计算机科学2001Vol.28№.12

基于 COM/DCOM 技术的共享电子白板实现

Implementation of Shared Whiteboard System Based on COM/DCOM Architecture

褚南俊　张　波　潘金贵

（南京大学软件新技术国家重点实验室　南京大学计算机科学与技术系　南京210093）

Abstract　With the appearance of internet, component software is applied extensively. COM/DCOM is a component software architecture that defines a binary standard for component interoperability. It not only provides a good solution to the component software problem and a high-performance architecture, but also simplifies and accelerates programmers' learning and working. This article describes the support for RPC and the unique thread model in COM/DCOM and gives out an implementation of the shared white board system based on COM/DCOM architecture.

Keywords　COM, DCOM, RPC, MTA/STA

引言

Internet 出现以后，软件常要支持多个用户访问复杂的数据库，支持在网络上多机环境中工作，支持易修改和可复用等功能，软件组件的设想应运而生。组件(component)就是一个软件块，完成专门的预定工作；并可被任何程序或其它组件使用。组件可一次编写，而后通过更新或替换接口来改进其功能。

但是，要让这一系统工作，必须要有一个建立组件的标准，保证其兼容性和互换性。那么一个最基本的问题[1]就是：系统如何设计才能使得由不同软件商提供的、在世界上不同地方和不同时间写成的二进制可执行代码进行交互？具体地讲，这个问题包括四个方面：

基本交互性：开发者如何能创建他们自己唯一的组件，同时保证这些由不同开发者开发的组件能正常交互。

版本控制：一个系统组件如何在其他系统组件不升级的情况下单独升级。

语言独立性：用不同语言开发的组件之间如何通信。

透明跨进程交互性：如何提供一种灵活性，使得程序员可以用一种编程模型开发进程内或跨进程(最终跨网络)运行的组件。

此外，高性能也是对组件结构的一个要求。否则，组件就不可能成为小规模、轻量级的类似于 C++ 类或 GUI 控件的软件。

COM/DCOM 很好地解决了上述这些问题。以下，本文简要介绍 COM/DCOM，然后阐述基于 COM/DCOM 的共享电子白板的实现。

1. COM/DCOM

COM 即 Component Object Model(组件对象模型)，是由微软提出的构造二进制兼容组件的规范。它使得系统和应用程序能够架构在由不同的软件商提供的组件之上。DCOM 是分布式 COM，它使得组件可以在使用本组件之外的机器上运行。标准 COM 组件与使用组件的软件在同一台机器上运行，位于同一进程空间。而 DCOM 组件通常在不同机器上运行(但也不一定)，在自己的进程空间运行。无论本地还是远程运行，组件的功能都提供给程序和其它组件。

COM/DCOM 是形成高层软件服务的底层结构，它提供了一个基本机制，使得不同软件商提供的二进制组件能够很好地连接和通信[1]：它定义了组件交互的二进制标准；它独立于编程语言，并且有多种平台的支持(Windows, Windows NT, Machintosh, UNIX)；它还是可扩充的。

此外，COM/DCOM 还提供下列机制：组件间的通信(包括跨进程和网络的通信)；组件间的共享内存管理；错误和状态报告；动态载入组件。

如图1所示，COM 是组件软件的通用架构，微软把它用于控件，复合文档，自动化，数据传输，存储和命名等领域；而开发者则可以利用它所提供的结构和基础。

褚南俊　硕士，多媒体技术应用。张　波　硕士，网络多媒体。潘金贵　教授，从事多媒体技术和多媒体远程教学系统等方面的研究。

遵循COM规则，构造的组件可以和其他组件(不论这些组件的作者是你还是别人)通信。COM论述了组件软件的四个基本问题并提出了很好的解决方案，具有如下的优点和特点。

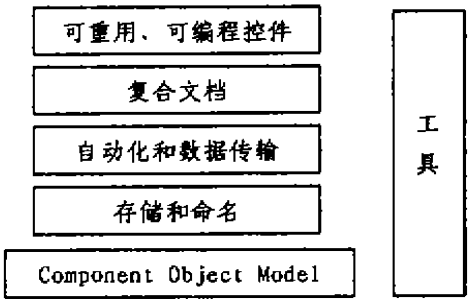

图1 COM作为高层OLE应用服务的基础

1)基本组件交互和性能 COM使用虚表概念定义了组件之间方法调用的二进制标准，正是这个标准实现了组件间的基本交互[1]。应用程序(客户或服务器对象)的功能可以随时间变化。它通过一个所有COM对象都支持的请求QueryInterface()来实现，使得对象可以对新客户生成更多可用接口(即支持几组新的函数)而同时又保持对原有客户代码的完全的二进制兼容性。添加新功能改变一个对象时，不需要对已存在客户的任何部分进行重编译。这也是解决版本问题的关键和形成组件软件市场的基本要求。此外，对象采用多接口是高效的，因为通过QueryInterface()查询接口实际上是成组查找函数，其代价显然小于查询单个函数。

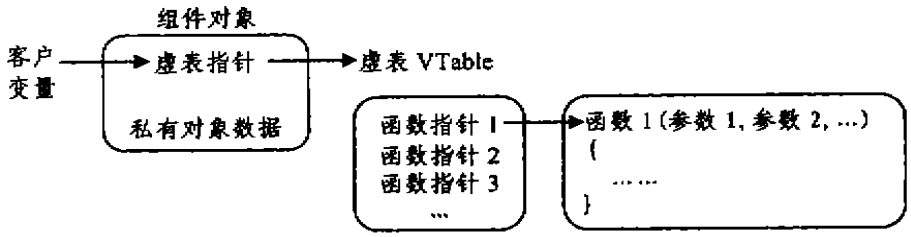

图2 COM中的函数调用机制

2)版本控制 COM的设计避免了版本储存库或组件版本集中管理问题的存在。在COM中，通过追加新接口来实现追加新功能或改进现存功能。已存在的接口不变，依赖于这些接口的组件将继续工作。而知道新接口存在的新组件将使用新接口。在运行时调用函数QueryInterface()，确定所要求的接口是新的还是旧的。接口的语法和语义都是不变的，可以自由地改变接口的实现，而不影响到依赖于这个接口的其它开发者开发的组件。在COM的体系结构中，组件在继续支持对旧客户提供服务的接口的同时，也支持对新客户提供服务的新接口。在运行时，旧的和新的客户可以安全地共存于一个组件对象的服务之下。

3)语言独立性 COM代表的是二进制对象标准，而不是源代码标准。这是COM(当然也是组件结构)相对于面向对象编程语言的一个基本优势。OOP语言中定义的对象一般只和本语言中定义的其它对象交互。这必然限制到它们的可重用性。不过，OOP语言可用于建立COM组件，这两项技术是互补的。COM可把OOP对象封装为组件以扩大它们的可重用性，使得由非常不同的语言所写的组件也可正常交互。

4)跨进程交互透明性 COM还解决了跨网络的组件间透明通讯的问题，即不同计算机上的组件间通讯可以像在同一台计算机上通讯一样，两种情况下使用的是同一接口。组件对象库是提供跨进程透明交互的关键。它封装了所有的发现并启动组件和管理组件间通讯的功能，它把组件从地域差别中隔离出来。组件对象不论是与在同一进程内、在不同进程内或在不同机器上的进程交互、任何情形下，实现或使用对象的代码都是相同的。

除了上述优点外，COM还提供了符合组件商品化市场化要求的高性能结构：对象交互快捷简单、接口重用性好、分布式组件等。

由于COM的这些优点，程序员的学习和工作也变得简单和快捷[2]：

一旦程序员学会如何处理接口，他们就学会了如何处理未来可能创建出的新的服务。这是对于每个服务都以一种不同的方式表露出来的这一环境的一种巨大改进。

COM简化了接口定义。开发者定义新的接口(即定义新的功能)时，可以使用接口定义语言(IDL)。从接口定义中，IDL编译器将产生头文件和源代码供应用程序、代理(proxy)对象和占位模块(stub)对象使用。这使得接口设计者们可集中精力于设计。

同时，COM组件有助于并行开发。COM组件易

于替换，减小了编程和调试的工作量。

2. 基于 COM/DCOM 的共享电子白板的设计

共享电子白板为多个用户提供了在线共享的可书写文本和图形的功能。即让多个用户在线共享一个画图板。我们采用客户机/服务器模式设计此软件。客户端负责提供和用户交互的 GUI，当它接收到来自用户的消息时，进行适当处理，调用服务器上适当的作图函数或接口，完成用户所需要的功能和操作。即客户程序实际上是把消息映射到远端的处理函数。当服务器处理完消息后，再把信息反馈给用户（可能包括其它的未发出消息的用户）。服务器可以同时接收来自多个客户的消息，可以为比较复杂的文本或作图处理创建新的线程，并负责线程之间的同步和并发。在系统设计上的两个主要特点如下：

首先，我们利用了 COM/DCOM 对远程过程调用（RPC）的支持。

COM 被设计为允许客户和组件透明地通信，而不论这些组件在哪里运行（同一进程内，同一机器上不同进程内，或不同机器上）。不论是对客户，还是对服务器而言，各种组件对象只有一个编程模型。

从客户观点看，所有组件对象都通过接口指针被外界访问。指针必须是在进程内的，而实际上任何对接口函数的调用总是先到达进程内某段代码。若组件对象在进程外，调用首先到达一个由 COM 提供的代理对象（proxy），而由 proxy 来产生适当的对其它进程内或其它机器上函数的远程过程调用（RPC）（如图3所示）。当然，客户从一开始就应编写好对 RPC 异常的处理；这样它才能够透明地连接到本进程内，或其它进程内，甚至是其它机器上的对象。

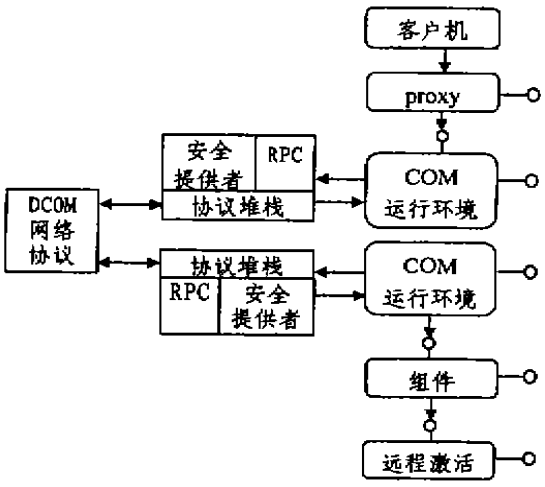

图3　DCOM 中的远程过程调用的模型

从服务器观点看，所有对于组件对象接口函数的调用都是通过一个指向这个接口的指针来实现的。同样，指针只知道同一进程内的上下文，并且调用的是同一进程内的某段代码。如果组件对象在同一进程内，调用者就是客户自身；否则，调用者就是一个由 COM 提供的占位模块对象（stub），它专门用来接收客户进程中的 proxy 提出的 RPC 请求并把请求变为对服务器组件对象某个接口的调用。

在客户和服务器看来它们总是在和同一进程内的某段代码进行通讯。这种本地/远程透明性提供了一个不受客户和服务器之间距离影响的问题一般解决方案。正是由于这种跨进程交互透明性，程序不需要经过任何额外的修改即可具有处理远程过程调用的能力。程序的简单模型如图4所示。

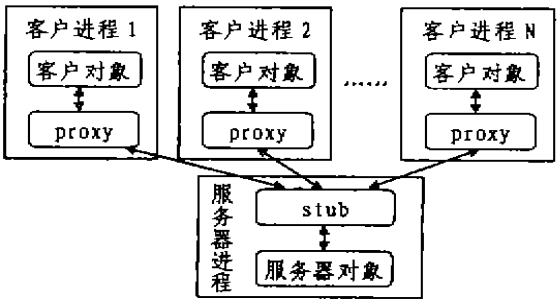

图4　共享电子白板的基本模型

其次，我们还利用了 COM/DCOM 对多线程的独特支持。

COM 有自己独特的线程模型。它引入公寓（apartment）概念来抽象线程。公寓不同于线程和进程等实际内核对象，它可以认为是执行环境，包含一个或几个对象。每个 COM 对象只能处在一个公寓内，而一个公寓可以有一个或多个 COM 对象。COM 对象共享一个公寓时，要遵守一些关于并发行和再入性的共同规则。COM 公寓有两种：单线程公寓（STA）和多线程公寓（MTA），后者又称为自由线程公寓[3]。COM 控制公寓的"钥匙"，并决定由哪个线程持有"钥匙"。只有持有"钥匙"的线程才能进入公寓。STA 只有一付钥匙，只能由一个线程进入。MTA 有数目不限的钥匙，允许多个线程同时进入，见图5。STA 中的对象在并发线程访问时是安全的，因为任何时刻只能有一个特定线程进入公寓。MTA 中的对象可由 MTA 中的任意线程并发访问，程序员必须做好并发和同步控制，编写线程安全组件。这样就能够建立和使用工作线程，改进通过量或增加组件对用户的响应性。

COM 服务器常用的线程模型有三种：单线程，单元线程和自由线程。单线程中，全部 COM 对象存在于

同一线程中，单线程拥有一个单元，在那里对 COM 对象的离线调用被串行化了。这个单元通常是 STA。单线程服务器易在消息队列方面造成瓶颈，解决的方法就是创建额外的单元线程（单元线程模型）或允许多个线程存在于一个单元中（自由线程模型）。单元线程服务器有一个或多个 STA。每个对象只存在于一个单元中，它们的数据无并行问题；但它们共享全局数据且并发运行，全局数据存在并行问题。自由线程服务器中，COM 对象可跨多个线程，对象的数据和全局数据都引入了并行问题，需要进行控制。

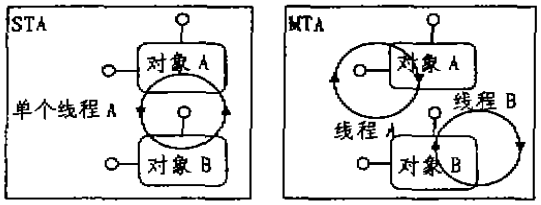

图5 STA 和 MTA

我们采用自由线程服务器，共享白板对象存在于一个多线程单元中，对于比较复杂的操作创建新的进程以完成其任务。

3. 系统实现

在实现工具的选择上，服务器端用 ATL 开发，客户端用 MFC 开发。开发步骤如下：

3.1 建立服务器

（1）对多线程和 RPC 支持　用 ATL COM AppWizard 创建一个名字叫做 WBServer 的服务（Service）作为服务器。把其中 CServiceModule::Run() 函数中的

HRESULT hr = CoInitialize(NULL);

改为

HRESULT hr = CoInitializeEx (NULL, COINIT_MULTITHREADED);

这样就可以支持多线程和 RPC 调用。

（2）封装作图操作函数　在 WBServer 中添加一个类 CWBService，这个类有一个接口 IService，其中封装了完成具体作图的各个函数。

3.2 建立客户程序

客户程序 WBClient 用 MFC AppWizard(exe) 建立，采用通常的文档视图结构，其功能类似于 Windows 中的绘图板。

（1）添加建立/断开连接的函数　首先为 CWBClientView 添加两个消息映射 ON_ID_CONNECT、ON_ID_DISCONNECT 和响应函数 OnConnect、OnDisconnect。在 OnConnect 中调用函数

```
CoCreateInstance (CLSID_CWBServ, NULL,
  CLSCTX_SERVER,IID_IWBServ,
  (void * * )&m_pWBServ);
```

完成实例化。而后，再调用 Connect 函数完成具体的连接服务器的工作。Connect 函数主要是调用 LPCONNECTIONPOINT 类型变量指向的 Advise() 函数或者直接调用 ATLAdvise 函数来把客户和服务器连接起来。

相应地，OnDisconnect 函数调用 Disconnect 函数，在其中调用 LPCONNECTIONPOINT 类型变量指向的 Unadvise() 函数或者直接调用 ATLUnadvise 函数来结束客户和服务器间的连接。

（2）定义其它消息响应函数　为 CWBClientView 定义其它消息映射和消息响应函数，这些在形式上均与一般的绘图板类似。但在函数内部实际上是通过接口指针调用远程服务器上的相应处理函数。例如，画一条直线的函数调用如下：

```
m_pWBServ->DrawLine(startpoint, endpoint);
```

客户直接调用远端函数，而在 IService 中的 DrawLine() 函数负责实现画线功能，并通知其它客户显示此线。

3.3 多线程支持

（1）创建线程　对于比较复杂的操作，IService 中的函数会创建新的线程完成任务。例如，创建一个新的线程用于画一个圆：

```
.....
.....
pData= new CRoundData;
pData->center=pre-point;
pData->radius=Distance(pre-point,cur-point);
HANDLE hThread;
DWORD threadID;
hThread=CreateThread(pSecurity, stacksize, DrawRound-
Proc,
pRoundData,0,&threadID);
.....
.....
```

（2）线程同步控制　对于线程之间的同步，通过调用函数 CoInitializeEx() 和 CoUninitialize()（这两个函数分别负责为当前公寓打开/关闭 COM 库，并完成相应的初始化/清除工作）进行控制：

```
DWORD WINAPI DrawRoundProc(void * pt)
{
  CoInitializeEx(NULL, COINIT_MULTITHREADED);
  CRoundData * pData=(CRoundData * )pt;
  .....
  .....
  CoUninitialize();
  return hresult;
}
```

此外还采用了 Semaphore 和 Mutex 对象（主要是为了控制同时操作的数目和某些操作如 eraze 等）。由于 ATL 中没有封装好的用于同步控制的类，我们必须自己创建 Semaphore 对象和 Mutex 对象。

结语　COM是目前广泛使用的组件软件，它提供

虚拟企业中动态的过程知识管理的研究

The Dynamic Process Knowledge Managements in Virtual Enterprises

王 茜

(东南大学计算机科学与工程系 南京210096)

Abstract In this paper, the importance and phases on the management of dynamic process knowledge are mentioned in view of dynamic changes of business processes in virtual enterprises. Recently, much attention from both Workflow area and CSCW area has been paid to the integration of the definition and the execution on flexible、dynamic (or emergent) enterprises' processes. Two typical systems that come from two areas are introduced. The system architectures, common properties and their contributions on the process knowledge management are discussed.

Keywords Process knowledge, Virtual enterprise, Emergent process, Coordination

1 概述

市场的全球化和竞争的日益剧烈,迫使若干个企业常常会为了生产某个或某几个产品在短时间内组成联盟——虚拟企业,当这个或这些产品不再赢利时,这种联盟就解散。在这种动态联盟中每个企业是业务伙伴。这些业务伙伴在地理位置上是分散的,可以是不同地域,不同国家。各业务伙伴各自独立运行在异构、分布的平台上。由于新产品的生命周期越来越短,企业为了要以新产品抢占市场,必须要不断地开发新产品。为了适应新产品的需要,合作伙伴会不断变更,业务过程也会作相应的调整以降低成本,缩短时间,提高产品的性能和质量。

2 动态的过程知识

2.1 过程知识

有两种形式的过程知识,第一种是潜在的/内在的知识(tacit /internal knowledge)。个人的经验、个人对于领域的一些心照不宣的理解,例如,个人对于如何完成项目、如何工作的认识。团队对于某一事务的共同的理解,这些理解隐含在他们日常工作的习惯中,或者在他们的潜意识中。这些知识存在于意识,不可捉摸[4]。每个人都有自己对事物的认识,不同的人有不同的观点。在虚拟企业中任何一个人要想成为所有领域的专家是不可能的。只有通过团队的协同交互,个人的潜在知识才能得到提取、分析并综合成为整个企业的资源。

第二种是明确表示的知识[4]。对潜在的知识进行记录,使它们形式化,表示为业务过程。这种外部化了的知识中含有团队成功执行任务的详细内容,而且它还和"为什么"(why),"如何"(how)有关——而不仅仅表示"什么"(what)[5]。这样所获得的业务过程就是外部化了的过程知识。经过评估之后,这个业务过程可以作为模板存储在过程库中。

2.2 紧急出现的过程

丰富的集成服务、各种易用工具和应用程序,它还提供了很好的可复用性。在具有了二进制和网络标准后,COM(包括 DCOM)打开了编程思维上的革新之门。

组件对象模型的核心是对于组件和它们的客户之间如何交互的具体描述。作为一个具体描述,它定义了诸多软件组件间交互的标准。此外,COM 还是一个包含于组件对象库中的具体实现。

我们实现的共享电子白板系统利用了 COM/DCOM 对 RPC 和多线程的独特支持。它可以作为一个独立的系统,COM 为将来升级这个系统,实现更复杂的功能(如支持声音等)提供了方便;它也可以作为一个插件用于多媒体教学或演示系统中,COM 为其它组件正确调用它的功能提供了保障机制。

参 考 文 献

1 Williams S、Kindel C. the Component Object Model A Technical Overview [M] http://www.microsoft.com, 1994.10

2 Rofail A, Shohoud Y 著. COM 与 COM+从入门到精通[M]. 邱中潘等译. 北京:电子工业出版社,2000.4

3 Armstrong T, Patton R 著. ATL 开发指南[M]. 董梁,丁杰,李长业等译. 北京:电子工业出版社,2000.11