

面向对象的框架设计^①

熊 江

重庆三峡学院 计算机科学系, 重庆 万州 404000

摘要: 分析与研究了面向对象的框架设计. 通过与组件和设计模式的比较, 针对框架的不足, 提出了改进思路, 以降低其设计的难度.

关键词: 框架; 组件; 复用; 设计模式

中图分类号: TP391

文献标识码: A

1 框架概述

框架是构成一类特定软件可复用设计的一组相互协作的类. 它规定了用户的应用的体系结构, 定义了整体结构、类和对象的分割、各部分的主要责任、类和对象如何协作, 以及控制流程. 因而, 框架更强调设计复用.

从组成来讲, 框架是抽象类和具体类的混合体, 抽象类存在于框架中, 具体类存在于应用程序中. 所以, 框架是一个有待完成的应用程序, 里面包含了特定领域的应用程序的共同方面; 另外, 通过定义一些设计参数, 以用于各个应用程序的特殊细节.

2 框架与组件

基于组件的软件开发是可复用软件开发的一种有效形式. 它使复杂系统简化为一些简单的对象模块, 体现了面向对象的思想, 但是它主要是针对发生频率高的, 比较具体的, 相对独立的问题解决方案. 从对组件的定义中我们了解到它所涉及的模块太小而不能容纳整个结构, 也就是说, 它不能对整个结构进行重用.

框架是可复用软件开发的最普遍的、可修改弹性最大的一种形式. 相对组件来说, 框架能提供一系列问题的解决方案, 因而它更有灵活性和可扩展性. 组件能提供的仅仅是可绑定的一个模块, 框架提供的是一个模板, 能被许多应用程序所采用的一个程序的框架轮廓.

3 框架实例

框架在面向对象的软件工程领域正发挥着日益显著的作用. 只要你是程序员, 你就会和框架有着或多或少的关系. Microsoft Foundation Classes (MFC), Microsoft's, Distributed Common Object Model (DCOM), Java-Soft's, Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA) 都是框架, 而其中的一些框架如 MFC, JAVA 在软件领域发挥着重要作用.

下面对 MFC 这样一个影响深远的 C++ 程序框架进行讨论.

① 收稿日期: 2003-01-17

作者简介: 熊江(1969-), 男, 重庆万州人, 讲师, 主要从事智能分布式数据库, 智能多媒体技术的研究.

3.1 MFC 应用模块与框架模块的切割

MFC 作为一个很复杂的应用程序框架,首先必须内含一个 class 层次结构.这个结构可以是抽象类,也可以是抽象类和具体类的混合体.其次,面向对象中的多态和虚拟技术是不可缺少的,也就是说, class 层次中的某些 classes 必须包含有虚拟函数.

MFC 采用 single(单)继承和 public(公共)继承.采用单继承的好处是:简单易用.采用 public 继承的原因是:这里的每一个上下类别都是 is-a 的关系.

3.2 MFC 类和对象的动态读写

MFC 程序框架以一个 Cobject 为基类扩展开来,应用程序中任何一个 class,只要从这个继承体系衍生开来,就可以具备框架 classes 类层次结构内在的协作关系.这是其一.

其二,应用程序中的 class 必须具备以下三个条件,才能融入 MFC 所构建的框架体系:

- (1) 执行期间的类别的识别(Runtime Type Identification)
- (2) 动态生成(Dynamic Creation)
- (3) 文件读写(Serialization)

每个 application 在执行初期,都必须拥有一整套与其所有 classes 呼应的结构,提供有 classes 名称和相应的 CreateObject() 函数.这个结构称为“类别型录网”.

另外,为了方便用户,MFC 提供三组宏,让应用程序轻松加入适当地点,就可以轻松的建立上面的三个条件.这三组宏的名称如下:

DECLARE_DYNAMIC / IMPLEMENT_DYNAMIC	执行期间的类别的识别
DECLARE_DYNCREATE / IMPLEMENT_DYNCREATE	动态生成
DECLARE_SERIAL / IMPLEMENT_SERIAL	文件读写

3.3 MFC 消息的获取,派送和处理

MFC 应用程序框架属于 Microsoft Visual Studio 系列中的一个产品,通过封装 Windows Api 成为 MFC 内的 classes 类层次结构.但是不管如何封装,都离不开 win32 应用程序特有的运行机制.这个运行机制最大的特点就是 message-base, event-driverd. 由此就引出了一个问题,其维持生命所需的“消息”该如何处理,才能配合程序框架所产生出来的 class 体系结构.

原本以 Windows Api 写成的程序,所产生的每个视窗都必须由应用模块提供一个行为中枢.所谓视窗函数即是,负责接收::DispatchMessage() 派送过去的消息,并拦下某些消息做进一步的处理,不感兴趣的放手让::DefWindowProc() 处理.

引入 MFC 程序框架,其目的是复用性,希望为所有应用程序服务.为此,框架设计者事先定义好数个 window classes,代表数个经常被使用而且行为已完全确定的视窗种类,这些视窗种类中的视窗函数指针一律指向同一个视窗函数:AfxWndProc().

这样一来每一个 CWnd-derived 所产生的视窗的视窗函数就都是 AfxWndProc()了.但是每个 CWnd-derived 对应的视窗必须有机会自行决定处理哪些消息,为达到此目的,只要令消息处理函数在框架结构中是虚拟函数即可.

在程序执行期间,对象需要动态识别,消息也需要动态识别(本来消息就是动态产生的).与对象的动态识别采用的“类别型录网”类似的方法,在 MFC 中,有一张消息映射表,而为了让用户方便,MFC 提供一组宏,可以轻松建立消息映射表.宏如下:

BEGIN_MESSAGE_MAP / END_MESSAGE_MAP	形成消息映射表
-------------------------------------	---------

而为了消息能够让特定的函数去处理,又定义了一个宏:

ON_COMMAND / ON_WM_CREATE / ON_WM_MOUSEMOVE / ...

于是,消息在“消息绕行”的机制下流动到合适的 class,就被相应的处理函数所获取处理.

3.4 MFC 中的 MVC 模型

所谓 MVC(Model/View/Controller) 是三个一组的 classes,用来在 Smalltalk-80 中建立图形使用界面(GUI).这个模型非常有名,被视为一种设计模式,之后也被许多系统(包括 MFC)采用.

MVC 模型在 MFC 中的应用，就是所谓的 Document-View 模型。Document 负责实际资料的存放，View 负责资料的显示。

4 框架与设计模式

设计模式描述了在面向对象软件设计中不断重复发生的问题，以及该问题的解决方案的核心(上面谈到的 MVC 模型就是一个设计模式)。因此，设计模式导致可复用面向对象软件的设计。

一般来讲，设计模式分成两组：一组聚焦于面向对象的模块和程序的设计；另外一组则针对高效、可扩展、并发、并行和分布式的程序设计。

从设计复用的角度讲，框架比设计模式更重要。

首先，框架可直接应用于软件的过程，而模式仅仅是一个向导，一个针对普遍问题的实现方案。虽然设计模式是面向对象软件的设计经验的提炼，但是它带有抽象性，只有依靠实例才能表示。而框架能够用代码表示，它能够使用程序设计语言写出来，它们不仅能被学习，也能被直接执行和复用。

其次，框架比设计模式更加特例化。框架总是针对一个特定的应用领域。而设计模式就不能这样了。

最后，设计模式是比框架更小的体系结构元素。一个典型的框架包括了多个设计模式，而反之决非如此。

框架能使应用程序的开发简单，价格低廉，但是开发框架不是一件容易的事。它是一个需要领域和设计经验的反复过程。为了保证框架的灵活性，必须提取和发现热点。设计模式可以简化这个过程，因为它提供了对过去经验的抽象。应用框架能高度抽象同一领域内的问题，进而降低开发难度和强度。

5 框架的不足及其改进思路

正如大多数软件一样，用户对于程序框架的分析视角往往不同于开发者的想法。用户总是希望程序框架能应用于各种各样的场合，希望基于程序框架所开发的应用程序容易维护，越“傻瓜”越好。对开发者来说，用户的需求是必须要考虑的，除此以外，程序框架本身的发展、进步和维护也是不容忽视的方面。由于这些原因，程序框架的开发变得格外的复杂，这正是框架的不足之处。

为了降低框架的设计难度，特提出以下改进方案：

改进方案一：在框架设计中引入组合体的概念。一个组合体是一组资源和类型，并包括有关这些资源和类型的元数据，也就是被作为一个单元配置的。元数据被称为组合体的名单，它包含象类型和资源表之类能被组合体外看得见的信息。这个名单也包括有关从属关系之类的信息，例如组合体建立时的版本号。开发人员可以指定版本策略，以指示运行语言是否装入系统上已安装的依赖于组合体的最新版本，装入一指定版本，或在编译时使用的版本。

组合体可以被一个应用程序私有，或被多个应用程序共享。一个组合体的多个版本可以同时配置在同一台机器上。应用程序的配置信息定义了应到何处去查找组合体，这样，Runtime 就能将同时运行的两个不同的应用程序装入到同一组合体的不同版本中，消除了由组件版本的不兼容性引起的问题，提高了系统整体的稳定性。如果必要，管理员可以为配置时的组合体增加配置信息。

因为组合体是自描述的，所以并不需要在系统上进行注册。应用程序的配置简单到了只需将文件拷贝到目录中即可(如果为了使应用程序能够运行，必须安装未经组织过的组件的话，情况会稍微复杂一点)。配置信息保存在可被任何文本编辑器编辑的 XML 文件中。

改进方案二：在框架设计中引入智能 Agent 技术。框架的某一个分支会发展成智能 Agent，因为框架设计中引入 Agent 技术有两个优点：第一，Agent 较普通组件具有更高的自主性，能够对周围环境具有一定的赶制性；第二，Agent 可以理解为移动组件，它可以根据需要在网络节点间移动。

改进方案三：在框架设计中引入中间件技术的建模新思想，提出基于总线的系统框架模型。系统中的对象是按照规范设计的模块，这些定义良好的软件模块(组件)在系统中共存，并且充分地相互作用。按照这种结构，可以将若干组件组合起来，以建立更大和更复杂的系统。这种模型的关键在于一种高效的总线结构，使组件之间能以一个公共的接口互相连接，做到组件的即插即用，无缝集成。这种模型的系统，组

件间的通讯链接数是线性的,并且由于各组件接口规范的一致性,通讯的复杂度大大下降,也提高了框架的互操作性.核心组件与核心总线构成了应用软件系统的原型和框架,在此基础上与各应用组件集成.

改进方案四:其一:经常参考一些成功的程序框架来吸取经验,多复用一些成熟的组件,这样就会使后续的开发工作简单;其二:可以多参考一些设计模式,系统设计必须要富于弹性,要能动态增加子系统和子功能,动态增加子类型;其三:在框架的发展过程中,要经常的使用,通过它开发一些应用程序.只有不断的使用,才能够不断的发现问题,从而解决问题,从而使框架日趋完善.

6 总 结

框架变得越来越普遍和重要.它们是面向对象系统获得最大复用的方式.较大的面向对象应用系统常常由多层彼此合作的框架组成.框架的设计实现是非常复杂的一个主题,也是非常富有挑战性的.框架的设计本身也是一个循环迭代的过程,它在不断应用于实践的过程中,得到更新和提高.如果说应用程序难以设计,那么工具箱就更难了,而框架是最难的.框架设计者必须冒险决定一个要适应于该领域的所有应用的体系结构.框架对应用的主要贡献在于它所定义的体系结构.本文针对框架的不足,提出了改进思路,以降低其设计的难度.

参考文献:

- [1] 侯俊捷.深入浅出MFC[M].第二版.武汉:华中科技大学出版社,2001.
- [2] Charles Petzold. Programming Windows[M].北京:北京大学出版社.1999.
- [3] 刘涛. Visual C++ . Net 核心编程与开发实例[M].北京:人民邮电出版社,2001.

The Design of Object-Oriented Framework

XIONG Jiang

Dept. of Computer Science, Chongqing Three Gorges College, Wanzhou Chongqing 404000, China

Abstract: The Object-Oriented Framework design is analyzed. And some improvement ideas on its shortage are suggested to lower the design difficulty.

Key words: framework; component; reusability; design pattern

责任编辑 覃吉康