一种面向Aspect的软件分析设计方法

张俐佳，唐雪飞，汪林川

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】基于在软件开发过程中缺乏统一的分离Aspect的标准。本文提出了一种面向Aspect的分析设计方法，采用Theme/Doc与定制UML Profile设计建模相结合，能够清晰地从需求分析中提取Aspect，并能直接从建模元素映射代码，使Aspect从需求到实现的整个开发过程中具有一致性。

关 键 词 一致性； 统一模型语言； 分析设计方法
中图分类号 TP311 文献标识码 A

An Aspect-Oriented Analysis and Design Method

ZHANG Li-jia，TANG Xue-fei，WANG Lin-chuan

(School of Computer Science and Engineering, Univ. of Electron. Sci. & Tech. of China Chengdu 610054)

Abstract The goal of the aspect oriented development is to extract Aspect thought the whole software development process, and make it consistent though the transformation. However there is no standard method for AOSD that covers the software development from requirements to design activities. In this context, a new method is proposed to extract Aspect. It combines Theme/Doc and customized UML Profile design method, so that the Aspect can be explicitly extracted from the analysis process and mapped from model element to code directly. And the consistence of Aspect can be archived.

Key words consistence; unified modeling language; analyze and design method

文献[1]提出面向Aspect的概念，用于解决面向对象程序设计中出现的代码冗余和代码混乱(Tangling)。与面向对象一样，面向方面(Aspect)的思想被应用于软件开发周期的各个阶段，然而针对面向Aspect的从分析到设计整体过程的方法并不多。文献[2]提出主题(Theme)方法，并结合主题文档(Theme/Doc)[3]以及组合设计模式(Composition Pattern)[4]，但建模模型中的Aspect在转化为程序时不能直接映射。

本文改进了Theme方法，采用Theme/Doc与轻量级扩展统一建模语言(UML)[5]方式结合。在设计过程中，使Aspect能够从设计模型直接映射为实现代码。此方法在从分析模型到设计模型的转换过程中也保持了Aspect的一致性。

1 扩展UML设计模型与面向Aspect编程范型的一致性

在建立建模元素时，应该考虑如何与程序设计范型相一致。程序设计范型，决定了程序员以一种什么样的视角来看待程序。正如面向对象的程序员将程序看作对象之间的交互、面向Aspect的程序员将程序看作Aspect与对象、Aspect与Aspect的交互。要使设计模型能够直接映射为程序的实现，就必须使建模模型元素与程序设计的范型相统一。

按照这一原则，本文在扩展建模元素的设置中，沿用了文献[5]中的联接点(Pointcut)、通知(Advice)、方面(Aspect)模型元素，改变了Aspect内部元素的关系，同时增加了Aspect之间的关联关系，使其更符合设计范型的需求。

1.1 Aspect及其内部建模元素

Aspect及其内部元素对应的建模元素如表1所示。Aspect封装了横切的功能和横切的属性，正如类封装了模块的功能和模块的属性。因而Aspect与类都是一种实体，它们是程序设计中的同阶元素。与面向对象的建模将类作为建模的基础元模型一样，面向Aspect的建模也应将Aspect作为独立的模型元素。因此Aspect被表示为类的原型<<aspect>>，在类图中Aspect与类为同价建模元素。

收稿日期：2004 - 09 - 21
作者简介：张俐佳(1980 –)，女，硕士，主要从事中间件以及Aspect技术方面的研究.

Pointcut表明了Aspect作用的位置，定义了Aspect的状态，应该将其作为Aspect的属性。Pointcut包含对代码、类型、对象和控制流的绑定，由一组joinpoint运算而成，可以由类的原型<<Pointcut>>表示，<<Pointcut>>中的属性以｜｜或＆＆符号相隔离，表明了joinpoint的组合。

Advice用于声明在Pointcut表达式中定义的joinpoint被调用时应执行的操作。Advice作为Aspect中提供的服务的实现，可以被Pointcut所激发，而影响Pointcut所绑定的上下文的行为。因此，Advice应该作为Aspect中的操作，同时Pointcut可以作为Advice的一个参数表明Advice应执行的位置。Advice分为在Pointcut前执行、在Pointcut后执行和在Pointcut 处执行，使Advice作为Operation的原型可分为<<after>>、<<before>>和<<around>>三种。

表1　**Aspect**及其内部组成的建模元素

基础元模型	原型	语义	约束
类	<<aspect>>	表达横切的实例，用于建模横切单元。	Aspect可以与其他类或其他的Aspect有关联，关系Aspect包含几个advice和多个Pointcut。
类	<<Pointcut>>	表达了advice操作的绑定位置。	可以与其他多个类或Aspect动态绑定，但只能包含于Aspect中，是Aspect的一个属性。
操作	<<after>> <<before>> <<around>>	表达对Pointcut绑定上下文下所作的操作。	与Pointcut绑定，只能包含于Aspect中，是Aspect的一个操作。

1.2　**Aspect**与类、**Aspect**与**Aspect**之间关系的建模元素

Aspect与类、Aspect与Aspect之间的关系，表现了程序设计中Aspect与对象之间，以及Aspect与Aspect之间的交互作用。因此在“方面”建模时需要支持这种关系的建模元素。

1.2.1　Aspect与类的关系

Aspect与类之间的关系，表达了Aspect对类的横切。通过这种关系，方面可以在功能模块不变的情况下，扩展整个软件的安全性和及时性。

表2　**Aspect**与类关系建模元素

基础原模型	标签值	语义	对类的影响
关联	{替换}	当包含这种关系时，aspect必须包含<<around>>advice。	取代类的行为
关联	{跟踪}	当包含这种关系时，aspect必须包含<<after>>或<<before>>advice。	跟踪类的行为
关联	{同步}	当包含这种关系时，aspect必须包含<<after>>或<<before>>advice。	改变类的功能

1.2.2　Aspect与Aspect的关系

当多个Aspect形成一个软件体系时，Aspect之间同样存在横切，而这种横切可以用Aspect自身来表示，因此形成了Aspect横切的Aspect。目前一些支持动态方面的系统如JBoss AOP，并不支持这种特性，而一些静态的方面语言如AspectJ却支持这一特性[6]。因此，在描述Aspect与Aspect之间的关系时，可以包含Aspect与类之间的所有关系包括{替换}、{跟踪}、{同步}，以及它们的语义。除此之外，方面还有继承关系，用{泛化}表示，以及优先级关系{优先}。

这种UML扩展建模方式，符合AspectJ的程序设计规范，因此可以方便地将建模模型转化成为代码。下面，以这种设计方法与分析方法相结合，作进一步的说明。

2　面向**Aspect**软件开发的方法

面向Aspect软件开发的方法结合Theme/Doc与扩展UML建模方式，分为识别方面、细化方面和方面建模三个阶段，对应分析图、细化图和类图。这种方法在细化方面的过程中，将Aspect作为与类同等的元素，并且在最后转换为建模元素时，也与面向Aspect的编程范型相一致。

2.1　方法简述

(1) 第一阶段：识别方面。识别方面的目的是从需求中分离出横切行为，输出第一阶段结果的分析图。在本阶段完全采用文献[3]中的Theme/ Doc方法。方法具体步骤见文献[3]。确定方面的关键是区分两个行为的关系是否为横切关系，但文献[4]中对此并

没有明确的标准。本文提出的横切的关系可以描述为：(1) 当两个行为的关系是横切的，记横切行为为b，普通行为为a，则b的功能可以在a中执行，而a可以完成其主要目标而不依赖于b；(2) b的功能并不单单针对a，还可以适用于其他行为。这样确定了横切的行为后，Theme/Doc方法就已经确定了哪些需求包含Aspect。同时，Aspect与功能模块的连接关系，也可以从横切行为与普通行为之间的连接表示出来。

(2) 第二阶段：细化“方面”。细化“方面”是从分析图中细化出Aspect所包含元素Pointcut的位置，以及决定Aspect的影响作用于这个位置之前、之后或这个位置本身，它的输出为细化图。由于Pointcut是附着在实体上的，而分析图中并不包含实体的概念，因此必须从中分离出这些实体以及这些实体的操作。细化“方面”的方法对包含横切行为的需求进行词法分析，从中分离出类和类所包含的操作。在分析时遇到Aspect行为则需要将Aspect当成一个实体对待。在细化的过程中，由于需求中所有可能的操作都被分析出来，使被横切的操作被清晰地表示，因而Aspect的Pointcut属性获取了具体的值。此时也可以决定Aspect中的Advice是在Pointcut之前、之后或就在Pointcut位置本身的作用。

(3) 第三阶段：“方面”建模。“方面”建模通过对细化图进行分析，将其中的Aspect实体映射为Aspect建模元素，从而形成类层的建模模型。在经过第二阶段的分析以后，“方面”建模中所应该包含的组成部分已经可以从细化图中直接转换得到，它们包括类、Aspect和Pointcut。而Aspect之中的Advice的类型，也可以从中得到；Advice的具体功能则需要分析获取。通过Pointcut的位置和Advice的作用，可以得知究竟哪两个实体之间存在横切关系，以及它们的横切关系属于哪一类。

经过这样的分析过程，程序设计中的Aspect所需要描述的内容已经具备，可直接映射到实现。整个分析过程能够实现问题空间、设计空间和程序空间之间的直接映射，并能更容易地控制需求变化所导致的在设计空间和程序空间的“涟漪”效应[7]。下面通过一个实例，具体阐述面向Aspect软件开发方法的应用。

2.2 实例分析

使用文献[3]中的与地点相关的游戏的需求实例来说明面向Aspect软件开发的方法。整个实例包含的59个需求，可以由Theme方法转化为分析模型，具体步骤见文献[3]。为了简化问题，本文只列出几个需求。

(1) 需求R_1。当玩家(Player)遇到同类型角色(Charactor)时，角色会给与(Give)玩家宝石(Crystal)。

(2) 需求R_2。当玩家与另一个玩家决斗(Duel)失败(Lose)时，它将给出(Give)它所有的赌注宝石。

(3) 需求R_3。当玩家完成(Complete)角色交予的所有任务时，角色会给(Give)它一个宝石。

通过Theme/Doc方法，可以得到如图1所示的分析图，(图中菱形表示行为；椭圆表示需求；弧线表示行为之间的横切关系)。give作为Aspect横切了几个行为。对R_1、R_2和R_3进行词法分析，分析出其中的实体(Player、Character)以及具体的行为。在细化方面的过程中，将give作为一个实体对待。分析exact diagram如下(+、－或无符号分别表示advice的类型为<<before>>、<<after>>或<<around>>)。

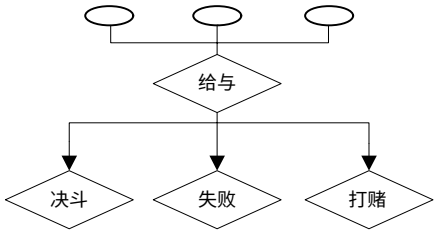

图1 分析图

从图2可以看出，give横切了player的lose()、meet()以及Character的complete()方法。这些行为将被视为<<pointcut>>的实例cut_1的属性值，而cut_1作为give<<aspect>>的属性包含在其中。因此，give的Advice应该为<<after>>，它包含的操作give()的功能是从给与类到被给与对象crystal传递的操作。由于give 改变了类player的操作，因此give与player的关系为{同步}；同样地，give与Character的关系为{同步}。用扩展UML建模类图表达，如图3所示。图中菱形表示<<aspect>>，椭圆表示<<point cut>>。

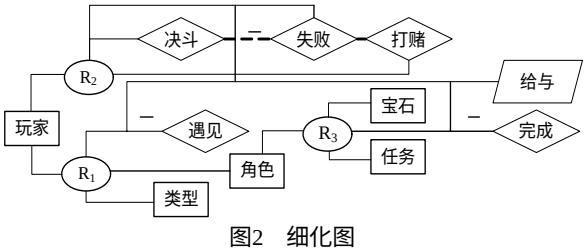

图2 细化图

从类图中可以直接得到aspect这个程序设计元素，以及它的内部元素和内部元素的具体取值。分析模型中的give也就唯一地映射为了AspectJ中的aspect give。

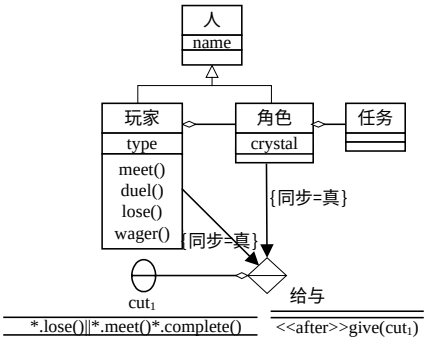

图3　类图

4　结 束 语

面向Aspect的分析设计尚处于研究阶段，没有形成一个特定的标准。本文提出的将需求分析模型Theme/Doc与定制UML Profile相结合的方法，在分析图、细化图以及类图中，可以清晰地分辨出Aspect，以及Aspect与类的关系。该方法提供了一种完整地分析Aspect的软件开发方法。同时，由于该方法使Aspect的转换过程能够直接映射，因此由需求改变所导致的Aspect的增加或减少将不阻碍转化在分析、设计和实现中。这种采用在设计过程中将Aspect作为与类同阶元素对待的方式，使Aspect从分析空间、设计空间到实现空间实现了一致性，让开发人员能够有效地从需求到实现过程中追踪Aspect。

另外，该方法要求的需求语句粒度小，因此在需求工程中要作大量的工作，将这种方法与一种粗粒度的需求分析方法(如基于视角分析模型的需求分析方法)相结合，能够更方便地提取分析阶段所需的需求语句，这是下一步的工作。

参 考 文 献

[1] KICZLES G, LAMPING J, MENDHEKAR A, et al. Aspect-oriented programming [C]. In: Proceedings of ECOOP. LNCS 1241, Springer-Verlag, Berlin, 1997.

[2] BANIASSAD E, CLARKE S. Theme: An approach for aspect-oriented analysis and design[C]. In: International Conference on Software Engineering (to appear). http:// www. cs. tcd. ie/ Elisa. Baniassad/theme. pdf; 2004-05-24.

[3] BANIASSAD E, CLARKE S. Finding aspects in requirements with theme/Doc[DB/OL]. Lancaster, UK Early Aspect 2004 http://trese. cs.utwente.nl/workshops/early-aspects-2004/ Papers/ Baniassad-Clarke. pdf, 2004-05-24.

[4] SIOBHAN CLARKE S, WALKER. ROBERT J. Composition patterns: An approach to designing reusable aspects [DB/OL]. Proceedings of the 23rd International Conference on Software Engineering, Washington. D. C. 2001.

[5] STEIN D, HANENBERG S, UNLAND P. A UML-based aspect-oriented design notation for Aspect[C]// In Proceedings of the 1st International Conference on AOSD. Enschede, Netherlands, 2002.

[6] GREENWOOD P，BLAIR L. Using dynamic aspect-oriented programming to implement an autonomic system [DB/OL]. http://aosd.net/2004/workshops/daw/Proc-2004-Dynamic-Aspects.pdf., 2004-05-24.

[7] 杨芙清, 梅　宏, 吕　建, 等. 浅论软件技术发展[J]. 电子学报, 2002, 30(12A): 1901-1906.

编　辑　熊思亮