计算机科学 2006Vol.33№.12(增刊)

设计模式在面向对象的框架中的应用

李卓伟[1] 王成良[2] 周树语[1]

(重庆大学计算机学院 重庆 400044)[1] (重庆大学软件学院 重庆 400044)[2]

摘 要 面向对象框架是面向对象系统获得最大复用的方式,它作为大型的可复用组件在面向对象应用中使用,应用的大部分设计和代码将来自于它或受其影响。设计模式有助于获得无需重新设计就可适用于多种应用的框架体系结构,一个设计良好的框架应得到多种设计模式的支持。本文比较详细地阐述了面向对象框架的特征,以及设计模式对框架的支持,并以某地质仪器生产厂家开发的PS测井分析处理系统为实例,说明了设计模式在使用和扩展面向对象框架中的应用。

关键词 设计模式,框架,模板方法,钩子操作,MVC(Model/View/Controller),文档/视图

1 引言

软件复用是软件工程领域一直以来追求的重大目标,它推动了面向对象技术发展。而面向对象框架作为面向对象系统获得最大复用的方式,近年来已经成为软件工程领域研究的热点和重点。

面向对象的框架是对某领域内应用系统的部分或整体的可重用性设计,是一种由特定应用领域的软件体系结构所决定的软件构架。它给出了一类应用系统在总体构造上的共性或相似性,忽略了各个应用系统之间的局部差异。框架强调的是软件的重用性和系统的扩充性。框架的使用能向应用提供大部分设计和代码,使应用程序设计者能集中精力于程序本身。但设计一个良好的应用框架并不是一件容易的事,它需要丰富的经验和技术。设计模式是这些经验和技术系统地总结后的结晶。设计模式的使用能使框架结构变得更简洁、更易于理解和使用。

设计模式和框架一直是相辅相成的,设计一个良好的框架要运用到大量的设计模式,而在框架的设计过程中也能提取出了许多新的设计模式。许多面向对象的框架已经成功被开发且在领域内得到广泛应用。如应用于系统体系结构的微软的MFC(Microsoft foundation classes)和应用于应用集成中间件的ORB(object request broker)。这些框架的应用大大地缩短了大型应用软件系统的开发周期,提高了开发质量。

2 面向对象的框架与设计模式

2.1 面向对象的框架

框架是一个系统或其一部分的可重用设计,通常用一组抽象类和存在于这些类的实例之间的相互关系来表示[1]。面向对象的框架规定了应用软件的体系结构;定义了类和对象的分割,各部分的主要责任,类和对象协作关系,以及控制流程;记录了其应用领域的公共设计决策。面向对象的框架具有如下特征:

(1)可复用性:框架强调应用领域内重复出现的大粒度问题及其解的抽象提取,适应于该领域内一组相关问题的求解,可作为应用程序的半成品,具有较大的重用粒度。可复用性是框架的最重要的特征,后面的其它属性都是由其衍生或受其影响。

(2)模块化:框架是对某领域内应用软件公用部分的抽象,具体表现为一组相关的抽象类。面向对象的封装性可为应用框架提供统一而稳定的使用界面,从而可使设计与实现的细节局部化并对用户透明,这为框架的使用和实现带来了诸多方便。

(3)集成性:框架的本质内涵为对一组问题求解的基础结构。通常表现为在统一基础结构下的一组设计模式(design pattern)与相应功能的有机合成。

(4)可扩展性:框架为用户提供了直接使用其内置功能和修改及扩充其功能的途径。白箱框架(whitebox frameworks)主要依靠面向对象的继承和动态绑定机制实现其可扩充性。用户可以直接使用框架提供的功能函数,也可以继承框架的特征并重载框架中预定义的hook方法,实现其可扩充性。由于白箱框架比较成熟,且更符合面向对象的概念,我们在后面还将重点讨论。黑箱框架(blackbox frameworks)使用对象组合和代理方法,通过定义组件界面使其可插入到框架中去的机制实现其可扩充性。用户可以定义特定界面的组件,并将其插入到框架中。

应用框架本身作为可重用的软件构件,就应具有重用性、抽象性和易扩展性等特点。它是面向对象应用框架方法学的主要成果。

2.2 设计模式与面向对象框架的关系

设计模式指的是针对某个领域中的公共设计问

题，对一些行之有效的设计结构和设计经验的系统地抽象、命名与定义。Christopher Alexander 对设计模式(design pattern)有精辟的描述：模式“描述了某种环境中反复出现的问题以及该问题的求解方案，它可以被反复地使用而不必从头作起”[2]。因此，设计模式的研究目标是显式地复用那些成熟的设计技巧与设计经验，提高软件整体设计的灵活性、可复用性。

面向对象的设计模式是基于面向对象的概念的，面向对象的特征如封装，继承，多态等是面向对象设计模式的基础和出发点。每一个设计模式都集中于一个特定的面向对象设计问题或设计要点，描述了什么时候使用它，以及使用的效果和如何取舍。

设计模式的使用有助于获得适用性更强的框架体系结构，一个使用设计模式的框架比不用设计模式的框架更可能获得高层次的设计复用和代码复用。成熟的框架通常都会用到多种设计模式。它们的主要不同在于：

(1)设计模式高于面向对象框架的抽象。设计模式是对领域内可复用的设计结构的抽象描述，应用框架则是对这些抽象描述用具体程序设计语言的实现。从这个意义来说，框架是个物理实体，而设计模式是个逻辑实体。因此，设计模式的抽象级别在应用框架之上。

(2)设计模式是比框架更小的体系结构元素，典型的框架都包含多个设计模式，而反之决非如此。

(3)因为设计模式是高于框架的抽象，所以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，而设计模式可以被用于任何应用。框架的威力在于它们能够用程序设计语言编写，不仅可以被学习，还能被直接执行和复用。

3 白箱框架设计核心—模板方法模式

白箱框架充分体现了面向对象的特征，最为符合面向对象的概念。模板方法模式是面向对象的白箱框架中运用最多的设计模式。是白箱框架中为用户提供的直接使用性和扩展性的基础。因此，可以毫不夸张地说，它是白箱框架设计的核心。

3.1 主要思想

通过在抽象父类中定义一个方法(模板方法)，用来定义一个算法的基本框架，该模板方法调用其他方法，其中一些方法在子类中被重新定义以实现子类自己独特的行为特征。这种方法使子类可以不改变一个算法的总体结构即可重定义该算法的某些特定步骤。如图 1。

3.2 模板方法的特点

(1)反向控制：模板方法导致了一种反向的控制结构，这里指的是一个父类调用一个子类的操作，即子类从父类中获得被调用的消息，而不是相反。

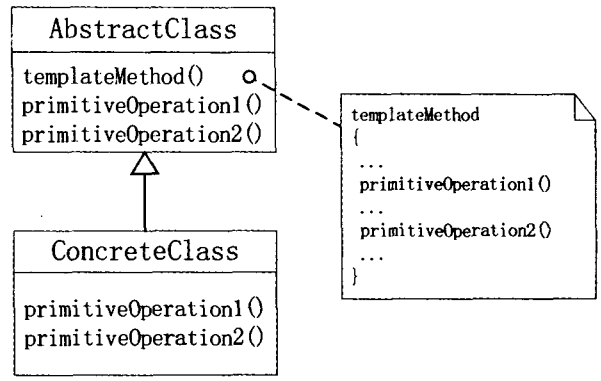

图 1 模板方法模式

(2)控制子类扩展：模板方法只在特定点调用钩子操作，这样就只允许子类在这些点进行扩展。

(3)在扩展上面也提供了一定灵活性：虽然模板方法只允许在特定的点进行扩展，但由于子类对父类的继承关系形成了一个树状结构，所以子类可以在框架范围内在树状结构的任意一级上进行扩展。

(4)钩子操作(hook operations)：模板方法模式提供两种子类可重定义的操作：抽象操作和钩子操作。抽象操作是必须被重定义的操作，而钩子操作提供了缺省的行为，子类在必要时可进行扩展。一个钩子操作的缺省操作通常是一个空操作。钩子操作比抽象操作更灵活和安全。

3.3 模板方法和框架的对应关系

框架分为公共部分和可变部分，公共部分是一种高层次的封装，体现了特定领域公共的设计决策和解决方案。而可变部分留给使用者自行定义和扩展。模板方法中将算法的骨架集中于公共的父类而将算法可变部分延迟到子类去实现和扩展的特征可用于框架的实现。

框架是一种运行时体系结构，这种结构按称之为“反向控制”。当事件发生时，框架的派发器(Dispatcher)调用事件处理对象(用户定义的类)的钩子方法，这意味着，用户定义的类将会从预定义框架类获得消息，反向控制允许框架而不是应用程序决定何种方法被激活来响应外部事件。模板方法模式中也有“反向控制”的概念，由父类决定调用某个子类的操作。实际上它们是相同的概念。

框架要求提供一定的灵活性，而模板方法模式充分利用了面向对象的继承特征，为框架的扩展提供了一定的灵活性。

钩子操作为框架提供良好的扩展性，模板方法模式的核心就是它提供钩子方法，以利于使用和扩展。

4 实例

4.1 系统简介

PS测井分析处理软件是为某地质仪器生产厂家开发的浅层地震综合测量与处理软件系统(SSMMP 2002)系列项目的一个子项目。它的功能包括地质测量信息采集,对采集信息的分析处理,处理结果的保存、显示及打印。这是一个典型的输入—处理—输入过程,对应于用户图形界面的控制器—模型—视图(MVC)。MVC是用户图形界面的经典参照结构,通过将领域模型和视图分离以提高灵活性和复用性,这一结构的相互之间的协作关系主要由观察者模式(Observer)、组成模式(Composite)、策略模式(Strategy)给出。MFC文档/视图框架实现和具体化了这一结构,体现了设计模式在框架中的应用。因此,我们选用MFC文档/视图框架作为本应用的基本框架,以文档类为数据的变换控制中心,并根据应用的实际情况对视图类进行扩展。系统需绘制出五个视图,分别为S波波场图视图(S-WaveCurveGraphView)、P波波场图视图(P-WaveCurveGraphView)、直方图视图(RectangleGraphView)、数据表视图(DataChartView)、时深曲线图视图(TimeDepthGraphView)。实际效果图如下(为便于编排,只给出S波波场图视图、P波波场图视图、直方图视图、数据表视图):

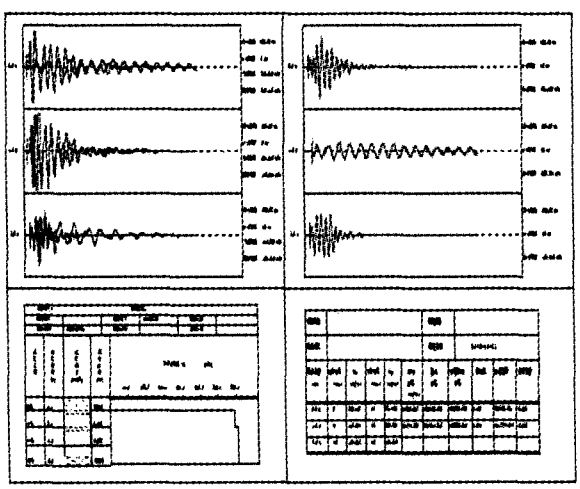

图2 PS测井分析处理软件实效图

4.2 文档/视图框架

MFC文档/视图的核心是文档对象以及相关的视图窗口的概念。文档对象维护应用性数据和功能(领域模型),而视图负责数据的显示和接受用户的交互,文档和视图是一对多的关系。文档、视图和应用框架之间包含了一系列复杂的相互作用过程,有关的书籍和文献中都有详细的介绍。图3给出MFC文档/视图框架中各个基类的相互协作关系图。

文档,视图,框架窗口都是类层次结构。它们中都包含了一定数量的钩子方法,使用者可以通过加入子类在这些钩子方法的扩展点进行扩展。模板方法模式在这里得到了体现。

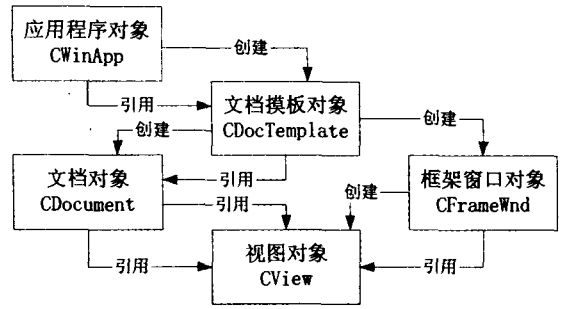

图3 文档/视图框架基类的相互协作关系模型

4.3 文档/视图框架的扩展和使用

在PS测井分析处理系统中,S波波场图和P波波场图的表示形式极其相似。因此,我们扩展一个波场图视图类以封装S波和P波波场图显示算法的基本骨架,且将可变部分定义成钩子方法,留到具体的S波波场图视图子类和P波波场图视图子类中实现。其它视图直接继承框架中的视图基类并实现了具体的视图显示方法。如图4所示。

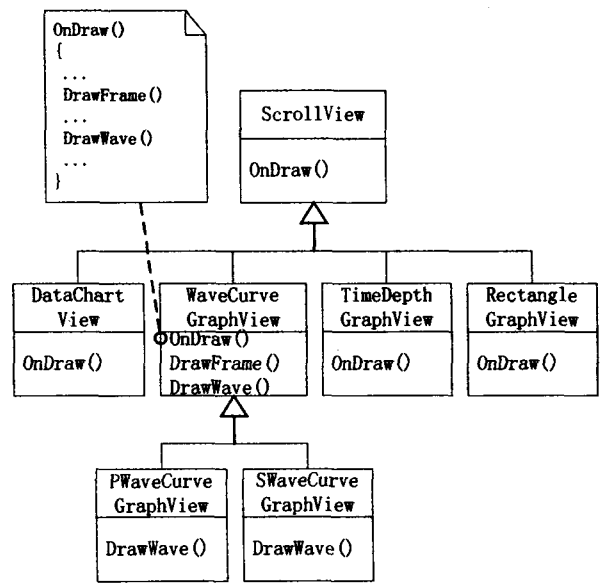

图4 PS测井分析处理系统视图类层次结构

4.4 文档/视图框架对扩展视图的调用

在对视图进行扩展之后,需要做的是把用户的切换视图消息发送到相应的派生视图类中,编写了一个小的分派器SwithToView(),能在框架自身事件分派机制的基础上将文档绑定到相应的视图。将分派器方法放在框架窗口类中较能体现M/V/C中控制器的思想。伪代码示例程序如下:

```
private void FrameWnd::SwitchToView(int MacroView)
{
  View OldActiveView = GetActiveView();
  OldActiveView. HideWindow();
  switch (MacroView) {
    case PWAVE
      PwaveCurveGraphView NewActiveView
```

(下转第221页)

遗传算法(Genetic Algorithm,GA)是近年来迅速发展起来的一种全新的随机搜索与优化算法,其基本思想是基于 Darwin 的进化论和 Mendel 的遗传学说。该算法由密执安大学教授 Hol2land 及其学生于 1975 年创建[5]。

遗传算法是一种模拟自然界生物进化过程的计算模型,遗传算法能解决随机算法的盲目随机性,并能从群体中选择更满足条件的个体,具有很强的智能性,同时它能根据不同的环境产生不同的后代,具有动态性,自适应性,从而能满足题库不断变化的要求,由于试题库容量大,覆盖面广,因此选题计算量大,利用遗传算法的内在并行性,可以有效的解决计算量大的问题。

与传统搜索算法不同,遗传算法从一组随机产生的初始解,称为群体,开始搜索过程。群体中的每个个体是问题的一个解,称为染色体。这些染色体在后续迭代中不断进化,称为遗传。遗传算法主要通过交叉、变异、选择运算实现。交叉或变异运算生成下一代染色体,称为后代。染色体的好坏用适应度来衡量。根据适应度的大小从上一代和后代中选择一定数量的个体,作为下一代群体,再继续进化,这样经过若干代之后,算法收敛于最好的染色体,它很可能就是问题的最优解或次优解。

遗传算法的实现[6]如下:

1)生成初始群体;

2)计算适应值;

3)根据适应值,选择染色体进行复制;

4)进行交叉、变异等操作生成新一代群体;

5)计算适应值;

6)若改变交叉、变异的概率条件成立,则改变;

7)如果最好个体满足所有上述约束条件,则输出当前最好个体,否则转向 3)。

遗传算法主要通过交叉算子繁衍后代,当交叉算子所作用的两个个体相同时,不能产生有意义的新的个体,因此要求初始种群体具有广泛多样性。当群体进化到其中的各个个体均相同时,交叉算子无效,此时仅靠变异算子产生新的个体,遗传迭代难以进行下去,即发生所谓"早熟收敛"现象。

结束语 一个自动组卷系统的性能主要取决于组卷算法,一个好的组卷算法既要保证组卷的成功率,又要保证数据运算的时间效率。在传统的组卷算法中(诸如随机抽取法,回溯法),组卷成功率较低,时间和空间开销都比较大,适合于小型题库系统。将遗传算法应用于组卷中,使组卷的成功率和收敛速度都得到明显提高,适合于较大型题库系统。由于求解精度和收敛速度是相互矛盾的,要使组卷的误差精度和收敛速度进一步得到改进,还需要做出更深入的研究。

参 考 文 献

1 毛秉毅. 基于遗传算法的智能组卷系统数据库结构的研究[J]. 计算机工程与应用,2003(6):2311

2 李小勇,刘开生,王瑛. 题库管理系统的设计与实现[A]. 计算机应用研究[C]. 北京:北京工业大学出版社,2000. 245~246

3 李小勇,王瑛. 题库管理系统中的自动化组卷算法[J]. 西北师范大学学报(自然科学版),2002

4 徐娟芬,袁小东. Pascal 题库系统的设计与实现[J]. 计算机应用,1998,18(6):16~19

5 Holland J H. Adaptation in Natural and Artificial Systems. Ann Arbor:University of Michigan press,1975

6 FEN G Xudong, CHEN G Fan. Usage and realization in genetic algorithm program ring [J]. Development in Microcomputers, 1997, 6: 426

(上接第 204 页)

```
      =new PWaveCurveGraphView;
    break;
    case SWAVE
      SwaveCurveGraphView NewActiveView
      =new SWaveCurveGraphView;
    break;
    ......
  }
  SetActiveView(NewActiveView);
  NewActiveView. ShowWindow();
}
```

结束语 软件系统日益复杂,面向对象应用框架作为面向对象系统获得最大复用的方式,变得越来越普遍和重要,因而有着广阔的发展前景。目前,已经有相当数量的成熟框架可供选择。根据特定的面向对象系统的特点,使用已有框架进行系统开发是一种普遍使用和行之有效的开发策略。成熟框架通常使用多种设计模式。了解框架的运行机制,了解这种机制所体现的设计模式的思想,并将设计模式的思想运用到对框架的使用和扩展上,有助于开发出更为健壮和更易扩展的面向对象系统,且有助于归纳特定领域设计经验,提取特定领域的设计模式,在已有框架基础上形成自己的领域框架。

参 考 文 献

1 Johnson R E. Framework=Components+Patterns [J]. Communications of the ACM,1997,40(10):39~42

2 Alexander C, Ishikawa S, Silverstein M. A Pattern Language: Towns, Buildings, Construction [M]. New York: Oxford University Press,1997

3 Fayad M, Schmidt D. Object-oriented Application Framework [J]. Communications of ACM,1997,40(10):32~38

4 Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-Oriented Software [M]. Reading Massachusetts: Addison-Wesley Publishing Company,1995

5 李英军,吕建,王宏琳. 面向对象应用框架在油气勘探领域的应用研究[J]. 软件学报,1999,10(4):349~354

6 何 昭,李传湘,崔巍. 基于面向对象框架的软件开发方法[J]. 计算机工程,2002,28(4):5~7