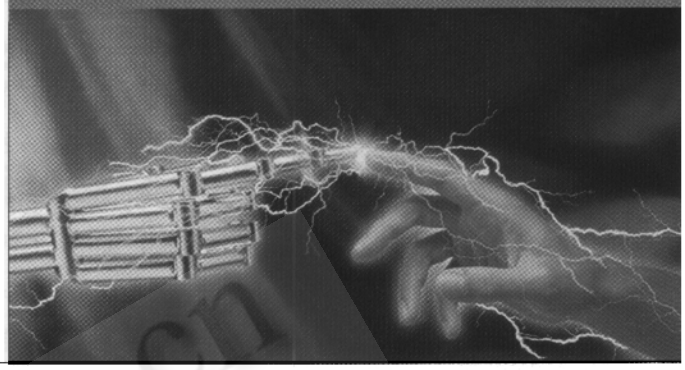
基于文档/视图结构的应用程序架构研究

Research of Application Architecture Based on Doc/View

刘惊雷 范 辉 范宝德（烟台大学计算机系 264005）

（资助项目：山东省自然科学基金项目 编号：Y2002G08）



摘 要：随着应用程序规模和复杂性的增加，应用程序架构的设计和选择的重要性已远远超过特定算法和数据结构的选择，良好的应用程序架构是保证应用系统成功的关键。本文阐述了软件复用的几种常见形式，并以微软的 Visual c++ 为例，阐述了其中的文档／视图结构及其之间的关系。最后给出了一个例子。

关键词：应用程序架构 算法和数据结构 软件复用 文档／视图

低生产率、低质量和高费用是现今软件开发需解决的重要问题。软件复用是缓和这些问题的最有效的方法之一。软件复用是指重复使用“为了复用目的而设计的软件”的过程，通过软件复用，在软件企业的应用系统的开发中可以充分地利用已有的开发成果，大大消除了在软件分析、设计、编码、测试等方面的许多重复劳动，可以提高软件的开发效率；同时通过复用高质量的已有的开发成果，避免了重复开发可能引入的错误，可以提高软件的质量，因此软件复用可以大大降低软件开发的费用，并显著地提高生产率和产品质量。

1 复用的种类

软件复用在软件开发过程中避免重复劳动的解决方案，其出发点是应用系统的开发不再采用一切“从零开始”的模式，而是以已有的工作为基础，充分利用应用系统开发中积累的知识和经验，从而将开发重点集中于应用的特有的组成成分。根据复用的程度，可以将软件复用分为以下几类：

（1）源代码复用。该方式是应用最为广泛的复用方式，它不仅表现在复用现有的代码，例如已经开发完的类、源文件、LIB库，还表现在每一代码的实现过程中。

（2）模块复用。典型的例子是Windows下的动态链接库程序DLL，一方面，多个程序可以共享这个动态库程序，另一方面，即使有多个程序调用该库中的某个函数，动态库中的代码部分也只是装入一份拷贝，这样接省了许多内存，同时为进程间的数据共享提供了一种手段。另外，目前模块复用比较活跃的技术有COM组件技术。每个COM组件有自己的属性和操作，是一个相对独立运行的实体，使用该技术构造应用程序就像是搭积木。

（3）接口复用。一个程序设计的过程定义好一套接口，而另一个应用程序可以依据这套接口来设计，也可以依据这套接口来调用另一个程序中的模块，或者提供这种接口供别的程序以及模块调用，这都是接口复用。例如微软的Visual Studio的IDE编程、IE扩展编程、屏幕保护程序等都是这样的例子。接口的定义使得软件的开发可以扩展到用户一级，由用户进行软件功能的深化或定制。

（4）应用程序复用。在一个应用程序中调用另一个程序，创建一个新的进程就是应用程序复用的惦记性例子。在Win32平台下，一般通过CreateProcess这个API函数和SHELL方面的API函数ShellExecute来创建新的进程。

（5）应用程序架构复用。

2 应用程序架构

应用程序架构是整个应用程序的可重用设计，表现为一组抽象类及类实例间交互的方法，它是可以被应用程序开发者定制的应用程序骨架。一个应用程序架构是一个可复用的设计类，它规定了应用程序的体系结构，阐明了整个设计、协作类之间的依赖关系、责任分配和控制流程。应用程序架构包括应用程序的总体组织和全局控制、通信协议、同步、数据存取，给设计元素分配特定功能，设计元素的组织、规模和性能，以及在各个设计方案间进行选择。

应用程序架构可以在具有相似需求的多个系统中得到重用，这比

代码级的重用具有更大的意义。通过对应用程序架构的抽象可以使设计者能够对一些经过时间证明是非常有效的体系结构进行重用，从而保证新的软件开发能够成功，提高软件开发的效率。在软件的设计过程中，我们常常会发现，对一个体系结构部件进行抽象就可以将它应用到其他的设计中，从而降低设计的复杂度。应用程序架构有利于形成完整的软件生产线和软件工厂，并共享公共的架构。应用程序架构的可重用性使得我们可以使用第三方应用程序架构。

随着Windows应用程序的规模不断扩大，为了提高软件的效率，也为了复用已有的应用程序架构，微软在其拳头产品VC中提供了一种很重要的应用程序架构，它就是基于文档/视图结构的应用程序设计架构。

3 文档／视图结构

大部分的Windows应用程序都要使用数据，其主要的工作可以分为两部分：一部分是对数据的管理，例如存储、复制和查询等任务；另一部分是对数据的处理和输入输出，包括显示和打印。在MFC中，VC提供了文档/视图结构来支持这类应用程序的开发。

3.1 文档／视图概念

“横看成岭侧成峰，远近高低各不同”，这段古诗隐藏了文档/视图结构的概念。“庐山”是文档概念，“成岭”是横看“庐山”这个文档所形成的“横看视图”，“成峰”是侧看“庐山”这个文档所形成的“侧看视图”。“横看视图”与“侧看视图”不同，因为它们一个“成岭”，一个“成峰”，但它们都对应着一个文档“庐山”。

在文档/视图结构里，文档可视为一个应用程序的数据元素的集合，MFC通过文档类提供了大量管理和维护数据的手段。视图是数据的用户界面，可以将文档的部分或全部分内容在其窗口中显示，或者通过打印机打印出来。视图还提供了用户与文档中数据交互的功能，将用户的输入转化为对数据的操作。以微软的WORD程序为例来说明，文档对应着某个DOC文档，视图对应着您在客户区所看到的内容，通常看的方式有页面视图或大纲视图。

在VC中，文档是靠CDocument类来实现，视图是靠CView类来实现。

3.2 文档与视图的交互

我们知道，文档对象容纳数据，而视图对象显示这些数据并允许编辑。一个SDI（单文档）应用程序有一个从CDocument类派生的文档类，有一个或多个视图类，每个视图类归根到底都是从CView类派生的。一个复杂的过程发生在文档、视图和应用程序框架的其部分之间。以下五个重要的函数可以完成文档与视图间的交互。

（1）CView::GetDocument

一个视图对象只有一个与文档相关联的对象。GetDocument函数求得了与该视图所关联的文档对象的指针。通过该指针，用户在视图下就可以对文档进行相应的操作。

（2）CDocument::UpdateAllViews

如果由于某种原因文档数据发生了改变，就必须通知所有的视图以便更新它们所显示的数据。实际上，当UpdateAllViews被调用时，引发视图类的成员函数OnUpdate的执行。

（3）CView::OnUpdate

该函数的典型应用是，在派生视图类的OnUpdate函数中调用其他函数访问文档，得到文档的数据，然后更新视图的数据成员或其他控件来反应这些变化。作为选择，OnUpdate可以使视图的一部分无效，导致视图的OnDraw函数的调用，以使文档数据在窗口中重画。

（4）CView::OnInitialUpdate

当应用程序启动时，用户从File菜单中选择了New或者用户从File菜单中选择了Open的时候，调用CView的这个虚拟函数。可以使用派生类的OnInitialUpdate函数来初始化视图对象。当应用程序启动的时候，应用程序框架在调用OnCreate之后立即调用OnInitialUpdate（如果在视图类中映射了OnCreate的话）。

（5）CView::OnNewDocument

在一个文档对象首次被构造之后并且用户从一个SDI应用程序的File菜单中选择了New的时候，框架调用这个虚拟函数。这是一个设置文档数据成员的初始值的好地方。

3.3 文档／视图结构中的应用程序类

应用程序类负责唯一的全局应用程序对象的创建、初始化、运行和退出清理过程，对于文档/视图结构，要在应用程序类的InitInstance（）函数中创建一个文档模板，来管理文档/视图结构涉及的框架窗口、文档和视图。

文档模板负责在运行时创建（动态创建）文档、视图和框架窗口。一个应用程序对象可以管理一个或多个文档模板，每个文档模板用于动态创建和管理一个或多个同类型的文档（这取决于应用程序是SDI程序还是MDI程序）。MFC的文档模板类CDocTemplate用于支持文档模板操作。对于单文档界面程序，应使用CSingleDocTemplate（单文档模板类），对于一个多文档界面程序，使用CMultipleDocTemplate（多文档模板类）。

文档模板定义了文档、视图和框架窗口这三个类的关系。通过文档模板，可以知道在创建或打开一个文档时，需要用什么样的视图和框架窗口显示它。这是因为文档模板保存了文档及其对应视图和框架窗口的CRuntimeClass对象的指针。此外，文档模板还保存了所支持的全部文档类的信息，包括这些文档的文件扩展名信息、文档在框架窗口中的名字、代表文档的图标等信息。

在创建了文档模板之后，InitInstance（）函数调用AddDocTemplate（）函数将创建好的文档模板加入到应用程序的可用文档模板链表中去。这样，如果用户选择了File/New或File/Open菜单选项要求创建或

打开一个文档时，应用程序类的OnNewDocument ( ) 成员函数和OnOpenDocument ( ) 成员函数，就可以从文档模板链表中检索出相应的文档模板提示用户选择适当的文档类型并创建文档及其相关的视图和框架窗口。

4 文档 / 视图结构的应用程序实例

我们创建这样的一个程序：当程序运行时，屏幕上显示一个红色滚动的小球，当单击鼠标左键时，可以选择小球的显示颜色；当单击鼠标右键时，可以选择小球内部的填充方案。同时在“文件”菜单上单击“保存”命令，可以把当前小球的颜色和图案保存起来，因此下一次在“文件”菜单上单击“打开”命令时，可以把以前保存的小球的颜色和图案调出来显示。

利用AppWizard生成一个单文档的应用程序Bounce,在向导的生成步骤中都选缺省的。

4.1 小球的数据结构

```
CPoint m_BallCenter;//球的中心位置
CSize m_BallTotalSize;//球的位图的总大小
CBitmap m_BallBmp; //球的位图
CSize m_BallRadius;//球的半径
CSize m_BallSpeed;  //球的运动速度
COLORREF m_BallColor; //球的颜色
int m_BallStyle; //球的内部填充类型
```

4.2 小球数据的永久化存储（文档类中实现）

```
void CBounceDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {//将改变了的球的颜色和填充图案保存
      ar<<m_BallColor<<m_BallStyle;        }
    else
    {//新打开一个文件用文档类的球的颜色和填充图案保存
       ar>>m_BallColor>>m_BallStyle;       }
}
```

4.3 小球的显示（视图类中实现）

```
void CBounceView::OnDraw(CDC* pDC)
{
     CBounceDoc* pDoc = GetDocument();
     CSize radius, move, total;        CBitmap* pbmBall;
    pbmBall = &(pDoc->m_BallBmp); //取得文档类中的球的位图
      radius = pDoc->m_BallRadius;  //取得文档类中的球的半径
      move = pDoc->m_BallSpeed;
      total.cx = (radius.cx + abs(move.cx)) << 1;
      pDoc->m_BallTotalSize = total;
      CClientDC dc(this);     CDC dcMem;
      dcMem.CreateCompatibleDC(&dc);
      pbmBall->CreateCompatibleBitmap(&dc, total.cx, total.cy);
      CBitmap* pOldBitmap = dcMem.SelectObject(pbmBall);
      CRect rect(0, 0, total.cx, total.cy);
      CBrush brBackground(::GetSysColor(COLOR_WINDOW));
      dcMem.FillRect(rect, &brBackground);  CBrush brCross
(m_BrushStyle, 0L);
      CBrush* pOldBrush = dcMem.SelectObject(&brCross);
      dcMem.SetBkColor(m_cBallColor);
      dcMem.Ellipse(abs(move.cx), abs(move.cy),
        total.cx - abs(move.cx),
        total.cy - abs(move.cy));
      }
```

4.4 程序运行界面

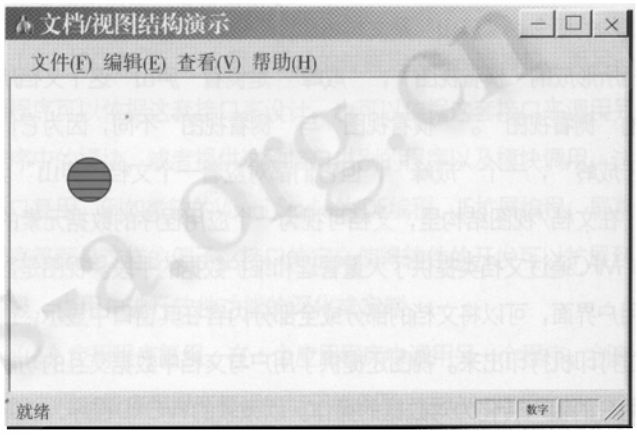


图 1　文档 / 视图结构的应用程序运行界面

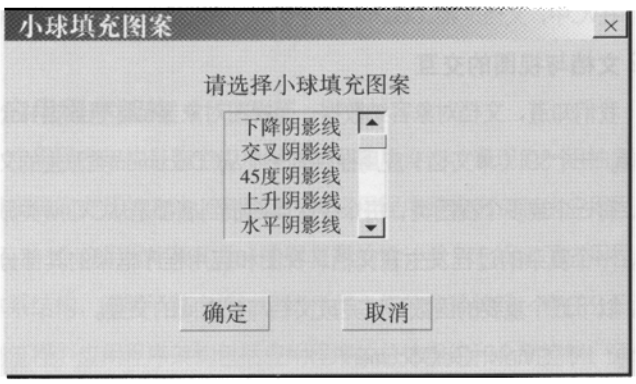


图 2　单击鼠标右键后的运行结果

下转第 39 页 >>

上接第 32 页

5 结束语

通过上述例子可以看出，在基于文档/视图结构的应用程序架构中，将对数据的操作与数据显示界面分离，放在不同类的对象中处理。这种思想使得程序模块的划分更加合理。文档对象只负责数据的管理，不涉及用户界面；视图对象只负责数据输出和与用户的交互，可以不考虑数据的具体组织结构的细节。另外，在文档/视图结构中提供了许多标准的操作界面，包括新建文件、打开文件、保存文件、文档打印等，大大减轻了程序员的工作量。程序员不必再书写这些标准处理的代码，从而可以把更多的精力放到完成应用程序特定功能的代码上，实现了较大粒度的软件复用。

参考文献

1 范辉、刘惊雷，Visual C++6.0 程序设计简明教程，高等教育出版社，2001.7。

2 张世琨、王立福、杨芙清，基于层次消息总线的软件体系结构风格，中国科学（E 辑），2002,32(6):393~400。

3 刘路放，Visual c++ 与面向对象程序设计教程，高等教育出版社，2000.7。