

文章编号: 1005—0523(2004)01—0017—04

面向方面程序设计概述

黄晓生

(华东交通大学 土建学院, 江西 南昌 330013)

摘要: 软件工程的一条重要原则是关注点分离, 现在大多数的软件项目都选择 OOP 的编程方式, 但 OOP 技术也有其局限性, 它对有些需求并不能很好的进行描述. 介绍了一种全新的编程范型——面向方面的程序设计, 这种编程技术可以有效地描述 OOP 方法不能描述的关注点; 介绍了面向方面程序设计的基本思想以及面向方面程序设计设计语言 AspectJ 的基本概念.

关键词: 关注点; 面向对象程序设计; 面向方面程序设计; 编程范型

中图分类号: TU991.35 **文献标识码:** A

1 引言

软件开发从本质上讲就是对软件所要处理的问题域进行正确的认识, 并把这种认识正确的描述出来, 既然如此, 那就应该直接面对问题域中客观存在的事物来进行软件开发, 这就是面向对象. 另一方面, 人类在认识世界的历史长河中形成的普遍有效的思维方法, 在软件开发中也应该是适用的. 人们在日常生活中习惯的思维方式和表达方式, 也应在软件开发中尽量采用, 这就是面向对象方法所强调的基本原理^[5].

事实上, 现在大多数的软件项目都选择 OOP 的编程方式, 因为基本的对象模型能较好地描述实际的问题. 面向对象技术很好地解决了软件系统中角色划分的问题. 借助于面向对象的分析、设计和实现技术, 开发者可以将问题域的“名词”转换成软件系统中的对象, 从而很自然地完成从问题到软件的转换.

但是, 问题域的某些需求却偏偏不是用这样的“名词”来描述的. 人们认识到, 传统的程序经常表现出一些不能自然地适合单个程序模块或者几个紧密相关的模块的行为. 例如日志记录, 对上下文

敏感的错误处理、性能优化以及设计模式等等. 我们将这种行为称为“横切关注点 (crosscutting concerns)”, 因为它跨越了给定编程模型中的典型职责界限.

虽然横切关注点会跨越多个模块, 但当前的技术倾向于使用一维的方法来处理这种需求, 把对应需求的实现强行限制在一维的空间里. 这个一维空间就是核心模块级的实现, 其他需求的实现被子嵌入在这个占统治地位的空间. 换言之, 需求空间是一个 n 维空间, 而实现空间是一个一维空间, 这种不匹配导致了糟糕的需求到实现的映射^[6].

为此, 施乐公司帕洛阿尔托研究中心 (Xerox Palo Alto Research Center) 的 Gregor Kiczales 等在实践的基础上于 1997 年提出了一个全新的编程技术——面向方面程序设计 (Aspect Oriented Programming, AOP), 并开发了第一个 AOP 开发环境——AspectJ^[1]. AOP 为开发者提供了一种描述横切关注点的机制, 并能够自动将横切关注点织入到面向对象的软件系统中, 从而实现了横切关注点的模块化. 通过划分 AspectJ 代码, 横切关注点变得容易处理. 开发者可以在编译时更改、插入或删除系统的 Aspect, 甚至重用系统的 Aspect.

2 AOP 基础

2.1 横切关注点 (crosscutting concerns)

一个关注点 (Concern) 就是一个特定的目的, 一块我们感兴趣的区域, 从技术的角度来说, 一个软件系统会包含一些核心的关注点和系统级的关注点. 例如, 一个个信用卡处理系统的核心关注点是借贷/存入处理, 而系统级的关注点则是日志、事务完整性、授权、安全性及性能问题. 很多系统级关注点一般来说是相互独立的, 但它们一般都会横切许多核心模块. 一个典型的企业应用会包含许多横切关注点, 如验证、日志、资源池、系统管理、性能及存储管理等, 每一个关注点都牵涉到几个子系统, 如存储管理关注点会影响到所有的有状态业务对象.

Tzila Elrad 等在^[2]所描述的图形编辑器的 UML 图中, 有两个关注点: 跟踪每个图形元素位置的数据关注点 (data concern) 和图形元素移动时更新显示的特征关注点 (feature concern). 在 OO 设计中, 很好地分解了数据关注点, 然而特征关注点却横切了数据关注点: 软件可以围绕特征关注点来设计, 但是数据关注点又将横切特征关注点. 总之, 不论怎么设计, 在这个图形编辑器中, 这两个关注点总是会相互横切 (如图 1 所示).

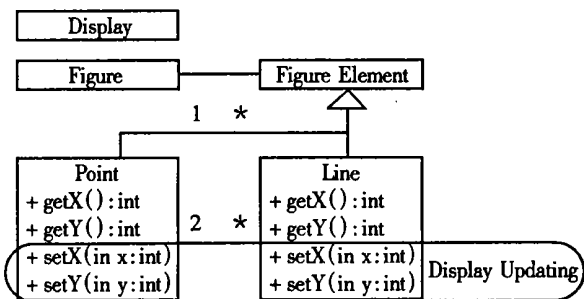


图 1 一个简单图形编辑器的 UML 图

2.2 什么是 AOP

AOP 是由 Xerox PARC 开发的一种新的编程范型, 通过 AOP, 开发人员可以对前述的横切关注点进行分离, 另外它还提供了一种可以明确捕获横切关注点的语言机制, 这使得它可以将横切关注点以模块化的形式来实现, 而且通常还使得代码更加简化更易于维护和更易于重用, 这些都是通过使用方面来实现的. OOP 所做的是对对象进行封装和继承, 而 AOP 则对横切点进行封装. 所谓方面就是横切其他单元结构的模块单元. 一个方面类似于 OOP 中的类, 它能继承类或其他方面, 也可以抽象化或

具体化或具有一定的域, 方法及类型, 它将会影响多个类的行为封装成可重用的模块^[3].

2.3 方面编织器 (Aspect Weaver)

编译由 AOP 语言开发的程序与通常的程序不太一样, AOP 首先通过方面编织器将代码和方面编织在一起然后再将程序编译成可执行文件, 如图 2 将显示了传统的程序编译过程和 AOP 的编译过程的不同^[4].

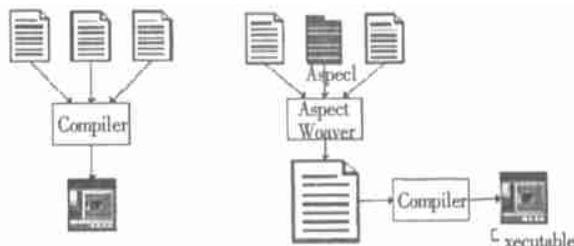


图 2 与传统方法相比, AOP 允许在程序编译成可执行文件悠扬前通过方面纺织器将代码及方面纺织在一起. 左图为传统编译方法示意图, 右图为 AOP 程序的编译过程示意图.

3 AOP 语言基础^[2,3,7,8]

面向方面程序设计 (Aspect-Oriented Programming, AOP) 是一个令人兴奋的新模式. 就开发软件系统而言, 它的影响力将会和有 15 到 20 年的面向对象技术一样^[1]. 但是作为一种新的技术, 它提出了许多全新的概念, 在 OOP 的工具里是继承, 封装和多态, 而 AOP 语言的组件是连接点、通知、导言、元数据、切入点 and 方面.

3.1 连接点 (Join Point)

连接点是 AspectJ 中的一个重要概念, 它是程序执行过程中明确定义的点. 连接点可能定义在方法调用、条件检测、循环的开始或是赋值动作处. 连接点有一个与之相关联的上下文. 比如, 一个方法调用连接点的上下文可能会包含一个目标对象及调用参数等.

3.2 通知 (Advice)

一个通知是一个逻辑, 这个逻辑由特定的事件触发; 它是行为, 这个行为能够被插入在调用者和被调用者之间, 在一个方法调用者和实际的方法之间, 通知是 AOP 真正的关键.

3.3 导言 (Introduce)

导言是一个增加方法或者域到一个存在的类中的途径. 它们甚至允许你改变当前存在的类的显式接口, 并且引入一个混合的类, 这个类实现了新

的接口. 导言允许你将多继承引入到一般的 Java 类. 导言一个主要的用例是当有一个方面, 并想让这个方面有一个运行时间接口, 跨越不同的对象层次时, 仍然要让应用开发者能够调用特定方面的 APIs. 导言能够是一个方法, 它将一个新的 API 绑定到一个存在的对象模型.

3.4 元数据(meta data)

元数据是能够在静态或者运行时间绑定到一个类的附加信息, 更强大的一方面是, 能够动态绑定元数据到一个给定的对象实例. 当真正编写应用于任何对象的一般方面, 而逻辑需要知道制定类的信息时, 元数据是非常强大的. 一个好的元数据类比如就是 EJB 规范.

3.5 切入点(Pointcut)

如果通知(Advice), 导言和元数据是 AOP 的特征, 那么 pointcuts 就是粘合剂. Pointcuts 告诉 AOP 框架, 哪些通知(Advice)绑定到哪些类, 什么元数据将应用于哪些类或者哪一个导言将被传入哪些类. Pointcuts 定义各种 AOP 特征将怎样应用于你应用中的类.

3.6 方面(Aspect)

方面是 AspectJ 的模块单元, 其地位就像是 Java 里的类. 它把切入点和通知包在一起. 和类相似, 方面也可以包含方法和属性、从其它类或方面扩展以及实现接口等. 与类不同的是, 不能用 new 来建立一个方面实例.

AspectJ 允许在类中声明切入点, 但在类中只能声明 static 的切入点. 而且 AspectJ 不允许类里包含通知, 只有方面可以包含通知.

4 面向方面程序设计的步骤^[6]

AOP 的开发包括三个清晰的开发步骤:

1) 方面分解: 分解需求提取出横切关注点和一般关注点. 在这一步里, 你把核心模块级关注点和系统级的横切关注点分离开来. 就前面提到的信用卡例子来说, 可以分解出三个关注点: 核心的信用卡处理、日志和验证.

2) 关注点实现: 各自独立的实现这些关注点, 如上面的信用卡例子, 要实现信用卡处理单元、日志单元和验证单元.

3) 方面的重新组合: 在这一步里, 方面集成器通过创建一个模块单元——方面来指定重组的规则, 重组过程——也叫编织(weaving)或集成(inte-

grating)——则使用这些信息来构建最终的系统.

5 讨论^[1,3,6]

5.1 使用 AOP 带来的好处

1) 由于 AOP 的出现就是为解决横切关注点问题, 因此, 使用 AOP 首先带来的好处就是可以模块化横切关注点的实现, 由此也使得代码有良好的可重用性.

2) 使用 AOP 可以解决代码混乱和代码分散问题, 这使得采用 AOP 实现的系统代码冗余小, 也易于理解和维护.

3) 由于方面模块根本不知道横切关注点, 所以很容易通过建立新的方面加入新的功能. 另外当你往系统中加入新的模块时, 已有的方面自动横切进来, 使系统易于扩展.

5.2 AOP 的不足

1) AOP 是一种新的编程技术, 没有完整的文档, 没有得到良好的测试和大量的实际项目的应用, 作为一种新技术, 还需要在实际工作中得到更多的检验.

2) 目前, AOP 在开发工具上还很缺乏, 目前主要是基于 JAVA 上的 AspectJ, 而基于 C# 和 .NET 以及其他语言上的开发工具都尚在研究中. 关于什么是 AOP 分析、设计过程, 设计和分解方面的基本原则, 如何调试, 什么是良好的方面模块等都有待进一步研究.

3) AOP 的理论支持方面也有待进一步研究. 如用什么理论能最好地描述方面与组件之间的交互等.

6 结束语

AOP 作为一种编程技术, 仍是一个新的概念. 它是一种对代码进行再分析、再处理而解决横切关注点问题的技术, 其提出不是一种 OOP 的替代, 而是对 OOP 的扩充. 虽然有越来越多的人开始关心 AOP, 使用 AspectJ, 但作为一种新思想, 要成为一个主流的编程方法还有很长的路要走.

参考文献:

- [1] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopez, John-Marc Loingtier and John Irwin. Aspect-Oriented Programming [J]., pp. 220 ff. in Proc.

11th European Conference on Object-Oriented Programming (Jyv? skyl?, Finland, 9. 13 June 1997) . published by Springer-Verlag as Lecture Notes in Computer Science no. 1241 (Mehmet Ak? it and Satoshi Matsuoka, editors)

[2] Tzilla Elrad, Mehmet Aksits, Gregor Kiczales, Karl Lieberherr, and Harold Ossher, “Discussing Aspects of AOP”[J]. Communications of the ACM, vol. 44, no. 10, pp 33—38, October 2001.

[3] Niklas P ahlsson, An Introduction to Aspect — Oriented Programming and AspectJ, Topic Report for Software Engineering 2002—11—03.

[4] Ken Wing Kuen Lee An Introduction to Aspect — Oriented Programming, Reading Assignment.

[5] 邵维忠, 杨芙清. 面向对象的系统分析[M]. 北京: 清华大学出版社, 广本西科学技术出版社, 1998.

[6] Ramnivas Laddad, 刘克科, 编译. 利用 AOP 分离软件关注点[J]. 程序员 2002 合订本下: pp167—170.

[7] Ramnivas Laddad, 刘克科, 编译. 通过 AspectJ 更好地了解 AOP[J]. 程序员 2002 合订本下: pp170—172.

[8] <http://Tech.ccidnet.com>

A Brief Introduction to Aspect Oriented Programming

HUANG Xiao-sheng

(School of Civil Engineering and Architecture, East China Jioatong University, Nanchang 330013, China)

Abstract: Separation of concerns is an important software engineering principle. The leading programming paradigm used today is object oriented programming (OOP), yet OOP has its limitations, some design decisions can not be illustrated with the OOP. This paper introduces an new programming paradigm called aspect oriented programming—a programming technique that make it possible to express those programs that OOP fail to support. Also, AOP fundamentals and the basic concept of Also, AOP language AspectJ are introduced in the paper.

Key words: concerns; OOP; AOP; programming paradigm

(上接第 11 页)

[2] 天津市标准. 大开间住宅结构钢筋混凝土异形柱框轻结构技术规程(DB29—16—98). 1998.

[3] 上海市标准. 钢筋混凝土异形柱结构设计规程(试行)(DG/TJ08—009A—2001). 2001.

[4] 过镇海. 钢筋混凝土原理[M]. 北京: 清华大学出版社, 1999.

Experimental and Theoretical Research on Load-bearing Capacity of Z-shaped R·C· Columns

XU Hai-yan, XUE Hai-hong, YUAN Zhi-hua

(School of Civil Eng. and Arc., East China Jiaotong University, Nanchang 330013, China)

Abstract: Based on the experimental study of 9 R·C short columns with “Z” shaped section subjected to axial compression and cyclically biaxial bending, this paper reveals the failure pattern and general regularity for load-bearing capacity of the columns. The theoretical analyses result using the digital iteration method is proposed.

Key words: Z shaped R·C columns; axial compression; biaxial bending; bearing capacity