面向方面状态模型的UML扩展实现

汤根生，王 姚
（广东工业大学 计算机学院，广东 广州 510006）

摘 要：面向方面编程作为一种基于关注分离的软件开发思想，抽取出软件的功能性和非功能性属性，并引入实现横切关注点的方面，以织入的方式完成系统集成。然而面向方面编程方法只体现在软件生命周期的编码阶段，而缺少在设计阶段的支持。通过分析面向方面编程的概念及其特点，利用UML的扩展机制将方面加入状态图中，给出基于AspectJ语法语义扩展UML的状态模型，实现状态图方面与核心组件之间的织入关系，最终实现代码自动生成。

关键词：面向方面；状态模型；UML扩展机制；横切关注点；AspectJ
中图分类号：TP311.5 **文献标识码**：A **文章编号**：1673-629X(2011)01-0116-04

Realization of Aspectual State Model by Extending UML

TANG Gen-sheng, WANG Yao
(Faculty of Computer Science, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: As a new software development paradigm, AOP is based on separation of concerns, introduces aspects to achieve crosscutting concerns, and extracts the software functional and non-functional properties to weaving the integration of the system. However, AOP embodied only in the encoding stage of software life cycle but lack of supporting at the design stage. This article analyzes AOP concepts and characteristics, using UML's extension mechanisms to join the state diagram in terms, given the state model with UML extension mechanism based on AspectJ semantic and syntax, achieving the weaving of state diagram and the core components, and ultimately code automatically generated.

Key words: AOP; state model; UML extending machanism; crosscutting concern; AspectJ

0 引 言

面向方面编程(Aspect-Oriented Programming)作为基于关注分离的新的软件开发范例，利用横切关注点的技术，将封装好的对象进行解剖，抽取出影响多个类的非功能核心功能，封装成可重用模块，以提高模块的内聚性，降低模块间的耦合度，使系统易于维护与复用，并具有较好的体系结构。

AOP技术的日趋完善的同时，一个能够贯穿从需求分析到设计、实现等全过程的面向方面软件开发(Aspect-Oriented Software Develop)方法也随之孕育而生。面向方面建模(Aspect-Oriented Modeling)是AOSD的一个重要内容，它利用建模语言，在软件开发的分析、设计和实现阶段进行对方面进行建模，使软件具有更好的模块化设计，保持需求、设计和实施阶段的连续性，有利于面向方面软件开发过程的顺利进行。

然而，当前AOM并没有得到很好的支持和关注，在建模工具上的支持也不够，标准的、统一的建模理论有待建立。事实上，这也是影响AOP技术推广的重要原因之一，基于此，文中提出基于面向方面的UML状态图扩展建模的思想和方法。

1 面向方面的程序设计

面向方面编程技术[1]是面向对象编程的补充和完善，利用“横切”技术，面向方面软件设计方法把系统建模分成两部分：核心组件和方面。核心关注点(即核心组件)是业务处理的主要流程，与之关系不大的部分就是横切关注点(即方面)。横切关注点的特点就是经常贯穿于核心关注点的多处，而各处都基本相似，功能一样，如日志、权限认证、事务处理等。AOP的作用在于分离系统中的这些横切的关注点，将它们与核心关注点分离开来。

1.1 AspectJ

AspectJ是第一种实现AOP的语言，它是对于Java语言的一种无缝扩展，在技术上比较成熟。文献[1]中描述了AspectJ的一些基本概念，其中advice、join-

收稿日期：2010-04-20；**修回日期**：2010-07-09
基金项目：国家自然科学基金项目(60474072Z)；广东省自然科学基金项目(07001774)
作者简介：汤根生(1984-)，男，江西赣州人，硕士研究生，研究方向为计算机网络系统；导师：张立臣，教授，研究方向为分布式与实时系统。

point、pointcut 关系如图 1 所示。

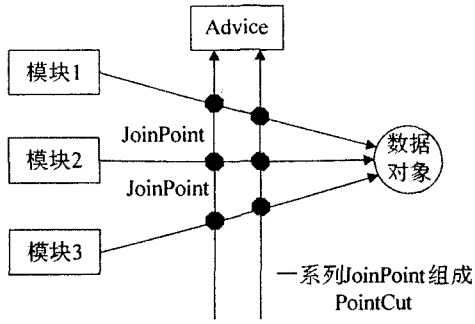

图 1　advice、joinpoint、pointcut 关系图

AOP 实现过程分为三个阶段,即方面分解、实现和方面的织入,如图 2 所示:方面分解把横切关注点从系统需求中抽出来,然后形成一个个方面;方面实现则把分离出来的方面独立地实现;方面织入把独立实现的方面集成到系统核心组件,最终构成需求的系统。

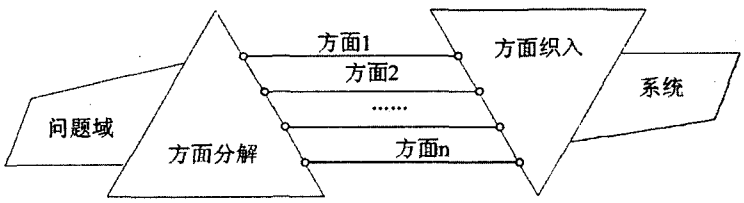

图 2　AOP 实现过程

1.2　动态横切

动态横切又称为动态代理技术,是通过切入点和链接点在一个方面中的创建行为的过程,连接点可以在执行时横向地应用于所有对象。可以说动态横切技术基本上代表了 AOP。

1.3　静态横切

静态横切又称为静态织入,与动态横切不同的是,它不修改一个给定对象的执行行为。相反,它不修改原有职责的基础上增加新的职责。AOP 实现中,静态横切又通常被称为 introduce 或 mixin。

2　UML 状态图建模设计

统一建模语言[2](Unified Modeling Language),是一种标准的图形化建模语言,它用定义完善的符号来图形化地展现一个软件系统。UML 的使用可以贯穿于软件开发周期的每一个阶段,适用于数据建模、业务建模、对象建模和组件建模。

2.1　UML 的扩展

目前 UML 扩展有两种方法[3]:一种是基于 MOF(Meta Object Facility)来进行扩展的方法,允许对元模型处理,包括增加和删除元类等;另一种是通过为模型元素添加一些构造型、标签值和约束来扩展 UML 的建模,而不对一个已有的元模型进行修改。

UML 的扩展机制包括构造型、标记值和约束,文献[4]中对三种扩展机制给出了具体的描述,它们提供增加新构造块、创建新特性和详述新语义的机制。这三个扩展机制允许被修改和完善 UML 以满足需要,我们所要完成的是在较早的阶段将用户提出的需求分离出核心组件和横切关注点,然后集成到系统中。

2.2　UML 面向方面建模

面向方面建模技术[5,6]允许在系统设计时模块化方面,从核心功能性需求中分离出不同的关注,例如异常处理、日志、实时性、安全性、资源共享等[7],并且使用织入机制,将方面织入到核心功能模块中来实现系统的集成。

当前对面向方面建模的研究存在两个方向:

第一是从软件体系结构的角度研究面向方面建模,就是将面向方面编程思想和概念引入到软件体系结构中,从而实现在软件体系结构高度上对系统进行关注点分离。

第二是研究如何应用 UML 进行面向方面建模,即使得 UML 支持面向方面建模。具体方法是利用 UML 所提供的扩展机制,将面向方面编程中的一些概念,例如 Aspect 作为与类不同的实体引入到 UML 视图中。

2.3　UML 状态图模型

状态图[8,9]描述一个实体基于事件反应的动态行为,由状态、转换、事件和活动组成,它强调一个对象按事件次序发生的行为。状态图中,当满足条件时事件被触发,从一个状态迁移到另一个状态,迁移的过程就是动作的发生。

状态图描述对象和方面行为具有以下优点:首先,提供丰富的语义表达横切行为;其次,假定系统是一个有限的状态系列,这样就降低了系统的复杂性;最后,状态图模型具备完整的行为规范,保留了设计和实现之间迁移,使自动产生代码成为可能。

从以上优点可知,状态图具有描述方面的横切行为的能力,并能实现横切关注隐式的织入到系统核心功能模块。

状态图能够很好地建模面向对象中对象内部行为,但是并不直接支持面向方面对象的行为建模。因此,必须对状态图实行扩展,增加表达方面的机制来建模方面行为,而不改变状态图基本原理,从而实现状态图支持面向方面建模。

普通状态图中,事件、方法调用或计时器时间的终结可以触发状态的迁移。而面向方面状态图的方面建模应该更多地考虑方面与方法、迁移的关联,而不是状态。

这里,层次状态图则清楚地表示每个对象的状态

图以及方面的织入过程。每个区域之间的交互可以通过共享变量、消息传递机制或者其它区域的状态变化等来实现。

3 面向方面状态模型及应用实现

3.1 面向方面状态模型用于仓库管理

仓库管理系统是针对仓库材料的出入库登记和查询统计等方面的管理软件，以入库管理和出库管理为重点，实现出入库、查询及用户管理等功能。

仓库管理系统是面向方面技术很好的应用场所，因为在仓库管理系统中有很多非功能需求，且这些非功能需求往往横切整个系统，典型的就是合法性的验证以及日志的记录。

传统的基于设计技术的状态图将这些关注点混合起来，即使授权和验证行为散布在整个系统的功能部件里，带来了代码混乱和代码分散问题。一个较好的解决办法是抽象出核心功能和分离关注点，最后通过织入的方式形成最终代码，这就是 AOP。下面介绍详细的解决过程：

用户要进入系统时，必须输入用户名和密码完成身份的验证。同时，校验（抽取出来的方面）过程又横切核心代码，来验证账户的合法性：如果成功，则进入系统并允许其他操作。当判断出现货少于最低储备时，操作员会制定采购计划。在这个处理过程中，系统会保存所有操作记录；如果无效，系统则会显示错误信息并拒绝相应的操作（见图 3）。

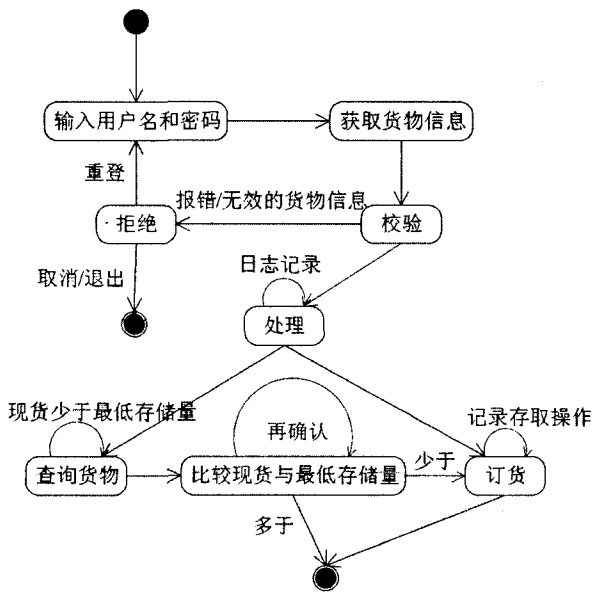

图 3 仓库管理系统终端操作的面向方面状态模型

模型表明，货物入仓的处理模块应当有日志记录以便于查询。而且，日志模块不应该和其他功能模块交织在同一代码块。事实上，校验操作也属于横切关注点，因为像日志记录操作一样，校验操作也分布于各个功能模块。

3.2 仓库模型的 UML 扩展描述

扩展 UML 建模来方面，是通过引入《aspect》这个构造型来表达方面的[10]。《aspect》构造型用于建模横切关注单元，确保《aspect》具有类实例相同的行为方式。

类、类元与构造型《aspect》的关系在 AOSD 框架中如图 4 所示。

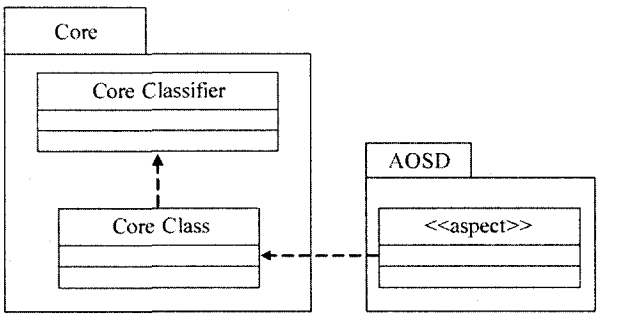

图 4 类、类元与《aspect》的关系

基于 UML 的面向方面的建模技术已经成为一个热门课题，其扩展机制更是为面向方面的描述提供便利，《aspect》、《pointcut》、《advice》等都可以通过 UML 扩展标记定义和描述。

图 5 表达了横切关注点的实现过程。

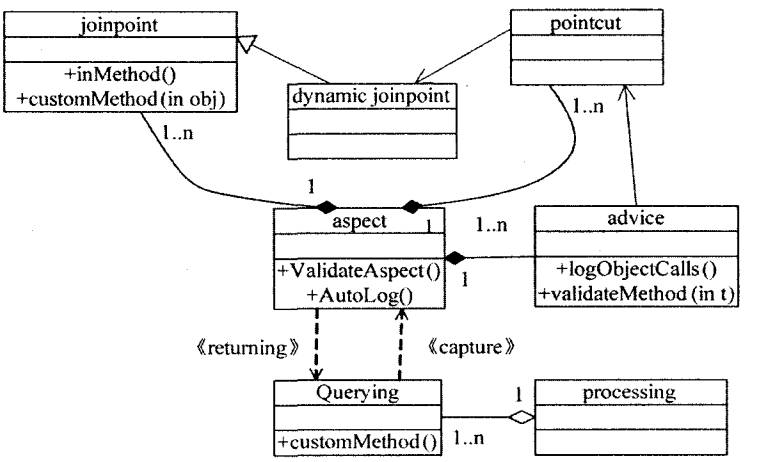

图 5 横切关注点的实现过程

接下来，将在 Eclipse 集成开发环境下，阐述基于 AspectJ 语言的横切关注点的面向方面代码实现。

3.3 面向方面的代码实现

接下来我们将列出仓库管理应用中相关的面向方面的实现[11]（伪码形式）。

首先，验证用户的认证和授权，这个不是核心功能模块所需要关注的，而是交给 AOP 去处理，代码如下：

```
public aspect ValidateAspect {
pointcut validateMethod( enterFrame t) :
call ( public void enterFrame. inmethod ) &&target
```

```
(t);
    before( enterFrame t) : validateMethod( t) {
    //query database to find whether there has the
    //corresponding username and password
    //if invalid, rejecting
    if( valid) {
    if( enterFrame. namestr. equals( "operator" ) ) {
    //setting the authorization the operator should have;
    //setting the authorization the other users
    //such as manager etc. should have} } } }
```

如果用户是合法的操作员,系统将授权相关的操作,在这期间,系统必须记录下每个操作员的操作历史,代码如下:

```
    public aspect AutoLog {
    pointcut customMethod( Object obj) {
    pointcut publicMethods( ) :
    customMethod( obj ) &&execution ( public  * org. apache. * ( . . . ) ) ;
    pointcut logObjectCalls( ) : execution ( * Logger. * ( . . ) ) ;
    pointcut loggableCalls:
    public Method&&! logObjectCall( ) {
    before( ) : loggableCall ( ) { thisJoinPoint. getSignature. toString( ) ;
    }
    after( ) : loggableCall( ) { thisJoinPoint. getSigure. toString( ) ; } } } }
```

核心组件模块发挥其功能上的作用,同时,ValidateAspect 和 AutoLog 两个 aspect 自动捕捉连接点,核心组件和核心关注点各执其能,从而确保整个模块完成设计者所要实现的功效。

在这个范例中,当用户验证需求发生变化时,并不需要重新研究和修改每一个新的模块以确保遵守现有的规则,只需根据具体需求,分别更新功能模块和分离的关注点。

4　结束语

AOP 是在 OOP 基础上提出的,并被用来解决代码纠缠问题。文中首先分析了已有的、用于描述面向方面概念和结构的方法,然后通过对 UML 状态模型进行扩展,获得面向方面的状态模型,并给出了基于 AspectJ 语言的面向方面代码实现。该模型易于理解,容易接受,更重要的是促使了 AOP 在更广泛的范围内被使用,从而有效地提高了所开发软件的可重用性、可维护性等。

然而,面向方面建模研究仍面临挑战,也是将来的研究重点,如:在 UML 扩展机制下,给出更多支持面向方面的软件模型及其代码的自动化生成;针对面向方面的软件模型给出相应的面向方面测试模型等。

参考文献:

[1] 李志纯,张南平. 面向 Aspect 编程的应用研究[J]. 计算机技术与发展,2006,16(5):217-218.

[2] Booch G, Rumbaugh J, Jacobson I. Tile Unifield Modeling Language User Guide[M]. [s. l.]:Addison-Wesley,2001.

[3] Eriksson H E, Penker M Lyons B, et al. UML2.0 工具箱[M]. 余安萍, 译. 北京: 电子工业出版社, 2004:258-297.

[4] 杨敬中, 张广泉. 基于 UML2.0 的面向方面建模方法研究[J]. 江苏大学学报,2007(1):21-22.

[5] Schauerhuber A, Schwinger W. A Survey on Aspect-Oriented Modeling Approaches[R]. Technical Report AOSD-Europe-ULANC-9, AOSD-Europe,2005.

[6] 李　婷,刘建勋,尹雁青. 面向方面建模方法的研究及其应用[J]. 计算机技术与发展,2009,19(1):113-115.

[7] Zakaria A A, Hosny H, Zeid A. A UML Extension for Modeling Aspect-Oriented Systems[C]//In:Intl. Workshop on Aspect-Oriented Modeling with UML. Germany:[s. n.],2002.

[8] Naveed M, Abdullah M K, Rashid K, et al. Representing Shared Join Points with State Charts: A High Level Design Approach[J]. Transactions on Engineering, Computing and Technology,2006,15:82-83.

[9] 王　斌,刘　菲,桂卫华,等. 一种基于 CSP 的面向方面状态图形式化描述方法[J]. 计算机工程与科学,2008(5):41-42.

[10] 刘瑞成, 张立臣. 基于 UML 的面向方面建模方法[J]. 计算机科学, 2005(10):205-206.

[11] Zhang Jingjun, Chen Yuejuan, Liu Guangyuan. The Realization of Aspectual State Model with UML Extension Mechanism [C]//4th IEEE Conference on. Industrial Electronics and Applications, ICIEA 2009. [s. l.]: [s. n.], 2009: 3068-3072.

[12] AspectJ. AspectJ Project Home Page[EB/OL]. 2009. http://eclipse. org/aspectj/.