# 基于 MVC 的软件界面体系结构研究与实现*

王映辉[1]，王英杰[2]，王彦君[3]，樊宏斌[4]

(1. 陕西师范大学 计算机学院，陕西 西安 710062；2. 中国科学院 地理科学与资源研究所，北京 100101；3. 中国石油测井有限公司 长庆事业部，陕西 高陵 710000；4. 西北大学 计算机系，陕西 西安 710069)

**摘 要**：软件体系结构研究是目前软件工程领域研究的新一轮热潮，是对软件的更高层次抽象。在分析 MVC (Model/View/Controller) 模式机理的基础上，给出了基于 MVC 的软件界面体系结构及其工作机理。软件界面体系结构带来了软件设计的灵活性和高度重用性。最后，以软件界面体系结构为指导，分析了 MFC( Microsoft Fundament Class) 的文档视图结构，并给出了软件界面体系结构的具体设计和实现。

**关键词**：软件体系结构；模型/视图/控制器模式；文档；视图

**中图法分类号**：TP311.13 **文献标识码**：A **文章编号**：1001-3695(2004)09-0188-03

# Study on Architecture of Software Interface Based on MVC

WANG Ying-hui[1], WANG Ying-jie[2], WANG Yan-jun[3], FAN Hong-bin[4]

(1. *College of Computer Science, Shanxi Normal University, Xi'an Shanxi* 710062, *China*; 2. *Institute of Geosciences & Resource, Chinese Academy of Science, Beijing* 100101, *China*; 3. *Changqing Department, China Petroleum Logging Co., Ltd, Gaoling Shanxi* 710000, *China*; 4. *Dept. of Computer Science, Northwest University, Xi'an Shanxi* 710069, *China*)

**Abstract:** Study on software architecture is a new upsurge in software engineering field. Software architecture is a higher level abstract of software. Analysis theory of MVC, then get the architecture of software interface and make clear the theory of architecture of software interface. Architecture of software interface bring the flexibility and reusability of software design. At last, analyze the structure of document/view of MFC, then demonstrate implementation of MVC-oriented architecture of software interface based MFC.

**Key words:** Software Architecture( SA); Model/View/Controller( MVC) Pattern; Document; View

## 1 引言

软件体系结构( Software Architecture, SA) 是目前软件工程领域的一个研究热点[1~3,10,11]。实践证明，一个成功的软件系统往往都依托着一个好的软件体系结构。由于软件体系结构对软件生命周期有重要的影响，并随着软件规模的不断扩大和软件复杂度的不断增加，以及软件开发费用的不断提高，特别是大规模软件构架技术的出现和应用[4]，软件体系结构已经成为软件开发能否成功的决定因素之一，也是实现软件复用的全新的重要技术手段。

综观软件体系结构的发展，大概经历了如下几个阶段：以汇编语言为代表的“无体系结构”阶段；以流程图为代表的软件体系结构的“萌芽”阶段；以 UML 为代表的软件体系结构的不同模型的描述阶段；以描述系统高层抽象结构为特征而不关心软件建模细节的高级阶段[1]。目前，由于对软件体系结构概念、描述还没有统一的规范，所以软件体系结构还不能在软件的工程化生产中发挥重要的作用[11]。

即便如此，人们还是针对不同的应用领域，从不同的侧面对软件体系结构的理论模型、软件体系结构的描述方法、软件体系结构的设计分析验证、软件体系结构的演化与复用和基于软件体系结构的软件开发等方面进行着不断的研究。其中基于模式的软件体系结构的研究正是从不同的侧面对软件体系结构进行有意义的探索[5,6]。

在软件设计中，软件界面是体现软件性能的一个重要方面，采用丰富的视图对同一模型进行表示和再现，可以大大提高软件的易用性和可操作性。本文正是对软件界面在体系结构方面进行探索和研究。

## 2 基于MVC 的软件界面体系结构

用户界面，特别是图形用户界面，承担着向用户显示问题模型和与用户进行操作和交互的作用。用户不仅希望交互操作的界面保持相对稳定，但更希望根据需要改变和调整显示的内容和形式。也就是说，如何在不改变软件的功能和模型的情况下，方便地完成对软件界面构造的调整。与软件处理问题的内在模型相比较，用户界面是经常要发生变化的，这也正是软件体系结构在软件界面方面的研究任务。MVC 的提出，正好将软件界面的构成独立于它的计算模型。因为 MVC 是对软件界面在体系结构方面的一种抽象，所以在面向对象的软件设计和体系结构的架构中有着重要的指导作用。

### 2.1 MVC 的组成描述

实践证明，问题的计算模型和显示形式是可以独立的[5]，如果将两者紧密地交织在一起，即使对于结构简单的界面，当

收稿日期：2003-09-08；修返日期：2003-10-28

基金项目：国家“863”计划资助项目(2003AA118105)

有各种灵活的需求时, 界面的设计也将会变成一个复杂的过程。针对界面变化的需求, MVC 的解决办法是将系统划分成模型、视图和控制器三部分。也就是说, MVC 是用三元组来构建用户界面的。

模型 Model 是软件独立于外部表现的内在抽象, 是包含了核心数据和逻辑功能计算的应用, 它独立于界面的表达。视图 View 是模型的外在表现, 它从模型中获得信息, 并在屏幕上加以显示。控制器 Controller 定义用户界面对用户输入的响应方式, 它的职责是对模型中任何变化的传播加以控制, 确保用户界面和模型之间的对应关系, 协调着模型和视图的工作。模型和视图的分离使得一个模型可以对应多个视图。模型、视图和控制器之间的关系如图 1 所示。

用户输入表示用户与用户界面之间的交互关系, 通过控制器, 把用户输入与用户界面连接起来。视图通过控制器来响应用户输入。视图与模型之间是相互协作的关系, 即一个视图得到用户输入, 以此来改变模型状态; 同时, 模型又将自己的改变状态通知给它的所有视图, 而视图在得知这些变化后, 自己作相应的改变, 将模型的改变及时展现给不同的用户。这样做的好处是将用户输入、用户界面的显示与模型分离开来, 减小了它们之间的耦合度。MVC 将模型与视图的分离, 不仅提高了系统的灵活性和复用性, 而且为用户软件界面体系结构的研究奠定了基础。

### 2.2　基于 MVC 的软件界面体系结构及其工作机理

从表面上看, MVC 只是将模型和视图进行了分离, 然而这种分离却解决了用户的软件界面体系结构的建立问题。基于 MVC 分离机制的软件界面体系结构可分为两种: ①将三组分分离的体系结构; ②基于视图和控制器结合后的与模型分离的体系结构。

基于三组分分离的软件界面体系结构如图 2 所示。视图和控制器具有同一个父类, 在此父类中定义了一个更新接口 Update( ), 并且具有向模型进行注册和撤销注册的功能。

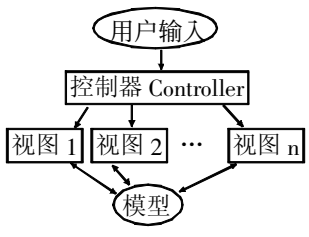

图 1　MVC 组成关系图

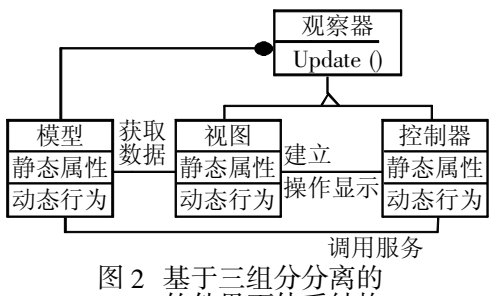

图 2　基于三组分分离的软件界面体系结构

一个模型可以拥有任意数目的视图, 一旦模型的状态发生变化, 所有注册过的视图都得到通知。作为对这个通知的响应, 每个视图都要查询目标状态, 并且保持其状态与目标的同步和一致。由此可知, MVC 模式的优点是模型与视图间是抽象耦合的, 模型所知道的仅仅是它有一系列视图类, 每一个都有抽象的视图接口, 模型不需要任何一个观察者属于哪一个具体的视图类, 这样模型与视图之间的耦合是抽象的和最小的。模型的变化——广播机制不需要指定它的接收者, 不关心有多少对象对它感兴趣, 它的唯一责任就是通知它的各个视图。控制器以事件触发机制接收用户的输入, 为输入的事件提供了相应的操作服务, 并把事件转换成对模型或相关视图的激发操作。

另一种基于 MVC 的软件界面体系结构如图 3 所示。在这种体系结构中, 首先将视图与控制器结合后形成新的视图, 然后再与模型进行分离。将视图与控制器结合形成新的视图可以满足许多应用的需要, 使视图具有事件激发机制, 即按照事件激发机制来接收用户的输入。这可增加视图工作的独立性及其与不同模型之间的适应性。

为每一个视图提供一个内部的模型, 模型的数据可能与视图的数据结构不同, 此时提供一个转换方法( Method) , 在模型的数据发生变化时, 经过转换后发送给视图。视图接收到新的数据后, 完成对视图的刷新。

## 3　基于软件界面体系结构的实现

以上阐明了基于 MVC 的软件界面体系结构的组成和工作机理, 下面给出在此体系结构指导下的具体实现和应用。

### 3.1　MFC 中的文档—视图结构及其关系

文档 - 视图结构是 MFC2. 0 以后提出的一个新结构[7,8], 它是在 MVC 支撑下软件界面体系结构的一个具体应用模型。MFC 的文档—视图结构把应用程序的数据与显示分离开, 使之协调工作, 给应用程序的开发提供了非常灵巧的接口和结构。文档用来管理和保存数据, 而视图用来显示数据, 它们之间如何各自独立、又相互协调的工作, 首先看一下它们在 MFC 中的类层次关系( 图 4) 。

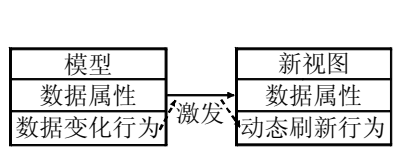

图 3　新的视图与模型分离的体系结构

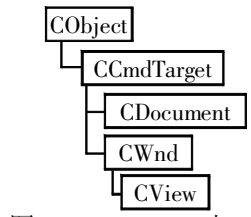

图 4　CDocument 与 CView 的继承关系图

从图 4 可看出, 文档和视图都继承了类 CObject 和 CCmdTarget。CObject 类是 MFC 的基类, 它提供了许多基本的功能, CDocument 继承它的虚函数 Serialise( ) 函数( 序列化函数) , 可自动实现保存数据文件和读取数据文件的功能。而 CDocument 与 CView 继承了 CCmdTarget 类中具有发送和处理消息的功能, 因为 CCmdTarget 是负责消息处理的基类。CView 通过函数 GetDocument( ) 来获得文档的数据, 从而实现对数据的操作。MFC 在实现文档与视图之间的通信时主要用了虚函数。在视图类的基类 CView 上有虚函数 OnUpdate 用来向文档类通知数据发生的变化, 而视图类的基类中的 UpdateAllViews( ) 用来向所有的视图类通知数据的显示要更新了。通过重载这两个虚函数来实现文档与视图之间的通信。

在 MFC 中文档是用来管理数据的, 主要用于数据的存取及其在应用中的交换; 视图是用来对数据进行实现中的用户界面。一个文档可以对应多个视图, 但一个视图只能对应一个文档。MFC 的文档—视图结构可分为单文档多视图和多文档多视图两种结构。单文档多视图就是对应于相同的数据, 让用户用多种方式来观察它。多文档多视图就是相对于不同的数据, 让用户以不同的方式来观察它。文档视图互不影响, 各自管理自己, 来共同实现应用程序的功能。这完全遵循基于 MVC 支持的软件界面体系结构的组成机制和工作机理。

### 3.2　单文档多视图的实现

在利用工具 Visual C ++ 实现基于 MFC 的软件界面开发

中, 单文档多视图的实现有以下两种方法。

(1) 切分静态视图(切分窗口)

在 MFC 类库中有一个从 CWnd 派生来的名为 CSplitterWnd 的子类, 该子类的主要功能就是对窗口的客户区进行分配和管理, 由它将客户区切分为多个视图(切分窗口)。其基本思想是将一个框架窗口分为几个子窗区域, 每个子窗区域代表一个视图。这些视图是并行的, 即属性相同, 每个子窗口的内容由对应的视图来管理, 而所有的视图各自又与文档相关联。这样, 就实现了多视图对应同一个文档的操作。具体实现如下:

①用 AppWizard 生成一个单文档的应用, 应用名假设为 Test1;

②在 CMainFrame 中增加一个类 CSplitterWnd 的成员变量;

```
class CMainFrame: : public CFrameWnd{
…
public:
CSplitterWnd m_Splitter;
…
} ;
```

③重载 CFrameWnd 的切分窗口函数 OnCreateClient( ), 在此函数中调用 CSplitterWnd 的成员函数创建多视图, 实现代码如下(以下代码是实现切分成两个视图, 切分成多视图的方法类似):

```
BOOL CMainFrame: : OnCreateClient( LPCREATESTRUCT lpcs,
CCreateContext * pCo ntext) {
BOOL bResult;
bResult = m_Splitter. CreateStatic( this, 1, 2, Csize( 5, 5) , pContext) ;
If( bResult) {
bResult = m_ Splitter. CreateView( 0, 0, RUNTIME _ CLASS( CTest-
View) , CSize( 200, 200) , pContext) ;
bResult = m_ Splitter. CreateView ( 0, 1, RUNTIME _ CLASS( CMy-
View) , CSize( 200, 200) , pContext) ;
}
return bResult ;
}
```

这样, 我们创建了一个具有两个子窗口(视窗) 的单文档应用, 两个视口之间被分隔栏分开, 当鼠标落在分隔栏上时, 可以用鼠标来重新分配两个视窗的大小。当我们不想让鼠标具有此功能时, 可从 CSplitterWnd 派生出一个新类 CLockSplitterWnd, 同时重载它的 OnMouseMove( ), OnLButtonDown( ) 等函数, 由此可实现我们需要的各种风格的控制。

(2) 基于文档模板的实现

在 Visual C++中, 文档模板的作用是将各种对象组织在一起, 以协调它们的工作。对一个应用程序而言, 它只负责管理文档模板即可。其结构如图 5 所示。

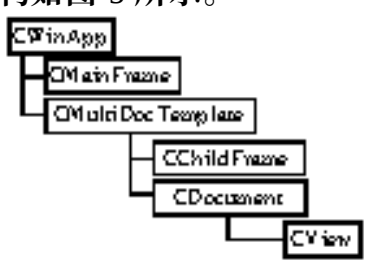


图 5 文档模板的结构图

基于文档模板的单文档多视图的实现过程如下:

①用 AppWizard 生成一个单文档的应用, 应用名为 Test2;

②在 CMainFrame 中增加两个类 CMultiDocTemplate 的成员变量* m_pDocTemplate1 和 * m_pDocTemplate2;

③重载 CWinApp 中的函数 ExitInstance( )

```
CStudyApp: : ExitInstance( ) {
delete m_pDocTemplate2;
}
```

④在 CStudyApp 中的 Instance( ) 函数中添加如下代码:

```
CStudyApp: : Instance( ) {
. . . ;
m_ pDocTemplate1 = new CmultiDocTemplate ( IDR_ MAINFRAME,
RUNTIME_ CLASS ( CStudyDoc ) , RUNTIME _ CLASS ( CChildFrame ) ,
RUNTIME_CLASS( CStudyView) ) ;
AddDocTemplate( m_pDocTemplate1) ;
m_pDocTemplate2 = new CmultiDocTemplate ( IDR_MAINFRAME,
RUNTIME_ CLASS ( CStudyDoc ) , RUNTIME _ CLASS ( CChildFrame ) ,
RUNTIME_CLASS( CMyView) ) ;
… ;
} ;
```

在以上代码中, 因为 m_pDocTemplate1 被 AddDocTemplate( m_pDocTemplate1) 所使用, 也就是说它的删除是由文档模板来管理, 而 m_pDocTemplate2 则不然, 必须通过 ExitInstance( ) 来删除。这样, 就建立了单文档 CStudyDoc 对应多个视图 CStudyView 和 CMyView 的应用。在此, 不同的视图实现了不同的功能, 但它们对应的数据是一致的。

### 3.3 多文档多视图

类似的多文档多视图应用只需将单文档模板中文档类改为不同即可。也就是说, 用相应的视图及其对应的文档来初始化文档模板, 同时注意要将每个文档模板都进行 AddDocTemplate( ) 操作。这样, MFC 就建立起它们之间的关系, 并用来控制多文档多视图中的变化。

## 4 结束语

近几年来, 软件从基于构件( Component) 的开发迈向了基于软件框架( Framework) 和体系结构的开发道路[6,10,11], 特别是对软件体系结构的研究已成为软件工程中实现软件复用的又一轮新的技术研究热潮, 对提高软件生产效率和软件产品质量有着重要的意义。目前虽然在软件体系结构的概念和理论方面还没有达成一定的共识和规范, 但这并没有阻止基于软件体系结构的开发和应用的研究, 反而, 基于不同领域( 领域工程) 和不同侧面的软件体系结构的研究如火如荼。而人们对软件体系结构研究的最终目的是为了获得所需的软件产品(软件的实现)。

在本文中, 分析 MVC 体系结构模式, 阐述了基于 MVC 模式的软件用户界面体系结构和这种体系结构的工作机理。在 MVC 体系结构的指导下, 结合 MFC 给出了软件界面体系结构的实现。可以看出, 基于软件界面体系结构模式的软件界面开发, 由于它的灵活性、可复用性和易扩展性等特点, 对用户界面管理和应用程序框架的构造有着重要的指导作用。

参考文献:

[1] 孙昌爱, 金茂忠, 刘超. 软件体系结构研究综述[J]. 软件学报, 2002, 13(7): 1228-1237.

[2] 赵会群, 孙晶, 王国仁, 等. 软件体系结构性能评价研究[J]. 计算机科学, 2003, 30(2): 144-146.

[3] 万建成, 卢雷. 软件体系结构的原理、组成与应用[M]. 北京: 科学出版社, 2002.

(下转第 193 页)

这其实也就是查询操作( 读操作) 的处理过程。如果是添加、修改等写操作, 则处理过程相反, 即前台按照通信协议对用户的输入内容进行打包; 然后发送给后台, 后台把接收到的字符串进行解析, 调用底层的 Daemon 执行相应的操作。

图 3　文件系统管理的页面

为了增强程序的可读性和扩展性, 验证用户输入、打包、解包等代码都封装成类, 页面中只嵌入对于这些类或者类对象的方法调用。因为这些类都已提前编译, 这样就提高了生成网页的速度( 安装和监控部分也是这样的) 。

## 6　监控部分前台的设计与实现

监控也分为功能池级的监控和节点级的监控。通过 Web 页面显示 CPU, Memory, Swap 等各项性能信息和它们的性能随时间变化的曲线图。因为每过一段时间就要查询一下系统性能信息, 并在 Applet 中重新画图, 所以与安装部分一样要使用 Applet。监控页面如图 4 所示。

图 4　监控页面

按照监控目标的不同, 我们采取了不同的图形。比如 CPU 的利用率使用折线图, 磁盘利用率使用饼状图。对于功能池级的管理, 与管理部分一样可以让用户从该功能池下的节点列表中任意选择若干节点, 每个节点的性能信息占 Applet 的一个区域, 这样用户可以看到不同节点之间的差异情况。　　通信和解包的思想与管理部分大同小异, 此处不再赘述。

## 7　结论

结合目前集群的实际应用需求和用户的操作习惯, 灵活运用多种技术开发的远程集群管理软件的前台, 在实际的应用中, 可以使集群系统的安装、管理、监控显得很方便、容易。当然很多地方还有待改进, 比如前台的监控图还可以更灵活些, 让用户选择他喜欢的图形( 比如说是折线图还是柱状图) , 然后按照用户选择的图形方式进行显示。相信本文中所述的对于集群管理软件前台的设计思想和所用的技术, 对其他的集群管理软件前台的研究和开发来说会有抛砖引玉的作用。

**参考文献**:


[1] 刘广红, 董小社, 吴维刚, 等. 基于 Web 的远程集群系统管理设计与实现[J]. 计算机应用研究, 2003, 20(5): 123-125.

[2] [美] Alex Vrenios. Linux 集群体系结构[M]. 马朝晖, 等. 北京: 机械工业出版社, 2003. 1-97.

[3] Saab C B, Bonnaire X. A Flexible Monitoring Platform to Build Cluster Management Services[C]. Cluster Computing Proceedings. IEEE International Conference, 2000. 258-265.

[4] De Mello R F, Paiva M S V. A Java Cluster Management Service [C]. Network Computing and Applications NCA 2001. IEEE International Symposium, 2001. 238-241.

[5] Collins D E, George A D. Achieving Scalable Cluster System Analysis and Management with a Gossip-based Network Service[C]. Local Computer Networks. Proceedings. LCN 2001. 26th Annual IEEE Conference, 2001. 49-58.

[6] Uthayopas P, Paisitbenchapol S. System Management Framework and Tools for Beowulf Cluster[C]. High Performance Computing in the Asia-Pacific Region. Proceedings the Fourth International Conference/ Exhibition, 2000. 935 -940.

[7] Vogels W, Dumitriu D. An Overview of the Galaxy Management Framework for Scalable Enterprise Cluster Computing[C]. Cluster Computing. Proceedings. IEEE International Conference, 2000. 109-118.


**作者简介**:


卫建国, 硕士研究生, 主要研究方向为计算机系统结构; 董渭清, 教授, 主要研究方向为计算机系统结构; 董小社, 博士生导师, 主要研究方向为计算机系统结构; 刘广红, 硕士研究生, 主要研究方向为计算机系统结构; 张露, 高级系统工程师, 主要研究方向为高性能服务器。


( 上接第 190 页)


[4] 王映辉, 冯德民. 大规模软件构架技术[M]. 北京: 科学出版社, 2003.

[5] rich Gamma et al. Design Patterns: Elements of Reusable Object-Oriented Software[M]. Addsion-Wesley, 1999.

[6] 面向模式的软件体系结构[M]. 贲可贵, 等. 北京: 机械工业出版社, 2002.

[7] icrosoft Corporation. MSDN Library Visual Studio 6. 0[M]. Microsoft Press, 1999.

[8] isual C ++ 技术内幕[M]. 潘爱民, 王国印. 北京: 清华大学出版社, 1999.

[9] 邵维忠, 杨芙清. 面向对象的系统分析[M]. 北京: 清华大学出版社, 2000.

[10] 杨芙清, 王千祥, 梅红, 等. 基于复用的软件生产技术[J]. 中国科学( E 辑) , 2001, 31(8): 363-371.

[11] uttorm S. The REBOOT Approach to Software Reuse[J]. System Software, 1995, 30: 201-212.


**作者简介**:


王映辉, 副教授, 博士, 主要研究方向为软件构架技术与可视化技术; 王英杰, 研究员, 主要研究方向为地图可视化。