基于 UML 模型的面向方面建模①

UML – Based Aspect – Oriented Modeling Approach

张敬波 刘琳岚 张恒锋 舒 坚 (南昌航空大学 计算机学院/软件学院 江西南昌 330063)

摘 要: 面向方面建模的目标是将面向方面的思想应用到设计阶段,目前面向方面建模的相关研究仍处于起步阶段。本文采用基于 UML 模型的方法,分别从动态横切和静态横切的角度,根据连接点、切入点、通知、类型间声明的语义,结合 AspectJ 对横切关注点进行建模。文章所描述的面向方面建模方法考虑了连接点所在的业务环境,用顺序图和活动图描述连接点,用交互概览图描述切入点,用活动图描述通知,用类图描述类型间声明,使得编写方面的编码人员能够更好地理解业务需求。

关键词: 面向方面建模 UML 横切关注点 AspectJ

1 引言

面向方面编程(Aspect – Oriented Programming, AOP)是一种新的软件开发方法,它利用称为‘横切’的技术,剖解开封装的对象内部,将那些影响了多个类的非核心功能需求封装到一个可重用模块(即‘Aspect’,方面)[1],可以解决软件系统中存在的代码分散和代码纠缠问题,提高软件的开发效率、可重用性、系统的可理解性和可维护性。面向方面建模(Aspect – Oriented Modeling,AOM)是在软件开发的设计阶段对方面进行建模,AOM 利用建模语言(如 UML)标识、分析、管理和表示软件设计和构架中的横切关注点,使所设计的软件具有更好的模块化设计,保持需求、设计和实施间的连续性,有利于面向方面软件开发过程的顺利进行,与面向对象建模一起构成了对系统核心功能和非核心功能需求的建模。

目前,国内外对 AOM 的研究处于探索阶段,还没有一个统一的建模标准,常用的方法有两种:一种是利用 UML 的扩展机制建模,如 Aida Atef Zakaria 通过扩展 UML 的符号来表达切入点、连接点、通知等面向方面的核心概念[2];另一种是不基于扩展 UML 进行建模,如 Walter Cazzola 提出一种针对连接点进行建模的方法[3],Vincenzo Grassi 基于 UML 对特定的面向方面编程语言 AspectJ 中所有的概念进行建模[4],UML 的模型图表示 AspectJ 的概念,Walter Cazzola 仅对连接点进行了建模,Vincenzo Grassi 对连接点建模时只考虑了连接点本身,没有考虑其所在的业务代码环境,使得编码人员不能准确理解业务需求。

本文针对大型应用软件系统存在多个横切关注点的特点,将 AOM 引入系统的设计,依据省级广电宽带客户服务系统的实际需求,采用上述第二种方法对其中的横切关注点——权限控制进行建模。

2 横切关注点

在省级广电宽带客户服务系统中,权限控制、日志管理、事务处理、数据缓存等都属于横切关注点,他们分散在各个模块中解决同一样的问题,跨越多个模块。

以权限控制为例,应用系统的用户包括:省公司总经理、副总经理、各部门经理、系统管理员,地市县各分公司的总经理、副总经理、各部门经理、营业员(操作员)、派工员、维修安装人员等等,将 RBAC(Role – based Access Control,基于角色访问控制)模型应用到广电宽带客户服务系统中,则该系统的角色有:总公司负责人、分公司负责人、系统管理员、营业员、派工员、维修安装人员,为每个用户分配相应角色,每个角色对系统中资费调整、IP 地址池管理块、派工管理、前台营业等模块具有不同的操作权限。例如,系统管理员、营业员、派工员对资费调整模块和 IP 地址池管理模块的操作权限分配如表 1 和表 2 所示,表中空白单元格表

① 基金项目:国家自然科学基金项目(60773055)

示该角色对相应的模块不具备相应的操作权限。系统中每个模块都需要检测登录用户所对应的角色是否具有权限,因此存在横切关注点——权限控制。

表1　资费调整权限分配示例

权限表	资费调整模块			
系统管理员	浏览	添加	修改	删除
营业员	浏览			
派工员				

表2　IP 地址池管理权限分配示例

权限表	IP 地址池管理模块			
系统管理员	浏览	添加	修改	删除
营业员				
派工员	浏览	添加	修改	删除

3　动态横切建模

动态横切是在程序的执行过程中织入新的行为。在 AspectJ 中,横切多以动态形式发生,动态横切通过横切模块的方式扩展或取代了核心程序的执行流,以此修改系统的行为。

动态横切主要由切入点和通知共同组成,如图1所示,包含两个部分,左边是切入点,用以捕捉程序执行中的特定连接点,并搜集该连接点上下文的程序结构;右边为通知,定义了在捕获的连接点处执行什么样的操作。

图1　权限控制动态横切图

3.1 连接点图和切入点图

切入点图描述程序运行中一系列横切行为事件的发生,而每个事件又可以用一个连接点图来描述,连接点存在于各个模块中。

在广电宽带客户服务系统中,资费调整模块编号为 P001,地址池管理模块编号为 P002,资费调整模块中的 servlet 为 P001Servlet,dto 为 P001DTO,dao 为 P001DAO。其中 P001Servlet 的核心代码如下:

```
public class P001Servlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
HttpServletResponse response) throws
    ServletException, IOException {
      P001DTO dto = new P001DTO();// 创建一个 DTO 对象
      dto.setFEE_PRC(request.getParameter("FEE_
PRC"));// 将单价赋给 DTO 的 FEE_PRC 属性
      String processmode = request.getParameter
("processmode");// 得到 processmode 值,即判断要进行的操作是添加、修改还是删除?
      P001DAO dao = new P001DAO();// 创建一个 DAO 对象
      dao.setConnection(conn);// 连接数据库
      boolean flag = false;
      if(processmode.equals("insert")) {// 如果是添加操作
    flag = dao.executeInsert(dto);
      }
      if(processmode.equals("modify")) {// 如果是修改操作
        flag = dao.executeModify(dto);
      }
      if(processmode.equals("delete")) {// 如果是删除操作
    flag = dao.executeDelete(dto);
      }
    }
  }
```

dao.executeInsert(dto)、dao.executeModify(dto)和 dao.executeDelete(dto)三个方法是连接点。本文用顺序图描述 P001Servlet 的流程,它可以清楚地显示哪些方法是连接点、连接点是在何时被捕获的,并提供了描述当前事件上下文的信息;用活动图中的 AcceptEventAction[5] 表示连接点,它继承自 UML 中的 Action,它的语义是等待特定条件下一个事件的发生,连接点图的构造型为 call。在 UML 中,这个事件被指定为 UML 中的触发器,能对方法调用、属性修改进行响应,用 AcceptEventAction 可以很好地描述连接点的语义。权限控制的连接点图如图2所示。

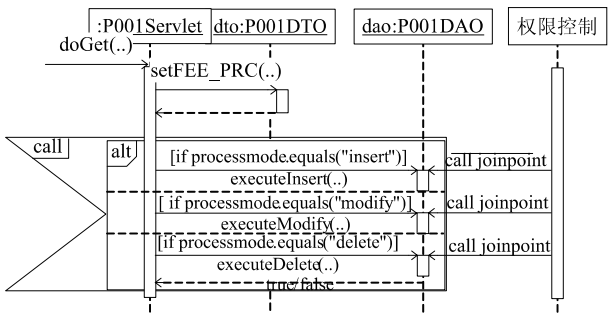

图 2　权限控制的连接点图

对连接点建模后，可以对切入点进行建模。系统中各个模块的 dao 都继承自一个 BASEDAO，每个 dao 的 executeInsert()、executeModify()、executeDelete()方法都在 BASEDAO 中定义。因此捕获连接点时不需要写多个切入点，只需捕获 BASEDAO 中的这三个方法就可以了，该切入点的代码如下：

```
pointcut accessPointcut(BASEDAO dao)：
(call(boolean BASEDAO.executeInsert(..))
&&target(dao)|| // 捕获插入方法
(call(boolean BASEDAO.executeModify(..))
&&target(dao)|| // 捕获修改方法
(call(boolean BASEDAO.executeDelete(..))
&&target(dao))；// 捕获删除方法
```

某一个角色在同一时刻间只会执行插入、修改或删除中的一个操作，因此可以用活动图表示，在活动图中要执行的动作是捕获连接点，本文已用顺序图对连接点进行了建模，可以将活动图和顺序图结合组成交互概览图以更好地表示切入点图，如图 3 所示，图中三个连接点由一个切入点进行捕获。

3.2 通知图

通知图用来指定当一些事件被切入点捕获时执行的逻辑，即要做什么，包括新行为的增加、新行为是怎样执行的。从图 3 可以捕获连接点处的参数，在通知图中可以使用这些参数，从而获得连接点处的上下文信息。

权限控制的通知的核心代码如下：

```
boolean around(BASEDAO dao)：accessPointcut
(dao) {
String subdaoname = dao.getClass().getSim-
pleName()；// 获取实际捕获连接点处 dao 的名称
```

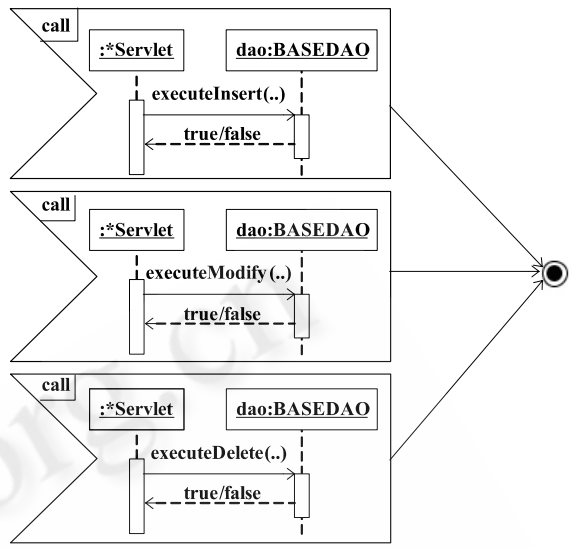

图 3　权限控制的切入点图

```
String executemode = thisJoinPoint.getSignature().
getName()；// 获取实际捕获连接点处方法的名称
    // 判断当前用户所在的角色是否具有权限
    if (dao.check(subdaoname,executemode)) {
// 如果具有权限
    System.out.println("pass")；
    proceed(dao)；// 回到连接点处继续执行连接
点后面的代码
    return true；
    }
    else {// 如果没有权限
    System.out.println("no pass")；
    … // 转到提示没有权限页面
    return false；
    }
  }
```

这里首先捕获两个参数，即连接点处 dao 的名称和方法名称，然后从 session 中取出当前用户对应的角色，判断该角色是否对该 dao 的这个方法具有权限，如果具有权限，则程序在连接点处继续运行，否则转到提示‘没有权限’的页面。

通知的实现有比较明显的流程性，用活动图可以很好的表示该通知体内的程序运行过程。权限控制的通知如图 4 所示。

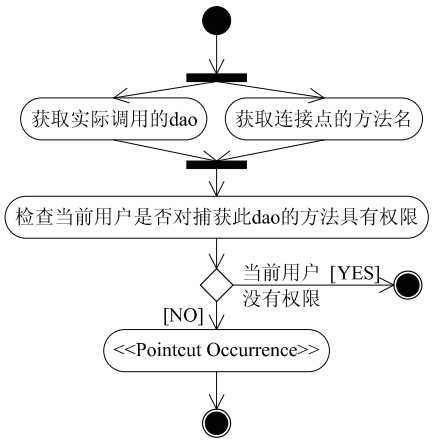

图 4　权限控制的通知图

3.3 动态横切图

对权限控制的切入点、通知分别进行建模后，如图 1 所示，将切入点图和通知图组合，可以构成权限控制的动态横切图。

4　静态横切建模

一个静态横切关注点通过增加新元素改变系统的静态结构。在 AspectJ 中，这种关注点叫做类型间声明。本文用一个类型间声明图表示一个类型间声明。类型间声明图包两个部分：左边是一个类图，指定了受类型间声明影响的类；右边包括对左边类修改后的类，它表述了属性和方法增加到这些被修改的类中。

权限控制的类型间声明如下：

```
private AuthorizationManager BASEDAO. authMgr = new AuthorizationManager();
public boolean BASEDAO. check(String subdaoname,String executemode) {
return (authMgr. verifyAccess(subdaoname,executemode));// 验证
}
```

权限控制的类型间声明图如图 5 所示。左边包含 BASEDAO 类，右边描述了在 BASEDAO 类中增加一个 AuthorizationManager（对象名为 authMgr）和一个 check(String subdaoname,String executemode) 方法，具体的验证通过 AuthorizationManage 类中的 verifyAccess 方法实现。

图 5　权限控制的类型间声明图

5　结束语

运用面向方面的思想进行软件系统开发的分析和设计，可以降低软件开发的复杂度，有效提高代码的可读性、可复用性，降低面向方面软件开发的风险和成本，充分发挥 AOP 的优点。AOM 的目标是将面向方面的思想应用到设计阶段。本文采用基于 UML 模型的方法，充分考虑连接点所在的业务环境，用顺序图和活动图描述连接点，用交互概览图描述切入点，用活动图描述通知，用类图描述类型间声明，给出了一个应用的过程实例。

在以后的工作中，我们希望能把所提出的符号整合到更复杂的 MDA[6] 框架，以期能够从方面模型转换为可执行的代码。

参考文献

1 amandaxp. AOP 技术简介. http://dev2dev. bea. com. cn/

2 Aida Atef Zakaria, Dr. Hoda Hosny, Dr. Amir Zeid. A UML Extension for Modeling Aspect – Oriented Systems. Workshop on Aspect – Oriented Modeling with UML. 2002.

3 Walter Cazzola, Sonia Pini. Join Point Patterns: a HighLevel Join Point Selection Mechanism. Workshop on Aspect – Oriented Modeling with UML. 2006.

4 Vincenzo Grassi, Andrea Sindico. UML Modeling of Static and Dynamic Aspects. Workshop on Aspect – Oriented Modeling with UML. 2006.

5 OMG Unified Modeling Lanauge Specification – UML 2.0 Superstructure Specification, p. 249.

6 MILLER J, MUKER J, MDA Guide Version 1. 0. 1. Copyright(c) 2003 OMG. http://www. omg. org/doc/omg/03 – o6 – 01. pdf