人员，他们分布在全国各地、各个行业、各个领域。然而，由于计算机产业，特别是软件产业充满了变化，极富动态性，加之，我国幅员辽阔，通信技术又落后于较发达的国家，从而，软件开发人员面临着这样的问题：即了解最新技术动态不及时、不方便，获取最新技术信息比较困难，所得到的信息往往不够全面、完整和深入，相互之间缺乏交流，因此，重复开发、重复别人走过的弯路的情况屡见不鲜。另一方面，全世界积累的资料浩如烟海，价格亦非常昂贵，鉴于国内经济承受能力有限，许多技术信息不能及时获得。有鉴于此，我们组织有关人员翻译、编写了以技术专集为特色的《软件开发文集》丛书，该套书将陆续出版、发行，及时向用户提供最新技术资料，力图为解决上述问题做一点尝试。

《软件开发文集》丛书的宗旨是为软件开发人员开辟一个相互交流经验和体会的园地，提供一条了解新技术、新工具、新产品的渠道，设立一个探讨软件开发理论和开发技术的论坛，帮助软件开发人员更快、更直接地获得有关软件开发的信息，提高软件开发人员的技术水平和工作效率，充分发挥软件的作用，提高软件的使用价值，促进我国软件产业与世界同步发展。

《软件开发文集》是面向软件开发人员的中、高档次的技术专集，所收集的文章向软件开发人员提供了最新科技动态，以及开发技术规范、开发经验和技巧、软件开发管理、应用设计策略、开发平台方面的内容等，同时，根据用户在软件使用和开发过程中遇到的难点、疑点给予一定的解答和帮助。《软件开发文集》资料主要来源于微软公司的“Microsoft System Journal”，“Developer News”，“Microsoft Development Library”，“TechNet”等，并收入国内软件开发人员撰写的有关文章。欢迎国内广大软件开发人员为《软件开发文集》撰稿，介绍自己的经验、心得、体会。特别要注重文章内容适合我国国情。

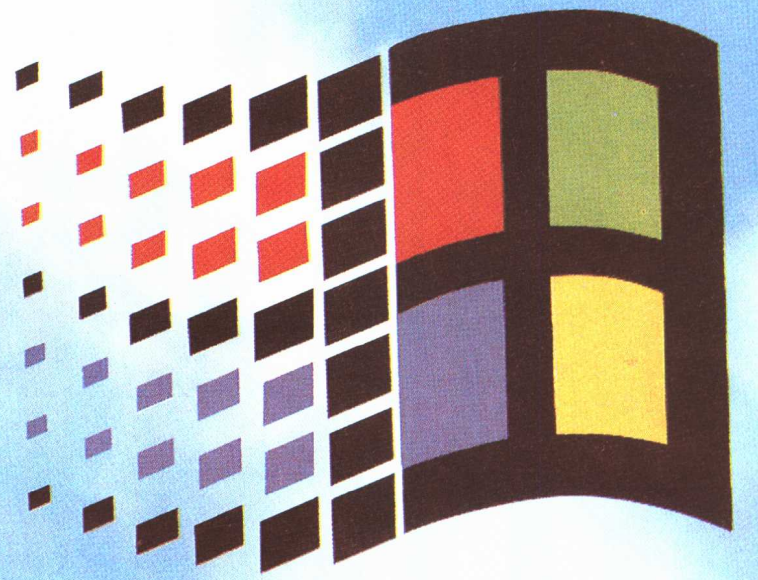
目前，在国内面向计算机最终用户的图书、杂志丰富多彩，但面向软件开发人员的图书、杂志却寥寥无几。我们期望《软件开发文集》能够弥补这方面的空白，并且不辜负广大用户的期望，把握好学术方向，确定好信息服务和技术支持的层次和深度，把《软件开发文集》办成深受广大用户喜爱的丛书。

由于时间仓促，《软件开发文集》难免存在这样或那样的问题，敬请广大的用户及读者提出宝贵意见和建议，以利改进，为我国软件产业的发展作出应有的贡献。

《软件开发文集》编委会
1996.3.25

和光集团

Microsoft 中国总代理



和光集团各地分支机构

北京和光达文有限责任公司
Tel: (010)62625960-63, 62615754,
62611032, 62617782
上海和光新技术有限责任公司
Tel: (021)62761186, 62762788,
62762995
广东和光有限公司
Tel: (020)7577213, 7549192, 7577207
成都和光电子有限责任公司
Tel: (028)5548804

和光集团沈阳分公司
Tel: (024)3909277
哈尔滨和光电子技术有限责任公司
Tel: (0451)6415255, 6415874
大连和光分公司
Tel: (0411)2603921, 2807730
长春和光分公司
Tel: (0431)5670145, 5621342

和光
DAWN 和光集团

目　录

1

部件对象模型

一、引　言

部件对象模型(COM)是一种部件软件体系结构，这种结构允许应用程序和系统由不同软件供应商提供的部件构成，COM 是构成类似 OLE 提供的更高水平软件服务系统基础的基本体系结构。OLE 服务系统囊括部件软件的各个方面，包括：复合文档、自定义控件、交互应用程序脚本、数据传送及其他软件的交互作用。

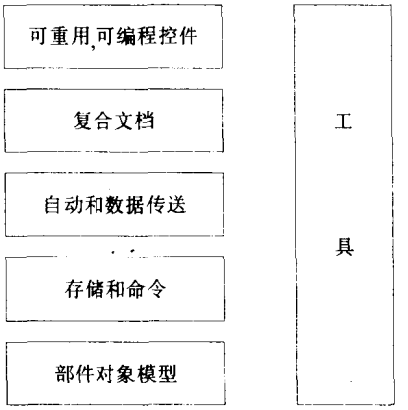


图 1

这些服务系统为用户提供了独特的各种不同的函数功能，对同一机理共享一个基本要求，允许由不同的软件供应商供应的二进制软件部件按一种定义好的方式相互连接和通信。这种机制由 COM 提供，部件软件体系结构如下：

(1) 定义一个部件互操作性的二进制标准。

(2) 独立的程序设计语言。

(3) 提供多种平台(Windows，Windows NT，Macintosh，UNIX)。

(4) 为基于应用程序和系统部件的健壮进展做准备。

(5) 可扩充。

此外，COM 提供如下机制：

(1) 部件间通信，可跨越进程和网络界限。

(2) 部件共享内存管理。

(3) 错误和状态信息报告。

(4) 部件的动态加载。

认识到 COM 是一个部件软件通用体系结构很重要，微软公司应用 COM 发布了如控件复合文档、自动化、数据传送、存储、命名及其他特定领域，每个开发者均可利用 COM 提供结构和基础。

COM 如何进行相互操作?是什么使它成为这样一个有用和统一的模型?为了回答这个问题首先定义基本 COM 设计原理和体系结构的概念是必要的。为此，需知道 COM 所要解决的这些特定问题及 COM 如何为解决这些问题而提供的解决方案。

在读过本文之后，请务必阅读“CODE article title here”，以便了解一个简单的 COM 对象是如何执行和被使用的，并阅读“OLE article title here”，看看 OLE 如何在 COM 体系结构的顶部提供更高层次的服务系统。

二、部件软件问题

COM 发表的最根本的问题是：如何设计一个系统使得由不同供应商在世界上的不同地区和不同时期编写的二进制可执行程序能够相互操作?为解决这个问题，我们必须首先解决下面四个特定问题:

(1) 基本的相互操作性：开发者是怎样创造自己独特的部件，并且确信这些部件与不同的开发者所创建的其他部件能够相互操作?

(2) 版本：如何在不要求系统全部部件升级的条件下，使一个系统组件升级?

(3) 语言的独立性：如何保证用不同语言编写的部件互相通信?

(4) 透明的交叉进程互操作性：我们给开发者怎样的灵活性，使他们能用一简单的程序设计模型来编写部件，使其能在内部进程或交叉进程(最终是交叉网络)中运行?

另外，高性能是对部件软件系统结构的一个要求。交叉进程和交叉网络的透明度是一个值得称赞的发展目标，对二进制部件市场的商业成功是至关重要的，在这个市场中在同一地比空间的部件能够利用相互服务系统而需要过高的“系统”。否则，部件将不现实地削减到如 C++类或 GUI 控件那样小而轻的软件。

三、COM 基础

COM 定义了几个基本概念，以提供模型结构的支柱。包括:

(1) 部件间函数调用的二进制标准。

(2) 为更严格定义的(strongly-typed)函数组进入界面而提供的规划。

(3) 基本接口提供下面内容:

——部件能动态地发现由其他部件实现接口的方法。

——使部件能跟踪本身的生存期并在适当时候能够进行自身删除的引用计数。

(4) 独特的识别部件及其接口的机制。

(5) 部件加载器是为了设置部件的相互作用，另外在交叉进程和交叉网络情况下，该“加载器”能帮助管理组件相互作用。

(一) 二进制标准

对任一给定的平台(硬件及操作系统的结构)， COM 定义一个在内存中设计虚拟函数表(V 表)的标准方法以及通过 V-表调用函数的标准方法。这样，凡是能通过指针调用函数的任何语言(C， C++ ， Smalltalk， Ada 甚至 Basic)都可用来编写部件，这些部件与另外一些按同一二进制标准编写的部件可相互操作。双重间接性能(客户机保持一指针对指针、指针对 V 表)使 V 表在同一对象类的多种实例中共享。在一个有对象实例的系统中， V 表共享可大量减少对内存的要求。

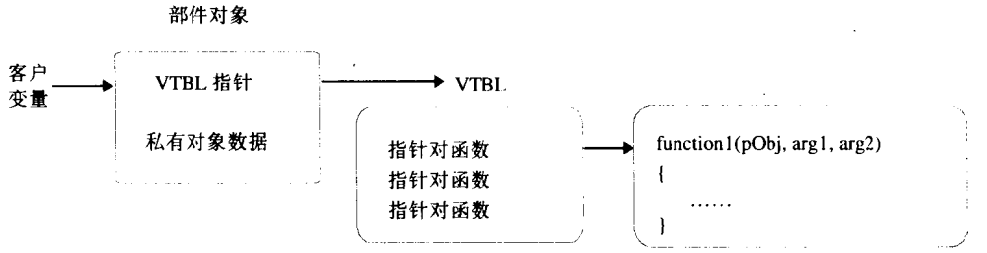


图 2

(二) 对象和部件

“对象”这一词对不同人有着不同的含义，为了澄清，在 COM 中，对象就是一些对系统的其余部分提供某种服务的、已编译的一段代码。为避免混淆，最好把将一个 COM 的对象称为一个“部件对象”或简单的说是一个“部件”，这就避免了把 COM 对象与在 C++中定义的源码 OOP 对象相混淆。部件对象支持一个叫做 IUnknown 的基本接口，该基本接口与其他接口的部件合并一起，依赖于部件目标选取的函数功能。

部件对象通常有一些有关的数据，但与 C++对象不同，一个指定的部件对象不含直接访问另一部件对象的整体，相反，部件对象经常通过接口指针访问其他部件对象，这是 COM 的主要体系结构特性，因为它允许 COM 完全保持数据和进程的封装，这种封装是真正的部件软件标准的基本要求，它同样考虑到了远程的透明度(交叉进程和交叉网络调用)，由于所有的数据都是通过存在于一个代理对象的方法进行访问的，这个代理对象转寄需求和无线电导引返回应答。

(三) 接口

在 COM 中，各个应用程序通过函数集合使其彼此之间及与系统之间的交互作用叫做接口，要知道所有的 OLE 服务系统都只是 COM 接口，一个 COM 接口就是两个软件部件间的严格约定，以提供一套小而有用的有关操作(方法)的语义。一个接口是一个期望的行为和期望的责任的定义。 OLE 的拖放支持是个很好的例子，部件必须执行的目标为拖放的所有函数功能集中在 IDropTarget 接口，全部函数功能集中在 IDragSource 接口。

接口名字习惯上以“I”开头。 OLE 提供大量有用的通用的接口(一般以“IOLe”开头)，个人也可以定义接口。我们期望客户在他们配置基础部件应用程序时开发自己的接口。另外一个部件对象指针实际上是一组件对象实现的一个接口指针。这意味着就

像上面所述那样，只能用部件对象调用一种方法而不去修改数据。下面具有两个成员方法 ILookup 接口的例子:

```
interface ILookup:public IUnknown
{
    public:
    Virtual HRESULT_stdcall Lookup By Name(LPTSTR lpName,TCHAR **lplpNumber)=0;
    Virtual HRESULT_stdcall Lookup By Number(LPTSTR lpNumber,TCHAR **lplpName)=0;
};
```

(四) 接口属性

假定就是部件对象对其服务系统的一种契约方式，那么有四个非常重要的问题需要了解:

(1) 接口不是类: 类可被具体实现而组成一个部件对象，但接口由于它没有携带执行程序而无法被它自己具体实现。部件对象必须执行那个接口，并且这个部件对象必须被具体实现有一个接口。而且，不同的部件对象类可以不同地执行一个接口，只要程序执行过程遵循接口的定义(比如两个对象执行 IStack ，一个用数组而另一个用链接表)。这种多组合的基本原理充分用于部件对象。

(2) 接口不是部件对象: 接口只是一个相关函数组，并且是客户机与部件对象通信的二进制标准。只要部件对象向接口成员函数提供指针，它就可以按照具有任何内部状态表示的语言来执行。

(3) 客户只能通过指针与接口交互作用: 当客户访问部件对象时，只有一个指针通过接口访问函数，简称为接口指针。指针是不透明的，它将内部执行过程的各个方面都隐藏起来。你不能了解部件对象的数据。这与 C++对象指针相反，客户通过 C++对象指针可直接访问对象的数据。在 COM 中，客户只能调用有一个指针接口的方法，这种封装允许 COM 提供一个二进制标准，以实现本地/远程的透明。

(4) 部件对象可执行多个接口: 部件对象可以——并且是典型地——执行几个接口。这就是说，类能提供几套服务系统。例如，类可支持按照客户要求将它的持久性状态信息(这些数据将在返回到当前状态时重新装载)存入一个文件中。这些能力都由不同接口来表示(IDataObject 和 IPersistFile)，所以部件对象必须执行两个接口。

(5) 接口是严格定义的: 每个接口都有其自己的接口标识符(GUID，下面介绍)，这就消除了与人名发生冲突的可能性。部件与接口的区别有两个重要的含义: 如果一个开发者创造一个新接口，他必须给这个新接口一个新的标识符，当一个开发者使用一个接口时，他必须用这个接口的标识符向接口请求一个指针。这个显示标识通过消除命令的冲突(该冲突将导致运行故障)而提高了系统健壮性。

(6) 接口的不变性: COM 接口从未定义过版本，这表明避免了新旧组件之间的版本冲突。一个接口的新版本是采用添加更多的函数或改变语义来创建的，是一个全新的接口，并指定了独一无二的标识符。因此，即使全部改变也只是在现有方法的操作和语义上(甚至语法也不变)，这就不会引起新旧接口的冲突。注意一种执行方式可能有两个

类似的接口共享一个公共的内部执行程序的情况，例如，如果一个新接口在有接口中添加一个方法，并且部件作者希望支持新旧两个客户，他将通过两个接口表述两个性能的集合，然而老接口的内部执行程序只能作为新接口执行程序的一个适当的子集。

对于对象及其接口来说，采用标准的图示表示是很方便的。图 3 是采用惯例，在对象的每一接口画一个“插座”。

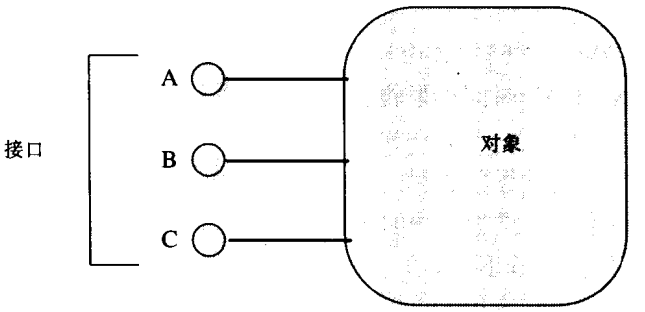


图 3　支持三个接口 A，B 和 C 的部件对象图

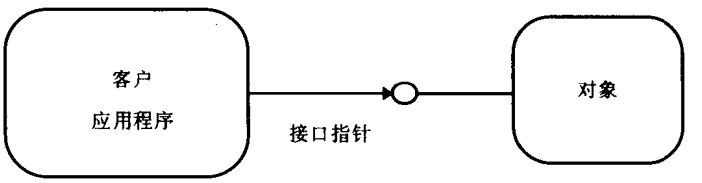


图 4　客户与对象通过接口相连接

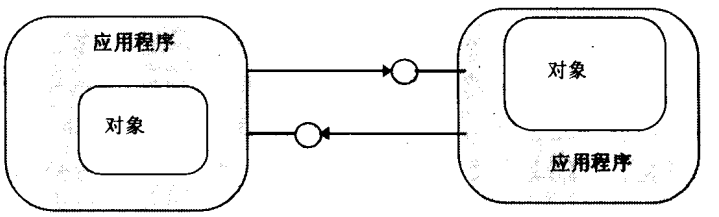


图 5　两个应用程序与对象通过接口彼此相连接

在 COM 中，独特的接口提供五个主要优点:

(1) 在应用程序中对象客户或服务系统函数功能的能力不断进展: 这是通过调用完全由 COM 对象支持的(除非不是 COM 对象)Query-Interface 来执行的。Query-Interface 允许一个对象有多个接口(即: 支持新函数组)，这些接口对新客户来说是可用的，并且完全保留了与现有客户代码二进制相容性。换句话说，通过添加新函数功能修正一个对象并不需要对现存客户代码的任何部分进行重新编译。这是对版本问题的关键解答方案，并且这是获取部件软件市场的基本要求。因为 COM 接口不变性，并且由于部件即使通过添加接口来添加新函数功能时仍继续支持旧接口，所以 COM 提供了健壮版本。当部件升级时，就保证了向后兼容性。另一方面，其他系统对象模型通常允许开发者改变现存接口，但当部件升级时，将最终导致版本问题。这些可在表面上解决版本

问题，但我们没有看到实际有效的工作，比如，如果只在建立对象时检查版本，后来的具体执行对象用户会很容易失败，因为对象虽是正确类型但版本是错的(检查每一调用版本代价太高甚至不敢问津)。

(2) 快速和简单对象交互作用：一旦一个客户建立了同一个对象的连接，对对象服务系统的调用(接口函数)只不过是通过两个内存指针的间接函数调用。结果当调用代码时，与内部进程 COM 对象(在同一地址空间的对象)交互作用的性能开销可忽略不计。在同一进程中，COM 部件间的调用只是少量的处理器指令，比标准直接函数调用要慢，但与激活 C++对象的编译时间界限相比并不慢。另外，由于与接口的对话是按函数组来做的(通过 Query Interface)，不是一次一个函数，所以对每一对象使用多接口是有效的。

(3) 接口重用：设计经验表明，在广泛的部件领域中，很多套操作系统是有用的。比如，为读写字节串而提供和使用的函数集是普遍有用的。在 COM 中，部件可在多种领域重复使用现存接口(如 IStream)。不仅对代码可重复使用，而且通过重用，程序设计者了解了接口，就可将它应用于很多不同的应用程序中。

(4) “本地/远程透明度”：二进制标准允许 COM 禁止接口调用一个对象，而该对象正在另一进程或另一机器上运行。关键一点就在于调用者所用的这次调用准确到就好像在同一进程中进行调用。这种二进制标准实现了 COM 在进程间(interprocess)和交叉网络函数调用的透明度。虽然在远程过程调用要花费更多的开销，但是客户没有必要区分内部进程对象和外部进程对象。这就表明，只要客户开始被登记操作， RPC 例外，对客户可用的全部对象(内部进程、交叉进程和远程)都采用均匀、透明的形式。微软公司将提供分布式的 COM 版本，但现有部件无需为获得分布式能力而做任何改动。即是说，程序设计者完全独立于网络问题之外，事实上当将来的 COM 版本公布，对今天的部件仍将以分布式形式操作。

(5) 程序设计语言独立性：每个可以产生指针结构或以显示的、隐式的方式通过指针调用函数的程序设计语言都可建立并使用部件对象。可以以大量不同的程序设计语言来执行，它也可以被使用完全不同的程序语言所编写的程序所使用。又由于 COM 与面向对象程序语言不同， COM 代表一个二进制对象标准，而不是源码标准。

(五) 全球独特的标识符(GUID)

COM 使用全球独特的 128 位整型的标识符，这保证了在全世界跨时空的独特性，以它来标识每一个接口和每一部件对象类。这些全球独特的标识符是由“开放软件基础的分布式计算环境”定义的 UUID(Universally Unique ID)。人类用名只是为方便而起的，并且是局部范围的。这有助于保证 COM 部件甚至在上百万部件对象网络中也不会意外地与一个“错误”的部件、接口或方式连接。 CLSID 是适用于部件对象类的 GUID，IID 是适用于接口的 GUID。微软公司提供了一个自动产生 GUID 的工具(uuidgen)。另外， CoCreate Guid 函数是 COM API 的一部分。这样，开发者在开发部件对象和自定义接口时，可建立他们自己的标识符。通过定义，开发者不用直接使用真的 128 位 GUID 。对于那些想见识一下真的 GUID 风采的人，下面给出两个 GUID 的例子。CLSID_PHONE BOOK 是一部件对象类，它为用户提供了查找一个电话簿的途径。IID-ILOOKUP 是自定义接口，由 PhoneBook 类执行，以访问电话簿的数据库:

```
DEFINE_GUID(CLSID_PHONEBOOK.0xc4910d70,0xba7d,0x11cd,0x94,0xe8,0x08,0x00,0x17,0x01,0xa8,0xa3);
DEFINE_GUID(IID_ILOOKUP.0xc4910d71,0xba7d,0x11cd,0x94,0xe8,0x08,0x00,0x17,0x01,0xa8,0xa3);
```

GUID 嵌入于部件二进制本身，并被 COM 系统在结合时间动态地使用，以保证两部件之间的连接无误。

(六) Iunknown

COM 定义二个特殊的接口 IUnknown，以执行某些基本的函数功能。所有的部件对象都要求执行 IUnknown 接口，并为方便起见，所有其他的 COM 和 OLE 接口由 Unknown 驱动。IUnknown 有三个方式：Query-Interface，AddRef 和 Release。在 C++ 语法中，IUnknown 有如下形式:

```
interface IUnknown {
    virtual HRESULT    QueryInterface(IID& iid,void** ppvObj) = 0;
    virtual ULONG      AddRef() = 0;
    virtual ULONG      Release() = 0;
    }
```

AddRef 和 Release 是简单的引用计数方法。当其他部件对象使用这个接口时，调用部件对象的 AddRef 方法；当其他部件对象不再要求使用接口时，调用部件对象的 Release 方法。当部件对象的引用计数为非零时，它必须保留在内存中；当部件对象的引用计数变为零时，部件对象可安全地自身卸载，因为没有其他组件对它保留引用。

Query-Interface 是实现客户动态查找是否一个接口被部件对象支持的机制，同时，它也是客户用来从部件对象得到接口指针的机制。当一个应用程序要使用部件对象的某个函数，它调用对象的 Query-Interface，向接口请求指针，以实现所需要的函数。如果组件对象支持这个接口，它将返回适当的接口指针和一个成功的代码。如果部件不支持所要求的接口，它将返回一个错误值。这个应用程序检查返回代码，如果成功，它将用接口指针访问所需要的方法。如果 Query-Interface 失败，这应用程序将采取其他行动，使用户知道期望的方法无法得到。

下面的例子说明在部件 PhoneBook 上，调用 Query-Interface。我们询问这个部件“你是否支持 Ilookup 接口?” 如果这个调用成功地返回，我们知道部件对象支持 Ilookup 接口(或者 LookupByName 或 LookupByNumber)的方法的指针。如果这个调用没成功，我们就知道部件对象 PhoneBook 没有执行 Ilookup 接口。

```
LPLOOKUP*Plookup;
TCHAR szNumber[64];
HRESULT hRes;

// call QueryInterface on the Component Object PhoneBook, asking for a pointer to the
// Ilookup interface identified by a unique interface ID
```

```
hRes = pPhoneBook->QueryInterface(IID_ILOOKUP, &pLookup);
if( SUCCEEDED( hRes ) )
{
    pLookup->LookupByName("Daffy Duck", &szNumber);    //use Ilookup interface pointer
    pLookup->Release();                                //finished using the IPhoneBook interface pointer
}
else
{
    // failed to acquire Ilookup interface pointer
```

注意：因为在返回一个接口指针之前， Query-Interface 添加了引用计数，所以AddRef()不是显式调用。

(七) 部件对象库

部件对象库是一种系统部件。提供 COM 技巧。部件对象库提供 IUnknown 调用交叉进程的能力；它也封装了所有与发射部件和建立部件间连接相联系的“准备工作”。典型地，当一个应用程序建立一部件对象时，它将部件对象类的 CLSID 传送到部件对象库。部件对象库用 CLSID 在登录数据库中查找相关的服务程序代码，如果该服务程序是可执行的， COM 发射激活 EXE 并等待它通过对 CoRegister Class Factory 的调用来进行类工厂的注册(类工厂是在 COM 中例证新部件对象的机制)。如果代码刚好是一个 DLL， COM 就加载 DLL 并且调用 Dll Get Class Factory。 COM 用对象的 IClass Factory 去询问类工厂，以产生部件对象的一个示例，并将一个指向调用程序的指针返回给被请求的接口。这个正在调用的应用程序不必知道服务器应用程序在哪运行；它只用返回的接口指针与新建立的部件对象进行通信。部件对象库是在 Windows 的 COMPOBJ.DLL 中执行，并且也可在 Windows NT 和 “ Windows 95 ” 上的 OLE 32.DLL 中执行。

概括一下， COM 定义了几个基本原理，它们是对象模型的支柱。二进制标准允许按不同语言编写的部件相互调用函数。接口是相关函数(这些函数在一起可以完成某些定义好的功能)逻辑组。 IUnknown 是 COM 定义的接口，使部件能自身控制其运行时间，同时还能动态决定其他部件的性能部件对象执行 IUnknown，以控制它的运行时间和访问它所支持的接口。部件对象不直接访问数据。 GUID 对每一类和接口提供独特的识别符，借此避免命名冲突。最后，部件对象库是作为操作系统的一部分来执行并且具有与查找和发射部件对象关联的“准备工作”。

现在我们对 COM 的基本原理有了很好的了解，让我们看看这些原理如何组合在一起形成部件软件。

四、 COM 解答部件软件问题

COM 论述了与部件软件有关的四个基本问题:

(1) 基本部件互操作性。

(2) 版本。

(3) 语言独立。

(4) 透明交叉进程互操作性。

另外， COM 具有符合商业部件市场要求的高性能体系结构。

(一) 基本互操作性和性能

这是由 COM 使用 Vtable 给出，以定义部件间调用方法的二进制接口标准。在同一进程中， COM 部件间的调用只是少数处理器指令比标准的直接函数调用慢，但并不比激活 C++对象所限制的时间慢。

(二) 版本

一种好的版本机制允许在更新一个系统部件的同时不更新其他部件。 COM 中的版本更新是通过接口和 IUnknown： Query Interface 实现的。 COM 的设计使得完全不必要采用版本存储以及部件版本的集中管理。

版本更新软件模块时，一般要添加新的函数功能，或改进现存函数功能。在 COM 中，采用对接口添加支持的方法来为部件对象添加新函数功能。因为现存接口不变，依赖这些接口的其他部件可以继续工作。更新的部件知道关于新接口，可以用新公开的接口情况。由于Query-Interface 调用是在运行时间不花任何代价而产生一些“功能数据库”(像在某些其他系统对象模式中使用的那样)，一个部件对象的目前能力可以有效地评估每次使用的部件；当新特征可用时，应用程序知道怎样使用这些特征并立即开始执行。

提高现有函数功能是很容易的。因为接口的语法和语义保持不变，可以自由改变接口的执行程序，而不必中止其他开发者依赖该接口的部件。例如，你有一个支持(假设)IStack 接口的部件。该接口包括类似 Push 和 Pop 的方法。目前你已执行接口，但你决定用链接表更适合。因为所有方法和参数不变，你可随意地用新的执行程序代替老的执行程序，并且使用你的部件应用程序，将获得改进的链接表函数功能。

Windows 和 OLE 采用这种技术来提高系统支持能力。比如，目前在 OLE 中，Structure Storage 是作为一套接口来执行，这些接口通常内部采用 C Runtime 文件 I/O 函数。在 Cairo(Windows NT 的未来版)，这些同样的接口直接写到文件系统上。接口的语法和语义保持不变，只改变了执行程序。现有的应用程序在不做任何改变的情况下，能使用新的执行程序；它们“成为免费使用”提高了函数功能。

用接口组合(不变的、定义界限分相的“函数功能集”，它们是由部件压制而成)及 Query-Interface(在运行时，能方便地决定特定部件对象的能力)能使 COM 提供一个体系结构，在该体系结构中，部件可以动态更新，但对其他可靠部件不需更新。这是 COM 超过其他对象模型的基本实力。 COM 解决了版本更新/发展问题，在未提出现有客户不兼容的条件下，对象的函数功能可以独立于客户而改变。换种说法， COM 定义一个系统，在该系统中的部件继续支持对老的客户提供服务的接口，并支持新的和更好的接口，而通过这些接口部件能为更新的客户提供服务。在运行时，老的和新的客户可以与给定的部件对象共同安全相处。错误仅出现在容易处理的时候：绑定时间和调用 Query Interface 期间。没有随机碰撞的机会，就好像对一个对象预料的方法不存在或参数已改变时，不会出现碰撞一样。