面向方面的软件体系结构建模研究*

张广泉，杨敬中
(重庆师范大学 数学与计算机科学学院，重庆 400047)

摘 要：分析了面向方面编程的核心思想及其概念，指出了编码阶段所存在的代码散射和缠结现象同样存在于软件体系结构中。在软件体系结构描述语言 XYZ/ADL 的基础上，通过增加一阶实体来表示 Aspect 概念，并对连接件作相应的修改，同时定义 Aspect 与组件之间的复合机制，提出了一种面向方面的软件体系结构建模语言 AO-ADL。

关键词：软件体系结构；面向方面建模；XYZ/ADL；AO-ADL
中图分类号：TP311 文献标识码：A 文章编号：1672-6693(2008)01-0058-06

面向对象编程(OOP)是一种在实际开发中得到广泛应用的编程技术，是软件开发史上最成功的编程技术之一。然而经过多年发展之后，通过大量的实践经验总结，人们发现 OOP 在软件开发的生命周期的多个阶段都存在一些不足之处：从需求到代码之间存在“跳跃”，程序编码阶段存在散射(Scattering)和缠结(Tangling)现象，系统很难以一种非侵入方式进行演化。

面向方面编程(AOP)[1]的提出成功地解决了面向对象编程中所存在的代码缠结和散射问题，使得软件系统以一种非侵入的方式进行演化成为可能。AOP 是对 OOP 的继承与发展，被认为是后面向对象时代一个重要的编程技术，将成为编程方法学上的一个新的里程碑。

随着面向方面编程技术的日趋完善，当前迫切需要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(AOSD)方法。面向方面建模(AOM)是 AOSD 的一个重要内容，其目的是采用一种较为普遍适用的设计方法来表达面向方面软件系统设计的特点和概念，从较高的抽象层次上对软件系统进行建模。然而，当前 AOM 并没有得到很好的支持，缺乏统一的、标准的建模理论，也没有得到相应建模工具的支持。事实上，这也是影响 AOP 技术推广的重要原因之一。基于此，提出一种从软件体系结构角度研究面向方面建模的思想和方法。

1 面向方面的软件建模方法概述

目前，对于 AOM 的研究还没有形成一个得到广泛认可的统一标准。考虑到当前面向方面软件开发尚处于技术启动阶段，多种面向方面编程语言相继出现，面向方面建模必将成为 AOSD 研究热点中的一个重要课题。随着 AOSD 研究的不断加深，当前对 AOM 的研究存在两种趋势：

第一种趋势是研究如何应用 UML 进行面向方面建模，其核心思想是：分析面向方面编程中的一些概念，应用 UML 所提供的扩展方法，将 Aspect 作为与类不同的实体引入到 UML 视图中，从而使得 UML 支持面向方面建模。扩展 UML 进行面向方面建模主要两种方式[2]：一种是所谓的“重型扩展”，实际上是基于 MOF 对 UML 进行扩展[3]；另一种是所谓的“轻型扩展”，实际上是基于 UML2.0 的 Profiles 机制进行扩展[4]。

第二种趋势是从软件体系结构的角度研究面向方面建模[5-8]，其核心思想是：将面向方面编程的思想和概念引入到软件体系结构中，以实现在软件体系结构层次上对系统进行关注点分离。当前，从软件体系结构的角度来研究面向方面建模尚处于探索阶段[9]，需要作进一步分析与研究。

笔者认为面向对象编程中所存在的代码散射和缠结现象是导致现有软件系统复杂性增加的核心因素，从软件体系结构的角度来看，也是一些负面现象。如果将 AOP 的相关概念引入到软件体系结构中，将 Aspec(方面)作为与组件、连接件等不同的一阶实体(First-class entity)，就可以在体系结构中分离出横切关注点，实现对横切关注点的模块化，避免

* 收稿日期：2007-05-06
资助项目：国家自然科学基金(No. 60073020)；重庆市自然科学基金(No. CSTC,2006BB2259)
作者简介：张广泉(1965-)，男，江苏连云港人，教授，博士，CCF 高级会员，研究方向为软件工程与形式化方法。

在软件体系结构中出现散射和缠结现象。

2 软件体系结构中的横切现象

在面向方面软件开发过程中,首先需要做的就是应用现有的面向方面需求工程技术对需求进行Aspect分解,以实现软件系统的关注点分离,由此可以得到两类关注点:核心关注点和横切关注点。当软件设计人员在对软件系统进行体系结构建模时,需要对系统进行功能分解,以实现将系统分解成一系列的组件。在理想情况下,系统的不同关注点应该清晰地分离到不同的组件中去,各个组件相互独立开发,然后将这些组件按照一定的原则组合在一起以实现一个完整的系统。因此,在软件开发过程中,在从软件体系结构角度对系统进行面向方面建模时,必须考虑以下两个部分。

(1)核心关注点。当前,几乎所有的组件模型都是从功能和服务角度对软件系统进行描述,采用的是"组件接口+组件实现"模式。由于核心关注点所表示的是软件要实现的主要功能和目标,因而可以很自然地将其表示为组件。

(2)横切关注点。横切关注点是那些与核心关注点之间有横切作用的关注点,例如系统日志、事务处理、权限验证。在当前的技术框架下,横切关注点在逻辑上与核心关注点相互之间彼此正交,但在实现上却趋向于与若干核心关注点交织。因而,很难将其表示为一般的组件。从传统的软件开发角度来看,通常采取的措施是在设计阶段将其忽略,在编码阶段将其实现,这使得程序编码人员必须重新设计系统的多个部分,以增加横切关注点,从而导致代码缠结和散射。由于当前AOP技术已经实现了在编码阶段对这些横切代码的封装,这就要求在进行面向方面建模时,需要将其表示出来,但是在当前的软件体系结构中并没有合适的元素可以用来表示系统的横切关注点。

在软件体系结构中,由于横切关注点与系统中的多个组件之间有横切关系,从而对这些组件产生一些负面影响,主要表现在以下几点:1)增加了组件之间的依赖性;2)分散了组件原来假定要做的事情;3)增加从体系结构到代码平滑过渡的难度;4)使得一些功能组件难以复用;5)源代码难以开发、理解和发展。

从软件开发角度来看,如果在系统开发的各个阶段都考虑到最后实现的结构,将有助于从早期设计到具体代码的映射。否则,在两者之间会存在断层。此外,在代码一级应用面向方面的优越性,也同样适用于早期的设计一级。因此,可以在软件体系结构层次对系统中的Aspect进行分离,以实现在体系结构设计和编码之间建立无缝的可追溯性。然而,在当前众多的软件体系结构描述语言ADL中,没有一种提供了可以用于描述Aspects的体系结构元素。因此,需要研究如何在现有的一种软件体系结构描述语言基础上进行扩展,以实现一种面向方面的软件体系结构描述语言AO-ADL。

3 软件体系结构描述语言XYZ/ADL

XYZ/ADL[10]是一种基于可执行时序逻辑语言XYZ/E的软件体系结构描述语言,它能在统一的时序逻辑框架中表示从形式规范到可执行程序的不同抽象层次的系统描述,既能表示软件体系结构的动态语义,又能表示其静态语义。它不仅适合于对软件体系结构进行形式化描述,也有利于对不同抽象级体系结构进行求精,而且还可以应用XYZ系统中提供的验证工具检验求精过程中的语义一致性。XYZ/ADL中的体系结构单元主要有:

(1)简单组件。简单组件分为接口和计算两个部分。简单组件分为接口和内部规范两个部分。在使用一个组件进行体系结构配置时,通过接口了解该组件"应如何使用"、"能做什么",因此接口包括一组说明组件与外部环境交互行为的端口描述和说明组件功能或相关性质的性质描述。内部规范描述组件的行为规范:通过刻画生成行为的数据变化过程来说明系统的行为是如何生成的,可以有不同的抽象程度,也可以为空。

每个端口是组件对外部环境的一个请求,或是能够为环境提供的一个服务,用XYZ/E的一组通道表示。语法为

%PORT PortName == ChannelType Declaration;[PortBehavior]

功能规范说明该组件"做什么",是一个时序逻辑公式,语法为

%FUNCTION == [Function Specification]

计算规范将各个端口连接为一个整体,是完整的组件行为描述,应满足性质描述部分所列出的性质,由一个XYZ/E单元表示,语法为

%COMPUTATION == [Computation Specification]

(2)连接件。连接件用于描述某类交互的共有

特性,它的实例可用于连接满足要求的不同组件。每个连接件分为接口和交互协议两个部分。接口由一组角色定义,表明参加该类交互的组件所应有的外部行为。交互协议描述如何将角色连接在一起产生交互,从而将参与交互的组件内部规范所表示的抽象行为进行组合。角色用通道表示,语法为

% ROLE RoleName = =

DataType Declaration ;[RoleBehavior]

交互协议由一个 XYZ/E 单元描述,语法为

% GLUE = = [Interact Protoco1]

(3) 复合组件。复合组件的接口描述与简单组件相同,而它的行为是通过几个子组件的连接来表示的。复合组件的组合声明指示它包含哪些组件和连接件实例,语法为

% COMPOSITION = =

[ComInsName : ComponentName ;

…

ConInsName : ConnectorName ;]

复合组件的粘接定义表明它的内部体系结构配置,语法为

% ATTACHMENTS = =

[ComInsName. PortName # ConInsName. RoleName ;

…

ComInsName. PortName ## PortName ;]

4 AO-ADL :基于 XYZ/ADL 的一种面向方面扩展

XYZ/ADL 是一种基于时序逻辑语言 XYZ/E 的软件体系结构描述语言,而 XYZ/E 是一种广谱语言,这种语言同时具有多种级别语言的特点,既可以用来描述规格说明,也可以用来编写可运行的程序代码,而且支持混合语言。因此,选择在 XYZ/ADL 基础上进行扩展,以实现一种面向方面的软件体系结构描述语言 AO-ADL,相对来说具有较好的实用性,有利于对面向方面的软件系统的体系结构进行描述、分析、求精、验证,直至最终系统的编译实现。

4.1 扩展方法分析

当前,对于如何将 AOP 概念引入到软件体系结构中,已经有了一些研究工作[11-12]。文献[13-14]试图应用现有的体系结构描述语言不作任何改变,直接表示面向方面概念。这样显然不能反映出实际的横切功能,也很难将具有横切功能的组件与一般组件相区分。针对这种情况,笔者选择在软件体系结构描述语言中引入一个新的体系结构元素,作为与 ADL 已有的组件、连接件等不同的一阶实体。此外,由于被引入的 Aspect 需要与组件进行复合,主要表现为 Aspect 的横切功能需要在组件中的某个位置发生,因而,还需要给出体系结构元素之间的复合规则。

4.2 AO-ADL 的语法与语义

在扩展过程中,笔者在 XYZ/ADL 中增加一个新的体系结构元素 Aspect,但 Aspect 本身并不是一个孤立的实体,它需要对那些与之有横切关系的组件产生横切作用。因而,还需要对连接件进行相应的修改为 Aspectual Connector,它的实例不仅可以用来连接不同的组件,也可以用来连接组件与 Aspect。下面将给出 AO-ADL 的语法和语义。

(1) Aspect 的表示。一个横切关注点是通过 Aspect 所提供的服务来表示的,它能够影响数个与之有横切关系的组件。笔者通过引入一个体系结构元素 Aspect 来对软件系统中的横切关注点模块化,Aspect 包括切点(Pointcut)和通知(Advice)两个部分。其中,切点用于捕捉和识别连接点,决定 Aspect 将横切入哪些组件中去,通知表示 Aspect 具有什么样的横切功能。

切点是由一系列连接点通过“与”、“或”、“非”(“&&”、“‖”、“!”)组成的,分别表示为“∧”、“$\vee$”、“~”,一个连接点(Joinpoint)就是组件应用程序执行过程中一个定义明确的位置,Aspect 将在此处横切入组件。切点的语法为

% POINTCUT PointcutName =

Joinpoint Declaration ;[∧ ; $\vee$;~]

通知是一个完整的横切行为,是对软件体系结构中存在的一些横切功能的具体描述。这部分采用 XYZ/E 描述为

% ADVICE AdviceName = [Advice Specification]

由于新增的 Aspect 与一般组件相比具有不同的含义和结构, Aspect 和组件的复合方式与组件间的复合方式也不相同,而且 Aspect 的影响范围也更为广泛,涉及到多个组件或者其他体系结构元素。对于 Advice 的横切行为何时横切到 Aspectual Connector 所连接的组件中去,主要有连接点之前(before)、连接点附近(around)、连接点之后(after)。Aspect 及其内部各部分之间的关系的语义解释如下。

定义 1　一个 Aspect 的 AO-ADL 描述分为切点

和通知两部分。切点包含一组连接点,通知是 Aspect 横切功能的完整行为描述。Aspect 的规范是一个 XYZ/E 单元:

$$(\text{POINTCUT}_1 \parallel \cdots \parallel \text{POINTCUT}_n) \parallel (\text{ADVICE}_1 \parallel \cdots \parallel \text{ADVICE}_n)$$

其中 POINTCUT 的集合为:$\{\text{JointPoint}_1, \cdots, \text{JointPoint}_n\}$,Aspect 横切入组件的时间顺序可以由时序算子表示。

(2) Aspectual Connector。通过引入 Aspect 实现了对横切功能的模块化,但是 Aspect 需要横切到相关组件中去。笔者对原有的连接件作相应修改,使其不仅能够连接满足要求的不同组件,也能够连接 Aspect 以及与之有横切关系的组件。连接件是由接口和交互协议两部分组成。接口由一组角色定义,可以将角色分为两类,一类是 BaseRole,用于连接组件的端口;另一类是 AspectRole,用于连接 Aspect 的端口。相应地,交互协议也要做相应的调整。glue 子句一方面表示组件间的交互协议,另一方面可以用来表示 Aspect 与组件之间交互协议。其语法如下。

% ASPECTUALCONNECTOR
AspectualConnectorName = [
% BASEROLE BaseRoleName =
Data Type Declaration;[BaseRoleBehavior]
% ASPECTROLE AspectRoleName =
Data Type Declaration;[AspectRoleBehavior]
% GLUE = = [Interact Protocol]
]

Aspectual Connector 规范语义解释如下。

定义2 Aspectual Connector AC 由一组 BaseRole $BR_1 \cdots BR_n$ 和方面角色 $AR_1 \cdots AR_m$ 以及交互协议 GLUE 定义,其中每个基本角色 BR_i 的行为描述为 $\text{BRBehavior}_i (i = 1, 2, \cdots, n)$,AspectRole AR_i 的行为描述为 $\text{ARBehavior}_i (i = 0, \cdots, m)$,那么 AC 的规范是一个 XYZ/E 单元:

$$\text{GLUE} \parallel (\text{BRBehavior}_1 \parallel \cdots \parallel \text{BRBehavior}_n) \parallel (\text{ARBehavior}_0 \parallel \cdots \parallel \text{ARBehavior}_m)$$

其中,在 GLUE 中出现的通道集合为 $\{R_1, \cdots, R_n\}$。

(3) 组件。有些组件可能与 Aspect 有横切关系,为了表示出其中 Aspect 横切的位置,需要定义一种 Joinpoint 的特殊端口。组件有需求组件和提供组件之分。对于这些与 Aspect 有横切关系的组件来说,表现为对 Aspect 的功能的需求,因而可以将这类组件认为是一种需求组件。对于组件中的连接点,可以用 XYZ/E 一组通道表示,其语法为

% JOINPOINT JoinpointName = =
FunctionName Declaration;

组件规范的语义解释如下。

定义3 一个组件 DC 分为接口描述和计算规范两个部分。接口描述包含一组端口、一个性质描述以及一组 Joinpoint。Joinpoint 显示 Aspect 横切入组件的位置,其个数可以为 0,表示组件不与任何 Aspect 有横切关系。计算规范是组件的完整抽象行为描述。若 DC 有 k 个端口,它们的行为描述为 $\text{PBehavior}_1, \cdots, \text{PBehavior}_k$ 功能规范为 f,计算规范为 ComSpec,则它们之间有如下关系:$\text{Pbehavior}_i (i = 1, \cdots, k)$ 是 ComSpec 关于各个端口上出现的不包含控制变量的变量集的求精,并且 ComSpec $\Rightarrow f$,即 ComSpec 逻辑蕴涵 f。

(4) Aspect 与组件之间复合。XYZ/ADL 中的复合组件是由一些组件根据一定的条件连接而成的。同样地,Aspect 与组件之间也是通过 Aspectual Connector 进行复合的。与组件之间的复合不同的是:Aspect 与组件通过复合,使得 Aspect 的功能扩散到与之复合的组件中去。复合后的组件对外表现的功能不是其自身功能和 Aspect 横切功能的简单叠加,而是通过 Aspect 的横切机制实现了在功能上的融合,展现出的是一个完整的功能,其端口也没有发生任何变化,但 Joinpoint 的个数减 1。Aspect 与一个简单组件复合后的组件仍然可以视为一个新的简单组件,还可以与其他组件进行复合。而对于一个 Aspect 而言,其横切功能本身没有发生任何变化,仍然需要在与之有横切关系的其他组件中的连接点处执行横切行为。复合语法为

% COMPOSITION = = [
ComponentInstanceName:ComponentName;
…
AspectInstanceName:AspectName;
…
AspectualConnectorInstanceName:
AspectualConnectorName;
…]

Aspect 与组件的连接定义表明它的内部体系结构配置,语法为

% ATTACHMENTS = = [
ComponentInstanceName. JoinpointName#

AspectualConnectorInstanceName. BaseRoleName；
AspectInstanceName. Pointcut#
AspectualConnectorInstanceName. AspectRoleName；
…]

其中,“#”定义了连接操作,表示组件端口与 AspectualConnector 的基本角色相连接,Aspect 的 Pointcut 与 AspectualConnector 的 AspectRole 相连接。这里的组件对外所表现出的始终是对一种横切功能的需求,而 Aspect 表现出的始终是对外提供一种横切功能。此外,Aspect 与组件之间的复合所影响的是组件中的连接点,并不会对组件中的端口产生任何影响。因而,不需要做绑定(bind)工作。对于一个复合组件,如果是由不同的组件连接而成的,即使其内部的子组件中存在连接点,仍然可以采用 XYZ/ADL 中的组件复合方式,对其内部体系结构配置表示方式不作任何改变。

5 面向方面的体系结构核心模型

在前面,笔者通过在 XYZ/ADL 中引入新的体系结构元素 Aspect,并对连接件作了相关修改,给出组件与 Aspect 之间的复合机制,从而实现了对软件体系结构中的横切功能模块化。最终的目的就是要表示“什么”、“何时”以及“怎样”,Aspect 的行为发生在那些与之有横切关系的组件的连接点处。经过这些扩展工作后,可以归纳得到一种面向方面的软件体系结构核心模型,如图 1 所示。

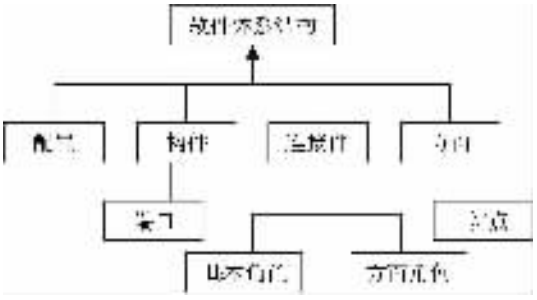

图 1　面向方面的软件体系结构核心模型

6 小结

本文分析了当前面向方面建模的两种研究趋势,指出了编码阶段所存在的散射和缠结现象同样存在于软件体系结构中。因而,笔者提出了将编码阶段的 AOP 相关概念引入到软件体系结构中。通过在 XYZ/ADL 中引入 Aspect,在连接件中增加一个方面角色来处理方面与组件之间的交互,给出了 Aspect 与组件之间的复合规则,从而得到了一种面向方面的体系结构描述语言即 AO-ADL。使得在体系结构建模层次上就将系统的横切关注点提取出来,实现了从需求到编码的平滑过渡,消除了软件体系结构中的散射和缠结现象,提高了软件系统的模块化程度。笔者下一步的工作就是在此基础上,对一个实例系统进行描述,并进行求精与验证工作。

参考文献:

[1] KICZALES G, LAMPING J, MENDHEKAR A, et al. Aspect-Oriented Programming [C]. ECOOP'97, LNCS 1241, Berlin Heidelberg: Springer-Verlag, 1997, 220-242.

[2] COTTENIER T, BERG A V D, ELRAD T. Modeling Aspect-Oriented Compositions [J]. MoDELS 2005 Workshops, LNCS3844, Berlin Heidelberg: Springer-Verlag, 2006, 100-109.

[3] HAN Y, KNIESEL G, CREMERS A B. Towards Visual Aspect J by a Meta Model and Modeling Notation [C]. In: Proc of the 6th International Workshop on AOM at the 4th International Conference on AOSD, 2004.

[4] 郭东亮,张立臣. 基于扩展 UML 的面向方面的建模[J]. 计算机工程,2006,32(19):100-102.

[5] 张广泉,刘剑云,黄正宝. 基于 XYZ/E 的 CA 系统体系结构描述[J]. 重庆师范大学学报(自然科学版),2005,22(3):33-36.

[6] 张广泉,张玲红. UML 与 ADL 在软件体系结构建模中的应用研究[J]. 重庆师范大学学报(自然科学版),2004,21(4):1-6.

[7] 张广泉,刘艳. 基于 UML 的图书管理系统体系结构模型及实现[J]. 重庆师范大学学报(自然科学版),2005,22(2):1-5.

[8] 张广泉,朱雪阳,郑建丹. 基于时态逻辑语言 XYZ/E 的软件体系结构(Ⅱ)——常见体系结构风格的描述[J]. 重庆师范学院学报(自然科学版),2002,19(1):1-3.

[9] CUSTA C E, DEL PILAR ROMAY M, DE LA FUENTE P, et al. Architectural Aspects of Architectural Aspects [J]. EWSA 2005, LNCS 3527, Berlin Heidelberg: Springer-Verlag, 2005. 247-262.

[10] 朱雪阳,唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL[J]. 软件学报,2003,14(4):713-720.

[11] GARCIA A, CHAVEZ C, BATISTA T, et al. On the Modular Representation of Architectural Aspects [J]. EWSA 2006, LNCS 4344, Berlin Heidelberg: Springer-Verlag, 2006. 82-97.

[12] PEREZ J, Nour ALI N, CARSL J A, et al. Designing Software Architectures with an Aspect-Oriented Architecture Description Language [J]. CBSE 2006, LNCS 4063, Berlin Heidelberg: Springer-Verlag, 2006. 123-138.

[13] NAVASA A,ANGEL PEREZ M,MANUEL MURILLO J. Aspect Modeling at Architecture Design[J]. EWSA 2005, LNCS 3527,Berlin Heidelberg:Springer-Verlag,2005. 41-58.

[14] MEDEIROS ELER M,CESAR MASIERO P. Aspect as Components[J]. ICSR 2006, LNCS 4039,Berlin Heidelberg:Springer-Verlag, 2006. 411-414.

Research into Aspect-Oriented Modeling Based on Software Architecture

ZHANG Guang-quan,YANG Jing-zhong

(College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047, China)

Abstract:Aspect-Oriented Programming is a kind of new technology based on concern separation, which succeeds in solving the code tangling and scattering deficiency with traditional Object Oriented Programming. However, the Aspect-Oriented concepts have not been well supported yet at the early phase of software design. By analyzing the key theory and concepts of AOP, it indicates that the disadvantages of code tangling and scattering also exist at the software architecture level. In order to find the architectural aspects, which crosscut components, and to separate them form components, it is necessary to introduce a new element that is distinctive from the concepts of component and connector. By adding the new first-class entity that is used to represent the architectural aspect, modification of connector to connect aspect and component, and composition mechanism to the software architecture description language XYZ/ADL, a kind of AO-ADL (Aspect-Oriented Architecture Description Language) is proposed to capture and modularize the crosscutting concerns as aspects at the software architecture level.

Key words:Software Architecture;Aspect-Oriented Modeling;XYZ/ADL;Aspect-Oriented Software Architecture Description Language

(责任编辑 游中胜)