# 软件设计模式课程实验教学系统探索①

周 宇, 赵洪达, 张倩雯

(南京航空航天大学 计算机科学与技术学院, 南京 211106)

**摘 要**: 设计模式是软件工程专业的重要核心课程, 兼具理论性和实践性, 但其高度的抽象性给学生学习带来较大困难. 针对该课程及学生的认知特点, 设计开发了一个基于 Spring 框架的设计模式实验教学系统-在线教务管理平台, 该系统采用 B/S 架构, 综合了多种常用设计模式, 不仅展示了单个设计模式的特点, 同时展示了这些设计模式在实际项目开发中的复合应用, 有助于学生加深设计模式方法学理论精髓的理解, 提高实践动手能力, 从而为进一步掌握高级软件工程知识奠定基础.

**关键词**: 软件工程; 设计模式; Spring 框架; 教学系统

## Teaching Platform for the Course of Software Design Patterns

ZHOU Yu, ZHAO Hong-Da, ZHANG Qian-Wen

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

**Abstract**: The course of design patterns is an important core course of software engineering. The course is characterized by both theory and applicability. But the abstract feature brings obstacles to students. According to the features of the course and the students' cognition, we design and implement a teaching platform based on Spring Framework-online education management system. The system adopts B/S architecture, and integrates multiple design patterns. It shows not only the features of individual design patterns, but also their composite application in a practical project. Thus it helps the students to have a deeper understanding of the essence theory of design patterns, enhance their programming ability and lay the foundation for them to learn more advanced knowledge in software engineering.

**Key words**: software engineering; design patterns; Spring framework; teaching platform

在软件工程的课程体系中, 设计模式有着承上启下的特殊地位. 程序设计语言和数据结构是学习设计模式的基础, 但是, 和这类课程不同, 设计模式要解决的不是功能性问题, 而是要解决其软件质量相关非功能性的问题(如可维护性等), 这就需要学生在学习时关注从功能性属性到非功能性属性的转变, 这个转变对于设计模式的学习以及软件工程思想的理解都非常关键, 然而在笔者的教学实践中发现, 这个思维转变对于初学者而言有一定困难. 软件工程的所有活动都是围绕提高软件质量、降低软件开发成本而展开, 大量工作都是在研究如何设计维护性良好的解决方案以应对环境或者需求方面引入的变化, 设计模式是软件工程专业课程体系中首次关注这个问题并提出解决方案的课程, 尽管是初级课程, 但为学生进一步学习软件工程中的高级知识如软件体系结构等奠定基础, 同时设计模式在工业界中有着广泛的应用[1]. 因此, 设计模式课程是软件工程专业的特色和核心课程.

由于软件工程专业初学者的思维惯性, 往往认为软件功能性属性重要, 而忽视非功能性属性, 这个软件设计模式教学带来了一定的障碍. 设计模式课程同时具有一定的抽象性和具体性. 从软件开发周期来看, 设计模式两端联系着需求阶段和实现阶段, 是沟通抽象设计和具体实现的桥梁. 设计模式本身是属于方法学的范畴, 是独立于特定的程序设计语言, 它强调的是一种质量高、鲁棒性强、可复用的方案. 而设计模式的落地应用需要在代码中体现, 需要大量依赖对象

① 基金项目:江苏省自然科学基金(BK20151476);中央高校基本科研业务基金(NS2016093)
收稿时间:2016-09-02;收到修改稿时间:2016-09-29 [doi:10.15888/j.cnki.csa.005754]

式语言中封装、继承、多态等机制, 针对不同的上下文需求, 初级设计模式课分为创建型、结构型、行为型三大类共 23 中设计模式[2,3], 每种设计模式都有它所针对的特定的问题, 简单地照本宣科讲授相关的定义和实现, 学生难以消化运用, 因此引入实验教学系统, 采用理实一体化教学手段, 在现阶段的设计模式教学中非常有必要.

基于上述分析, 本文设计和实现了面向设计模式课程实验教学系统-在线教务管理平台, 该平台采用了流行的Spring框架结构[4], 集成了多种设计模式, 既可以展示设计模式的单个特点, 也可以展现这些设计模式的复合协作, 有助于学生理解设计模式思想, 从而有效提高教学效果.

## 1 系统概述

### 1.1 设计背景及思路

教务管理系统是教育机构常用的一种信息管理系统, 为相关不同涉众提供信息服务, 主要对象包括教师、学生以及管理员. 基本的功能包括: 课表的添加、修改功能, 即管理员可以为每个专业的学生添加、修改课程信息, 可以是必修课也可以是选修课; 课表的显示功能, 即学生查看自己当前需要进行的课程以及该课程的详细信息, 在查看该课程的同时在允许范围内可以对该课程进行退课等操作, 可以查看已完成课程的情况, 教师可以查看自己的需要准备的课程, 以及正在进行该课程的学生信息; 成绩的输入、修改功能, 即教师有权限输入或者修改学生所选课程的成绩; 学分的统计功能, 即学生可以在系统查看已拥有的学分, 和未完成的学分, 以及学分的分布情况; 意见反馈功能, 即学生和教师都可以通过该系统反馈教学意见, 并且学生、教师和管理员都可以看到所有的反馈意见, 并且可以进行点赞, 评论等操作, 并且每次点赞该反馈的重要程度就会被提高.

为实现以上基本功能, 我们采用 B/S 架构, 以 Spring 框架作为平台基础, 设计和实现了相应的教务管理系统, 由于不同的涉众用户有不同的权限、视图以及安全等方面的需求, 为便于扩展, 具有良好的可维护性, 系统集成多种设计模式, 设计思路如下:

(1) 基于Spring框架中MVC设计模式的三层架构, 分别为 UI Layer(表现层)、Data Access Layer(数据访问层)和 Business Layer(业务逻辑层), 目的是保障系统的可维护性和可重用性, 加强系统对功能领域的专注.

(2) 基于多种设计模式来实现相关业务逻辑.

(3) 基于角色的访问控制机制, 教务管理系统将会有三方人员进行访问分别是教师、学生以及管理员. 不同的人访问教务管理系统将会有不同的访问权限.

### 1.2 系统的架构

系统架构层次使用了 MVC(Model–View–Controller) 模式[5]. MVC 模式是软件工程中的一种经典软件架构模式, 把软件系统分为三个基本部分: 模型(Model)、视图(View)和控制器(Controller). 控制器负责请求处理逻辑, 视图是用户接口界面, 模型部分是数据模型相关管理和数据库设计等各种功能. 本文采用 MVC 软件架构模式实现与用户的交互以及后台的数据管理, 能良好的体现高内聚性、低耦合性思想.

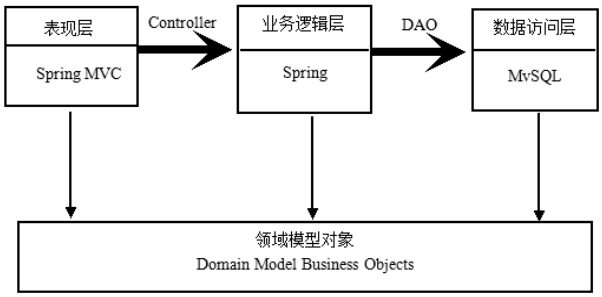


图 1 系统 MVC 架构图

如图 1 所示, 本文设计和实现的教务管理系统Spring框架下采用基于MVC设计模式的三层架构, 分别为 UI Layer(表现层)、Data Access Layer(数据访问层)和 Business Layer(业务逻辑层). 目的是保障系统的可维护性和可重用性, 加强系统对功能领域的关注.

根据以上的思路我们设计的面向设计模式教学实验的教务管理系统结构如图 2 所示. 其中, 身份认证模块主要负责对用户身份的甄别. 系统通过该模块对用户提供的登录凭证进行校验, 甄别用户是否是合法的访问主体, 确定用户是否有权进入系统. 授权访问模块主要负责判断一个资源要不要给当下的访问者操作或者展示. 这里系统会通过该模块将用户拥有的权限和请求的安全资源的权限列表进行匹配, 对匹配成功的用户授予相应的权限, 并且将资源对其开放, 若匹配失败则不具有相应的权限, 那么就通知用户不具有相应权限无法访问对应的资源. 数据加密模块主要职责为针对敏感数据进行 md5 加密保证数据的安全性

以及可靠性. 同时防止用户利用请求完成页面因为数据未加密而导致的非法跳转的情况. 同时保证了数据意外泄露的安全性. 课表管理模块, 学分、成绩管理模块以及意见反馈管理模块主要针对前文所述的功能需求, 在此不做赘述. 数据库我们采用开源数据库 MySQL[6].

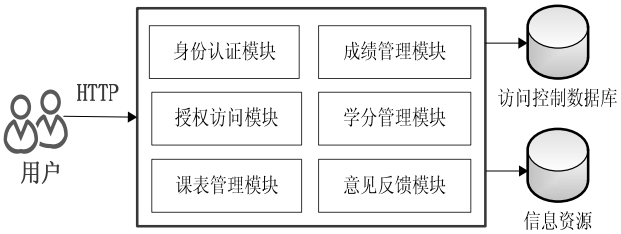


图 2　系统结构图

## 2　基于设计模式实现方案

在系统实现过程中, 我们融入了多种设计模式, 展示了它们在实际系统中的应用, 主要包括代理模式、工厂模式、策略模式、装饰者模式、观察者模式、组合模式、以及迭代器模式, 本节主要介绍这些设计模式在该系统中的使用情况, 由于篇幅所限, 重点介绍其中较有代表性的一种设计模式-装饰者模式的使用.

### 2.1 代理模式

代理模式通过使用代理对象完成用户请求, 屏蔽了用户对真正对象的访问. 同时做到对真实对象的控制和管理访问.

系统设计中, 使用 JDBC 实现网页前端接口与 MySQL 数据库交互的时候需要创建数据库连接, 而每一个数据库链接都有比较大的开销, 所以不到真正使用的时候没有必要创建数据库连接对象. 所以这里使用的代理模式的虚拟代理. 使用虚拟代理作为创建开销大的对象的代表. 而每一个数据表对应的操作类都有数据库连接, 所以对每一个数据表操作类都有一个对应的代理类. 这样只有在需要对相应的数据表进行操作的时候才会创建相应的对象, 之后代理类将请求给已创建的对象进行数据库操作,这样可提高系统的运行效率.

在实现中, 我们对每个数据表的操作都抽象为了一个接口如 AdminDAO, 然后 AdminDAOimpl 类是对应接口的具体实现, 代理类 AdminDAOProxy 是实现了同样的接口, 将真正的操作传递给 AdminDAOimpl 对象并调用 AdminDAOimpl 中的具体操作方法, 图 3 展示了代理模式结构的 UML 示意图.

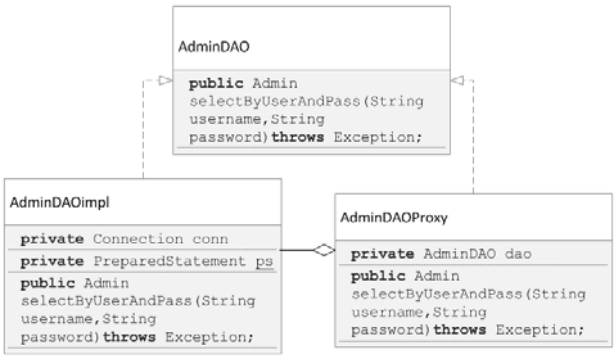


图 3　代理模式 UML 示意图

### 2.2 工厂模式

工厂模式主要用于对象的创建, 分为三种, 一种是简单工厂模式(Simple Factory Pattern). 在该模式中, 定义了一个类来负责创建其他产品类的实例, 可以根据参数的不同返回不同类的实例. 第二种是工厂方法模式(Factory Method Pattern), 它属于类创建型模式. 在工厂方法模式中, 工厂父类负责定义创建产品对象的公共接口, 而工厂子类则决定生成具体的产品对象, 这样类实例的创建就可以在子类中单独完成. 第三种是抽象工厂模式(Abstract Factory Pattern): 提供一个创建一系列相关或相互依赖对象的接口, 而无须指定它们具体的类. 在我们的系统中数据库表越多, 需要代理对象就越多.

在 model 层涉及到大量数据库操作, 比如登录, 添加课表等, 这样需要一个工厂类来获得所需要的代理对象. 因为系统仅仅是一个演示系统, 规模比较小, 实现中我们采用简单工程模式就可以. 在需要某个代理的时候就可以根据需要创建相应的代理对象. 若是使用频繁某一代理对象, 可以使用单例模式减少代理对象创建次数.

### 2.3 策略模式

在策略模式中, 存在一系列算法, 每一种方法称之为一个策略, 将每一个方法封装起来, 并让它们可以相互替换. 策略模式让算法独立于使用它的客户, 然后根据环境或者条件的不同选择不同的策略来完成该项任务.

在我们的系统中的开发中有很多情况需要用到策略模式, 如意见反馈管理模块. 反馈意见分为教师意

见和学生意见, 都有相同的操作如意见的添加, 修改等操作, 但操作实现的时候处理略有不同. 所以就采用了策略模式. 另外实现学生教师的各自的操作(行为)即相同的操作但是有不痛的具体的实现方式. 先实现关于意见反馈的操作的接口. 然后实现学生意见和教师意见, 在实现教师意见的时候将关于教师意见的操作添加进去, 实现学生意见的时候将关于学生意见操作添加进去. 这样即使要修改有关操作的具体实现也只需要修改有关操作的类即可. 下面以意见的更新操作为例, 首先声明一个意见更新的接口 IUpdateSuggestionAction. 然后教师和学生实现意见更新操作时的具体内容并不相同, 所以创建两个类 UpdateStuSuggestion 以及 UpdateTchSuggestion 分别实现这个接口. 创建一个类 Suggestions, 它拥有关于意见所有操作的方法, 同时拥有所有如 IUpdateSuggestionAction 一样行为接口类型的成员变量. 无论是教师的意见还是学生的意见都需要实现这些方法, 再创建 StuSuggestion 和 TchSuggestion 两个类继承 Suggestion 类, 然后在构造函数中将 UpdateStuSuggestion 以及 UpdateTchSuggestion 的实例赋值给对应接口类型的成员变量, 这样在成员函数中使用成员变量就可以直接调用其相应的行为.

### 2.4 装饰者模式

装饰者模式动态地将新的功能附加到对象上, 所以若要扩展功能, 装饰者提供了比继承更有效的方法. 装饰者模式给一个对象动态的添加新的功能, 装饰者和被装饰者实现同一个接口, 装饰者持有被装饰者的实例.

在该模式中, 首先声明一个主题接口, 通过实现这个接口创建一个装饰者类和被装饰者类. 由于实现了同一个接口所以拥有相同的方法, 然后在装饰者类中声明一个主题接口类型的变量以便拥有被装饰者对象. 在装饰者类中添加需要扩展的功能, 然后在执行接口继承的方法的同时执行扩展的方法, 这样相对于被装饰者类, 同样的方法却多出来扩展的功能. 也符合上面所说的在运行的时候增加行为.

教务管理系统中每个学生或者教师提交的反馈意见都有点赞等功能. 也就是说每次点赞后在完成其他相应操作(记录点赞人等操作)的同时还需要将点赞数量更新或者将该条意见的重要程度进行更新. 我们将原始的点赞操作定义为一个抽象类(IUpvoteSuggestion), 具体的更新重要程度的类(ImportantDecorator) 和改变点赞次数的类(ChangeTimesDecorator) 则继承了公共的装饰者类(Decorator), 而该装饰者类继承了抽象类 IUpvoteSuggestion, 同时与该抽象类存在组合关系. 使用装饰者模式可以较容易的扩展增加功能, 比如通过对点赞操作的再一次装饰使其拥有在记录点赞人的同时更新点赞数据以及重要程度的功能, 实现面向对象中的“对修改是封闭的, 对扩充是开放的”设计原则. 图 4 展示了在我们的系统中所采用的装饰者模式 UML 类图.

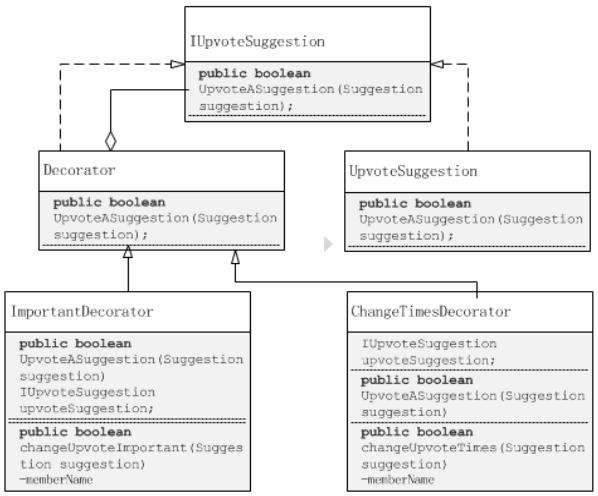


图 4 装饰者模式 UML 示意图

### 2.5 模式的复合使用

模式的复合主要体现在用户界面相关的设计中. 复合并不是简单的将几个设计模式组合到一起, 而是更侧重于解决一般性或者重复性的问题. 如前所述, 该系统设计是基于 Sping 框架, 而该框架的重要特征是采用了 MVC 架构. 在我们的实现中, 复合的模式主要包括了观察者模式、组合模式、策略模式和迭代器模式.

观察者模式定义了对象间的一种一对多依赖关系, 使得每当一个对象状态发生改变时, 其相关依赖对象皆得到通知并被自动更新. 观察者模式也称发布-订阅(Publish/Subscribe)模式、源-监听器(Source/Listener)模式等. 观察者模式所建立的一对多的关系通常称为一个观察目标对多个观察者的关系, 每当观察目标获取到新的数据, 即观察目标本身发生改变就通知其对应的所有观察者, 然后观察者获取到新的数据, 然后跟

据自身的需要进行更新, 而且这些观察者之间不存在相互依赖的关系, 即可以根据需要添加观察者以及删除观察者, 使整个系统更易于扩展.

组合模式把一组相似的对象当作一个单一的对象处理. 一般而言, 组合模式依据层次式的树形结构来管理对象, 当一组对象和单个对象执行相同操作时便可使用组合模式, 这样可以更加方便的处理一组对象. 比如 XML 结构中的原子节点和复合节点可以用同样操作进行访问. 迭代器模式相对比较简单, 它提供一种方法访问一个容器对象中各个元素, 而又不暴露该对象的内部细节, 在 JDK 中, 已经把迭代器模式融入语法之中, 针对于所有的 Java 容器类, 都定义了相同的接口用于遍历其中元素.

在我们的系统用户界面部分实现中, 综合使用了上述设计模式, 管理员在系统后台添加专业课程后, 对应专业的学生的课表就需要更新, 添加上对应的课程. 由于权限不同, 每个专业的学生看到的课表也仅仅是自己专业的课表. 每个专业都有自己的课表操作类来实现对该专业学生课表的更新. 我们使用观察者模式完成课表的更新. 主题对象获取到管理员提交的新数据便通知注册的操作类, 然后操作类开始对该专业学生课表的更新. 课表的显示视图控件有多个, 我们采用组合模式, 针对不同的控件定义一个公共的父类和抽象方法, 具体的显示逻辑在控件子类中实现, 这样我们只需按照统一的接口调用它们的方法, 简化了客户端逻辑; 同时, 在更新操作中由于每一个专业的学生都不止一个, 这样我们将专业里的学生对象放到该专业的列表中, 而对学生的遍历操作则使用迭代器模式. 在管理系统中的视图和控制器两部分刚好实现策略模式. 视图作为一个对象对应主体部分来调用不同的策略, 而控制器作为行为部分, 具体实现不同的策略. 在视图中的每一个行为无论是显示还是更新都是交给控制器来实现具体的操作. 策略模式本身的主体和行为的解耦性也正好完成了视图和控制器的解耦.

## 3 结语

设计模式是软件工程专业的特色和核心课程, 对于学生掌握软件工程思想、进一步学习架构方面知识有着重要作用, 笔者在教学实践中, 针对课程特点, 设计和实现了一个面向设计模式实验教学的教务管理系统, 该系统基于 Spring 框架, 采用 B/S 架构, 融合了多种设计模式, 不仅展现了模式的单个应用, 同时也示例了若干设计模式的协作复合. 从课程反馈来看, 该实验系统的采用可有效激发学生学习的兴趣和主动性, 加深理论理解程度, 提高动手实践能力, 达到了预期的教学效果.

## 参考文献

1 严华,张欲蓉.设计模式在通讯接口设计中的应用.计算机系统应用,2012,21(5):172–175.

2 Erich G, Richard H, Ralph J, et al. Design Patterns: Elements of Reusable Object-Oriented Software.北京:机械工业出版社,2005.

3 Freeman E, Robson E, Bates B, Sierra K. Head First Design Patterns. US: O'Reilly Media, 2004.

4 Johnson R, Hoeller J, Arendsen A, et al. Professional Java Development with the Spring Framework. New York: John Wiley & Sons, 2009.

5 刘红霞,陆文迪.改进的 MVC 设计模式的研究与应用.计算机工程与科学,2015,37(9):1688–1691.

6 韩兵,王照清,廖联军.基于 MySql 多表分页查询优化技术.计算机系统应用,2016,25(8):171–175.