第16卷 第12期
2006年12月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 16 No. 12
Dec. 2006

基于 AOSD 的 Web 架构规划

潘 晔,任广伟
(贵州大学,贵州 贵阳 550003)

摘 要:基于 MVC 设计模式的软件框架逐渐兴起,成为 Web 架构的开发主流。为了在此架构基础上进一步提高模块结构和组件的重用,提高软件开发的效率,使得架构更加易于扩展和维护,文中基于 AOSD(Aspect - Oriented Software Development)方法来规划 Web 架构,利用 AOSD 的动态和静态的用例分析方法,来有效地实现关注点的分离,并使用方面来包装关注点,实现对当前流行的 Web 框架的改进。利用面向方面的分析和编程方法来规划架构,令 Web 应用程序的开发更加灵活、高效。

关键词:AOSD;AOP;用例切片
中图分类号:TP311 **文献标识码**:A **文章编号**:1673 - 629X(2006)12 - 0052 - 03

Web Structure Programming Based on AOSD

PAN Ye, REN Guang-wei
(Guizhou University, Guiyang 550003, China)

Abstract: Based on the gradual rise of the software frame of the MVC design pattern, it becomes the main development current of the structure of Web. For the sake of this foundation mold a structure of further exaltation and module of heavy use, raise the efficiency of the software development, make structure easy to expand and support more. This text programs the structure of Web according to the AOSD (Aspect - Oriented Software Development) method. It makes use of the methods of analysis and programming to plan the structure, which let Web application programme more flexible and effective.

Key words: AOSD; AOP; use case slice

0 引 言

网络技术和 Internet 的迅速发展,使得大量的 Web 应用不断涌现。其中良好的 Web 架构是 Web 应用存在的必要条件。一个好的架构不仅要具备易用、开发高效、易维护、高扩展等优点,更主要的是实现架构的重构(refactoring)。如果没有重构,程序的设计会逐渐遭到破坏。当为短期目的就贸然修改代码,程序将逐渐失去结构,代码的设计意图将变得晦涩,重复的程序结构或代码出现在系统的各个模块,凌乱的系统结构必然造成高维护和难于测试的局面,更不用说系统的扩展性了。因为系统需要扩展的点缠绕在整个框架中。

具有易用性、通用性和良好的可扩展性等优点的架构正逐渐成为企业应用信息系统市场的主流。这样,在众多的设计模式中 MVC(Model - View - Controller)架构[1]脱颖而出,成为 J2EE 平台的首选,相应的基于 J2EE 的 MVC 架构不断涌现出来,其中以比较成熟的 Struts[1], Webwork 架构为最流行,并实现了大量非常成功的案例,成为当今 Web 应用框架的主流。

目前的 MVC 结构,多是使用组件构建系统,利用组件封装的特性,对外隐藏内部结构、提供接口来实现系统间消息传递、功能调用。当前的新软件都主要基于组件进行开发。但考虑到系统的目标就是满足需求,也就是满足涉众(最终用户、项目发起人、开发人员)关心的功能需求、非功能需求或系统的设计约束等关注点。它的内容通常要比系统的需求更多一些。所以成功地分离关注点,并把关注点分解到模块中,是需求分析的关键。尽管一些关注点可用特定的和单独的组件实现,但通常会发现许多关注点,任何一个组件都无法完全满足要求。这就是影响多个组件的横切关注点。稳妥的处理横切关注点是使模块间松耦合和代码重用等需求的保证。

为此出现了基于 AOSD(Aspect - Oriented Software Development)的软件开发方法,最大限度地分离关注点,有效地弥补了组件技术的缺陷。

1 面向方面的软件开发(AOSD)

AOSD[2]是对面向方面编程(AOP, Aspect - Oriented Programming)[3]的扩展,包括一个贯穿从需求到分析和设计、实现和测试的面向方面的软件开发的整体方法。其目标主要是围绕如何使系统更好地模块化。AOSD 不仅仅

收稿日期:2006 - 03 - 28
作者简介:潘 晔(1982 -),男,山西晋中人,硕士研究生,从事数据库、信息集成的研究;任广伟,副教授,硕士生导师,主要研究领域为网络、数据库、多媒体传输。

是 AOP,它包括一系列的模块建模技术。但如何识别关注点、如何对关注点建模、如何分离方面和非方面仍需要系统化的方法。于是,借助成熟的用例驱动建模方法,使用用例模型来真正实现面向方面的软件开发。

AOSD 中涉及的基本概念[2]:

(1)用例切片(Use Case Slice):是对模块中特定于某个用例的部分进行模块化,它使用方面作为合成机制,在共享类的基础上扩展特定于某个用例的特性(属性、关系和操作)。

(2)非用例特定切片(non - UC - specific slice):不在任何用例切片中定义,供特定用例切片扩展或调用。

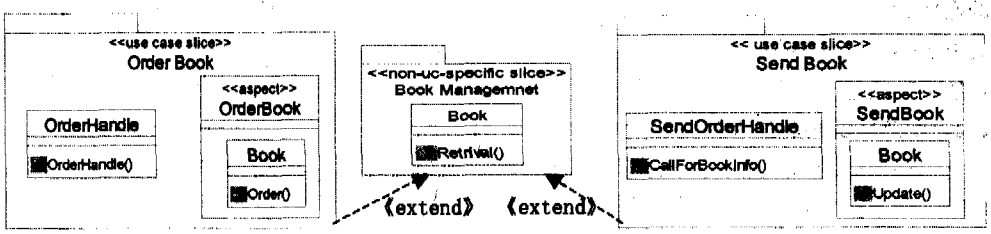

图 3　用例切片与非用例特定切片关系

(3)用例模块(Use Case Model):把需求到实现的各个阶段的用例切片组合起来形成的即是用例模块。

2　Web 架构规划

利用 AOSD 方法来规划 MVC 框架的 Web 应用,可以避免以往利用面向对象设计中,业务对象会因为混合的属性和与对象最初意图不一致的操作而变得混乱的缺点。AOSD 中为了有效地分离关注点把用例分为对等用例、扩展用例(具有包含、扩展及泛化关系)。对应的 AOP 中处理以上两种用例分别称为动态横切[4](包含方面(aspect)、连接点(join point)、切入点(pointcut)、通知(advice)等概念)与静态横切[5](通过引入附加的方法字段和属性来修改对象的结构)。下面分别以静态横切、动态横切的处理用例的方法分析网上书店的 Web 架构。

2.1　静态横切实现对等用例

系统中用户订购用例与图书配送用例从用例建模的角度,两者是对等用例,但通过分析两个用例都使用到了实体图书,而且订购用例调用了实体图书的图书检索(Retreval)和订购(Order)两个操作,图书配送用例同样也调用了 Retrival,更改库存的 Update 操作。图 1 与图 2 为两用例的交互图。

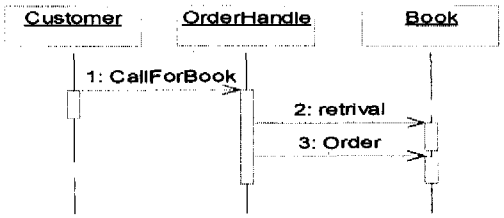

图 1　图书检索用例交互图

调用实体图书的操作中有两用例都使用 Retrival 操作,订购用例与图书配送用例还有各自特有的 Order 和 Update 操作,图 3 为相应的用例切片与非用例特定切片的扩展关系。

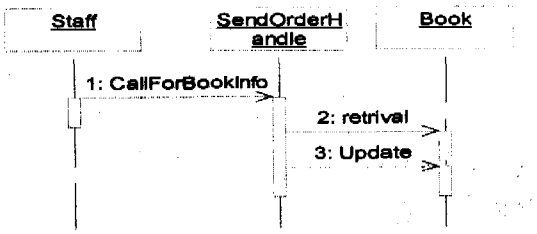

图 2　订购用例交互图

两个用例切片把对 Book 实体特定的操作封装为方面,实现了对实体 Book 的操作扩展。如此,对于实现用户订购用例与图书配送用例的 MVC 框架如图 4 所示。

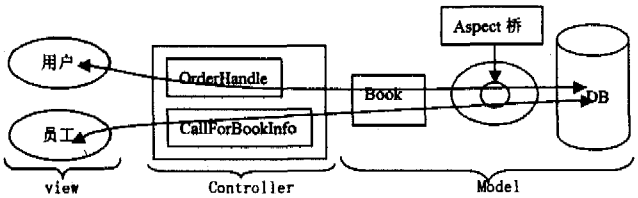

图 4　用户订购用例与图书配送用例的 MVC 框架

MVC 框架通过使用用例切片,实现了关注点的分离,并把涉众关注点分离到方面中。通过图中的"Aspect"桥,把涉及的用例操作扩展到 Book 实体,而不是将角色混合到受影响的 Book 类中。"Aspect"桥是密切紧耦合的,它知道连接的行为模式和组件类,这样降低了组件的耦合,提高了 MVC 架构的重用性。

模拟的代码片断如下[6]:

```
public class Book{
    String BookName;
    String author;
    String ISBN;
    ……
    String retrival( String CustomerName);
    }//基本的 Book 类,便于重用
    public aspect bridge extends Promise{
    declare parents:Book implements PromiseObject;
    public void Book. Order(String CustomerName){
    System. out. println("用户订购");
    …… }
    private boolean PromiseObject. Update(){
    System. out. println("图书配送");
    …… }
    }//Book 类继承接口的函数,实现对其的操作扩展
    public interface PromiseObject{
    public abstract aspect Promise{
    private void PromiseObject. Order(String CustomerName);
```

```
private boolean PromiseObject.Update();
……}
}//定义协议接口,包含嵌入到实体中操作的方面
```

以上的程序片断中,抽象方面 Promise 只了解用例在其中充当的角色,对于 Book 类,Promise aspect 一无所知,只有通过详细定义的接口,aspect 才能了解到 Book 类并与这些类进行交互。通过 aspect 桥(方面 bridge)把要扩展的操作封装在接口中,Book 类通过 declare parents 声明扩展 PromiseObject,减小了 Book 类的内聚性。当 Book 类支持较多的角色时,其重用性表现会更加突出。

2.2 动态横切实现扩展用例

以用户订购用例为例,当用户订购图书时,控制类检索的结果是,该图书缺货或新书还未上市。此时应该具有处理该情况的策略。即出现一个异常界面提示用户,并允许用户输入 E-mail 把书的信息及时通知用户。所以,需要对用户订购用例扩展,添加异常处理用例。图 5 为异常用例切片。

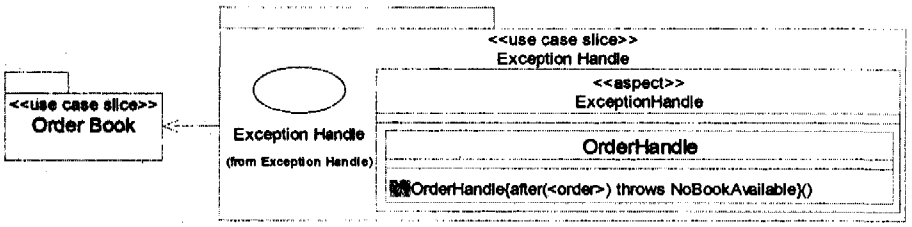

图 5 异常用例切片

在切片中的的方面部分实现了动态的对控制类 OrderHandle 扩展,表现为对用户订购用例的扩展。从用例切片中可以看到,连接点为控制类中函数 Order-Handle,当其调用实体类方法 Order 后,发现有 NoBookAvailable 异常被捕获,则进入异常处理用例。如下即是对 OrderHandle 的扩展操作。

OrderHandle{after(<order>) throws NoBookAvailable{

具体的模拟的代码片断如下:

```
public aspect Order Handle{
 pointcut OrderHandleException();
    withincode(void OrderHandle.Order_Handle())
    &&call(void Book.Order(String CustomerName));
after():OrderHandleException(){
    //处理具体的异常
    }
}
```

以上的片断中 withincode 和 call 指明了连接点的位置,after():OrderHandleException()说明了是后(after)增强(advice),即调用 order 函数后可能被触发。使用面向方面来处理用例扩展,使得各个用例之间保持了高度的松耦合。如果系统修改处理异常用例,只需要调整 Order Handle 方面即可,而不影响任何其它用例。

3 结 论

通过运用 AOSD 分析系统需求和使用 AOP 中动态和静态横切对 Web 架构的代码实现,使得关注点被建模在用例切片中,系统各个功能用例之间结构、层次清晰明确。同时 AOSD 分析方法可实现一种非强制性的整洁和模块化的方法来添加物件行为。这样不但易于基于用例的单元测试,而且克服了 OO 中的业务对象会因为混合的属性和与对象最初意图不一致的操作而变得混乱的缺点。这样便于架构重用,使当前的 Web 框架更趋于完善。

参考文献:

[1] Husted T. 实战 STRUTS[M]. 北京:机械工业出版社,2005.

[2] Jacobson I,PAN-WEI NG. AOSD 中文版——基于用例的面向方面软件开发[M]. 北京:电子工业出版社,2005.

[3] Laddad R. I want my AOP! Part1~ Part3[EB/OL]. 2002-03-01. http://www.javaworld.com.

[4] 张广红,陈 平. 关于 AOP 实现机制和应用的研究[J]. 计算机工程与设计,2003(8):16-17.

[5] 陈 峰,饶若楠. 一种基于 AOP 的分布式企业应用开发技术[J]. 计算机仿真,2004(7):172-173.

[6] 曹东刚,梅 宏. 面向 Aspect 的程序设计———一种新的编程范型[J]. 计算机科学,2003,30(9):95-101.

(上接第 51 页)

[8] Yan Hanbing, Liu Yingchun. A New Algorithm for Finding Shortcut in a City's Road Net Based on GIS Technology[J]. Chinese J Computers(in Chinese),2000(2):210-215.

[9] Lu Feng, Zhou Chenghu, Wan Qing. An Optimum Vehicular Path Algorithm for Traffic Network Based on Hierarchical Spatial Reasoning[J]. Journal of Wuhan Technical University of Surveying and Mapping(in Chinese),2000,25(3):226-234.

[10] HUANG Yun-wu, JING Ning, Elkea R. A Hierarchical Path View Model for Path Finding in ITS[J]. Geoinformatica, 1997,1(2):125-159.

[11] Henzinger M R, Klein P, Rao S, et al. Faster Shortest path Algorithms for Planar Graphs[J]. Journal of Computer and System Sciences,1997,55(1):3-23.

[12] Ma Jun, Ma Shaohan. The Algorithms for Computing and Updating the Shortest trees[J]. Computer Research & Development (in Chinese),1995,32(12):45-49.

[13] Ahuja R K, Mehlhom K. Faster algorithms for the shortest path problem[J]. Journal of the Association for Computing Machinery,1990,37(2),213-223.