一种面向方面软件体系结构模型 *

朱春国, 曾国荪
(同济大学 计算机科学及技术系, 嵌入式系统与服务计算教育部重点实验室,上海 201804)

摘 要: 为了分离软件系统中的核心关注点和横切关注点,通过引入面向方面软件开发的思想设计了一种面向方面软件体系结构模型,并详细分析了该模型的三个基本构成单元,即构件、连接件和方面构件。最后通过一个网上支付实例验证了该模型具有一定的理论意义和实用价值。

关键词: 面向方面软件体系结构; 横切关注点; 构件; 连接件; 方面构件
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2010)09-3387-03
doi:10.3969/j.issn.1001-3695.2010.09.050

Aspect oriented software architecture model

ZHU Chun-guo, ZENG Guo-sun
(*Embedded System & Service Computing Key Laboratory of Ministry of Education, Dept. of Computer Science & Technology, Tongji University, Shanghai* 201804, *China*)

Abstract: For making good separation of the corn concerns and crosscutting concerns of software systems, this paper introduced an aspect oriented software architecture (AOSA) model based on the thought of aspect oriented software development (AOSD), and designed its three basic units with details: component, connector and aspect component. Finally, this model shows its theoretical and practical value through a case of online payment.

Key words: aspect oriented software architecture; crosscutting concern; component; connector; aspect component

20 世纪 60 年代的软件危机使得人们开始重视软件工程的研究。起初,人们把软件设计的重点放在数据结构和算法的选择上,然而随着软件系统规模越来越大,对总体的系统结构设计和规格说明变得异常重要。随着软件危机程度的加剧,软件体系结构(software architecture)这一概念应运而生。软件体系结构着眼于软件系统的全局组织形式,在较高层次上把握系统各部分之间的内在联系,将软件开发的焦点从成百上千的代码上转移到粒度较大的体系结构元素及其交互的设计上。与传统软件技术相比,软件体系结构理论的提出不仅有利于解决软件系统日益增加的规模和复杂度的问题,有利于构件的重用,也有利于软件生产率的提高。面向方面软件开发(AOSD)认为系统是由核心关注点(corn concern)和横切关注点(crosscutting concern)有机地交织在一起而形成的。核心关注点是软件要实现的主要功能和目标,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,如系统日志、事务处理和权限验证等。AOSD 通过分离系统的横切关注点和核心关注点,使得系统的设计和维护变得容易很多。

Extremadura 大学的 Navasa 等人[1]在 2002 年提出了将面向方面软件开发技术引入到软件体系结构的设计中,称之为面向方面软件体系结构(aspect oriented software architecture, AOSA),这样能够结合两者的优点,但是并没有给出构建面向方面软件体系结构的详细方法。

尽管目前对于面向方面软件体系结构这个概念尚未形成统一的认识,但是一般认为面向方面软件体系结构在传统软件体系结构基础上增加了方面构件(aspect component)这一新的构成单元,通过方面构件来封装系统的横切关注点。目前国内外对于面向方面软件体系模型的研究还相对较少,对它的构成单元模型的研究更少,通常只关注方面构件这一构成单元。方面构件最早是由 Lieberherr 等人[2]提出的,它是在自适应可插拔构件(adaptive plug and play component, APPC)基础之上通过引入面向方面编程(AOP)思想扩展一个可更改的接口而形成的,但它关于请求接口和服务接口的定义很模糊,未能给出一个清晰的方面构件模型。Pawlak 等人[3]提出了一个面向方面的框架,该框架主要包含了一个方面构件模型——Java 方面构件(Java aspect component, JAC),但该方面构件模型仅包含了切点(pointcut),并把 AOP 中装备(advice)集成到了切点的表达式中,它主要从实现的角度进行了阐述,并没有给出详细的方面构件模型。本文没有只关注面向方面软件体系结构中方面构件这一构成单元模型,还详细分析了它的另外两个构成单元,即构件和连接件,因为面向方面软件体系结构各部分之间是相互关联的。

1 面向方面软件体系结构相关概念

面向方面软件体系结构涉及诸多概念,以下将分别介绍。

收稿日期: 2010-03-22; **修回日期**: 2010-04-26 **基金项目**: 国家"863"计划资助项目(2007AA01Z425, 2009AA012201);国家"973"计划课题(2007CB316502);国家自然科学基金资助项目(90718015);NSFC—微软亚洲研究院联合资助项目(60970155);国家教育部博士点基金资助项目(20090072110035);上海市优秀学科带头人计划资助项目(10XD1404400);高效能服务器和存储技术国家重点实验室开放基金资助项目(2009HSSA06)

作者简介:朱春国(1985-),男,硕士研究生,主要研究方向为软件构件技术和软件体系结构(nestar2008@gmail.com);曾国荪(1964-),男,教授,博导,博士,主要研究方向为网格计算、信息安全.

软件体系结构在软件工程领域有着广泛的影响,但当前仍未形成一个统一的、标准的定义。目前国内外普遍认可的看法是软件体系结构包含构件、连接件和约束[4]。其中约束描述了体系结构配置和拓扑的要求,确定了体系结构的构件与连接件的连接关系。这样就可以把软件体系结构写成

软件体系结构(software architecture) = 构件(components) + 连接件(connectors) + 约束(constraints)

构件是软件体系结构的基本元素之一。一般认为,构件是指具有一定功能、可明确辨识的软件单位,并且具备语义完整、语法正确、有可重用价值的特点,然而目前对于构件的具体结构及构成并没有一个统一的标准[5],而且一些主要的构件技术也没有使用相同的构件类型。另外,当前被广泛接受的构件定义并不包含具体的软件构件模型(software component model)。例如,Szyperski 等人[6]给出了软件构件一个很有名的定义:软件构件是一个仅带特定契约接口和显式语境依赖的结构单位,它可以独立部署,易于第三方整合。但是关于软件构件模型有一个被普遍接受的观点是:软件构件是一个具有服务提供和服务请求功能的软件单元[7]。

连接件是软件体系结构另一个基本的构成元素,是用来建立构件间交互以及支配这些交互规则的构造模块。连接件最先是由 Shaw[8]提出来的,她建议把连接件作为软件体系结构中第一类实体,用来表示普通构件之间的交互关系。目前对于连接件尚未形成统一的认识,尽管在软件体系结构中强调了连接件存在的必要性,但是关于连接件模型的研究还很少,连接件的实际应用还不成熟。

面向方面软件体系结构在传统软件体系结构的基础上增加了方面构件单元。通常认为,方面构件是封装了系统横切关注点的一类特殊的构件。目前关于方面构件模型的研究还处于起步阶段。

2　面向方面软件体系结构模型

由于传统软件体系结构模型包含构件、连接件和约束,而面向方面软件体系结构是在传统软件体系结构的基础之上扩展了方面构件,所以面向方面软件体系模型结构包含构件、连接件、方面构件和约束。其中约束描述了面向方面体系结构配置和拓扑的要求,确定了体系结构的构件、连接件和方面构件之间的连接关系,而构件、连接件、方面构件是它的三个基本的构成单元。以下对这三个构成单元的模型进行详细的设计。

2.1　构件模型

构件模型由以下几个要素构成(图 1):

a)端口。构件的服务请求和服务提供功能是通过端口来实现的。端口是构件与外部环境进行交互的惟一通道。一般的构件模型通常采用两种端口,即双向端口和单向端口。在使用双向端口的构件模型中,服务请求和服务提供功能可以在同一个端口中实现。本文中的构件模型使用单向端口,此种端口分为请求端口和服务端口两种类型。

(a)服务端口。构件通过服务端口向其他构件提供服务。构件通过服务端口向其他构件的请求消息进行应答,返回响应消息。每个服务端口对应一个接口。

(b)请求端口。构件通过请求端口向其他构件请求服务。构件为了实现自己的业务功能,需要通过请求端口向其他构件发送请求消息。每个服务端口也对应一个接口。

b)接口。它定义了一个到多个业务功能。这些业务功能由服务端口进行提供,并由请求端口进行使用。一个接口限定了一个特定端口可以进行的交互功能,接口是构件间交互的契约。通常的接口类型有:Java Interface、WSDL 1.1 portTypes 和 WSDL 2.0 Interfaces 等,也可以自定义接口类型。

c)属性。与类或对象相似,构件也具有属性,属性可以在构件使用前进行配置,它能够反映构件在交互过程中状态的变化。

2.2　连接件模型

连接件是用来建立构件间交互以及支配这些交互规则的体系结构构造模块。连接件为构件间信息交互提供传输和路由服务。在最简单的情况下,构件之间可以直接完成交互,这时体系结构中的连接件就退化为直接连接。在更为复杂的情况下,构件间交互的处理和维持都需要连接件来实现。对于构件而言,连接件是构件的粘合剂,是构件交互的实现,也可以看做是一种特殊的构件[8]。与构件相似,连接件也具有端口。连接件的端口可分为两种类型,即源端口(source port)和目标端口(target port)。源端口用于接收构件请求端口中的消息,目标端口用于向构件服务端口中输入消息。连接件通常需要使用一种合适的绑定(binding)机制,构件的请求端口使用这种绑定机制来描述服务请求的方法,构件的服务端口也使用这种机制来描述构件进行请求的方式。常用的绑定机制有:Web Service Binding 和 JMS Binding 等,也可以自定义绑定机制。与构件一样,连接件也具有属性,来表示构件间交互的状态变化,如图 2 所示。

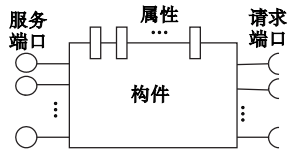

图1　构件模型

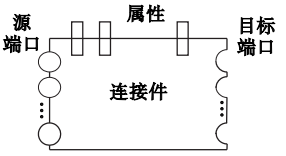

图2　连接件模型

2.3　复合构件模型

构件可分为两种,即原子构件和复合构件。前者是不可再分的构件。后者是可再分构件,它封装了若干个子构件。子构件间通过连接件相互连接,且子构件的端口也可以暴露成为复合构件的端口,子构件也可能是复合构件。如图 3 所示:复合构件 A 包含两个子构件 B 和 D,子构件 B 和 D 通过连接件 C 进行相连,构件 B 的服务端口 E 暴露成为复合构件 A 的服务端口 F,其请求端口 G 暴露成为 A 的请求端口 H。

2.4　方面构件模型

方面构件是面向方面软件体系结构的一个核心的构成单元,它封装了横切关注点,这是与传统软件体系结构最大的不同之处。图 4 给出了方面构件模型,与普通构件一样,方面构件也有服务端口和请求端口以及属性,但是它还有普通构件所没有的方面端口。当一个构件具有一个方面端口时,即可认为此构件就是方面构件。一个方面端口中包含若干个方面,这与一般面向方面编程(AOP)技术中方面概念有所不同。面向方面编程具有以下四个基本概念:方面(aspect)、连接点(join point)、通知(advice)和切点(pointcut)。连接点是应用程序执行过程一个定义明确的位置,如方法调用是一种典型的连接点。切点是一系列连接点的集合,是方面的作用点。通知表述了在切点所选定的连接点处要执行的动作,常见通知类型有 before、around 和 after 等,分表代表在连接点之前、连接点附近和连接点之后执行相应的通知代码。方面是用来描述和实现

横切关注点的基本单位,由切点和通知构成。方面端口中的方面横切关注的是构件,这与一般 AOP(如 AspectJ)横切关注的对象(object)不同,由于构件能够表达对象所不能表达的请求服务的能力[9],这使得方面端口中方面所采用的连接点模型和切点语言具有很大的不同。

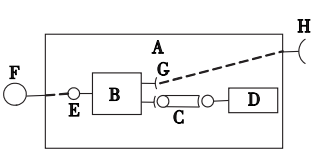

图3　复合构件模型

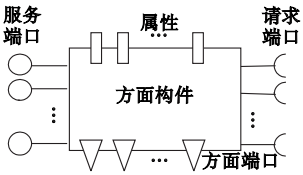

图4　方面构件模型

2.4.1　连接点模型

该连接点模型包含两种不同类型的连接点,即构件服务端口中的服务提供操作和请求端口的服务请求操作。由于构件的内部结构通常被视为黑盒,因此连接点模型应该仅考虑构件的外部可见元素,如构件请求端口和服务端口中的服务操作。如果连接点模型包含构件的属性,那么它将会破坏构件的分装性。

2.4.2　切点语言

用来选用连接点的切点语言基于切点表达式,表 1 给出了切点的五个组成部分,即 component、jp_type、port、interface 和 service,然后分别对其进行了说明。其中,jp_type 代表选用的连接点类型,可以是请求端口中的服务、服务端口中的服务或所有端口中的服务,详细如表 1。表 2 给出了切点语言的一些例子,其中正则表达式基于 java.util.regexp 包。

表 1　切点语言语法

域	说明
pcd	component; jp_type; port; interface; service
component	构件名的正则表达式
jp_type	provided|required|both
port	端口名的正则表达式
interface	接口名的正则表达式
service	服务名的正则表达式

表 2　切点语言实例

切点表达式	捕获的元素
* ; * ; * ;account * ;void	任意构件的使用任意接口的任意端口中以 account 开头且返回值为空的服务
A;required; Java interface; * account	构件 A 的请求端口且使用 Java 接口中的以 account 结尾的服务
B;provided; * ; *	构件 B 的使用任意接口的服务端口中的任意服务

2.5　面向方面软件体系结构模型

面向方面软件体系结构由构件、连接件、方面构件组成,详细请参见图 6。

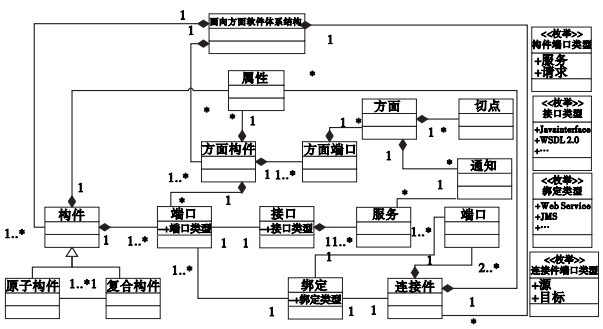

图6　面向方面软件体系结构模型

3　基于面向方面软件体系结构模型的网上支付实例

近年来,网上购物发展迅速,网上支付是消费者主要的支付手段之一,图 7 给出了基于面向方面软件体系结构的网上支付模型,它由四个原子构件,即一个复合构件、两个方面构件和三个连接件组成。其中 WebClientComponent 代表客户端构件,它可以向网上银行构件 WebBankComponent 请求 AccountService( )服务,该服务有三个参数,即 username、password、cost,分别对应于用户的网上银行账户名、密码及购买商品的消费金额。

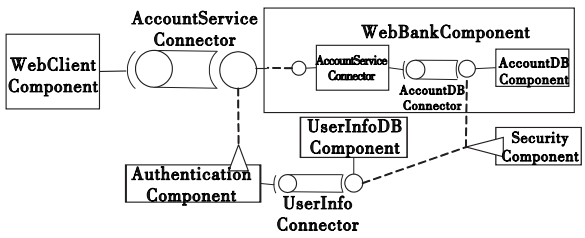

图7　网上支付模型

```
〈component name = "WebClientComponent"〉
  〈required. port name = "WebClientRequest"〉
    〈java. interface interface = "AccountServiceInterface"〉
      〈service name = "AccountService( )"〉
        〈param name = "username" type = "string" /〉
        〈param name = "password" type = "string" /〉
        〈param name = "cost" type = "float" /〉
      〈/service〉
    〈/java. interface〉
  〈/required. port〉
〈/component〉
```

连接件 AccountServiceConnector 用于连接客户端构件和网上银行构件,它采用 WebServiceBinding 绑定机制。

```
〈connector name = "AccountServiceConnector" binding = "WebServiceBinding" /〉
    〈source name = "S" /〉
    〈target name = "T"〉
〈/connector〉
〈connect. source from = "WebClientComponent. WebClientRequest" to = "S" /〉
〈connect. target from = "T" to = "WebBankComponent. Bank-Response" /〉
```

网上银行构件是一个复合构件,由账户服务构件 AccountServiceComponent、账户数据库连接件 AccountDBConnector 和账户数据库构件 AccountDBComponent 组装而成。其中该复合构件的服务端口也使用接口 AccountServiceInterface,这是为了兼容客户端构件请求端口使用的接口。

身份验证构件 AuthenticationComponent 用于验证用户的身份信息,它通过 UserInfoConnector 连接件访问用户信息数据库构件 UserInfoDBComponent。pointcut = " WebBankComponent; BankResponse; AccountServiceInterface; AccountService( )" 是该方面构件的方面端口中使用切点的表达式。

为了保证数据库构件 UserInfoDBComponent 和 AccountDBComponent 的安全性,方面构件 SecurityComponent 使用方面端口 Security 监视这两个构件的服务端口,使得在这两个构件服务调用之前增加日志和事务功能,而日志和事务功能在系统中通常表现为横切关注点,面向方面软件体系结构能够对它进行很好的封装,便于设计和维护。

(下转第 3394 页)

索在 Web 为中心,通过对用户提供资源分配服务的网络计算条件下,如何进一步提高地理空间数据可用性的系统架构和数据传播方式。为了解决复杂时空数据对变长和嵌套的存储元组问题,本文提出了一个基于分布式文档数据库的地理空间数据多版本并发控制和 B + 树索引的管理方法,与现有数据库扩展方法相比,该方法具有无限扩展部署、协议平台透明等优点。在数据检索分析方面,本文提出的全栈式定向聚合方法通过异域对接代码元编程的方式生成映射规约高阶索引函数并发处理和容错监控,提高了地理空间数据的共享和传播效率。通过对原型系统的模拟压力测试进一步验证了本文模型在实际开放网络条件下并发地理时空数据传播的有效性和鲁棒性。本文的进一步工作包括:扩展支持文档数据库的文件系统,优化动态语言运行效率和分布系统稳定性和容错率。

参考文献:

[1] 刘仁义. 面向网络的海量空间数据与时态空间数据模型及其应用研究[D]. 杭州:浙江大学,2004.

[2] 徐志红,边馥苓,陈江平. 基于事件语义的时态 GIS 模型[J]. 武汉大学学报:信息科学版,2002,27(3):311-315.

[3] HEINE F, HOVESTADT M, KAO O. Towards ontology-driven P2P grid resource discovery[C]//Proc of the 5th IEEE/ACM International Workshop on Grid Computing. Washington DC: IEEE Computer Society, 2004.

[4] FOSTER I, KESSELMAN C, NICK J M, *et al*. Grid services for distributed system integration[J]. Computer, 2002, 35(6): 37-46.

[5] ALPDEMIR M N, MUKHERJEE A, PATON N W, *et al*. Service-based distributed querying on the grid[C]//Proc of the 1st International Conference on Service Oriented Computing. 2003.

[6] RDF Working Group. Resource description framework (RDF)[M]. [S. l.]: World Wide Web Consortium, 2004.

[7] DEAN J, GHEMAWAT S. MapReduce: Simplified data processing on large clusters[C]//Proc of the 6th conference on Symposium on Opearting Systems Design & Implementation. 2004.

[8] CHANG F, DEAN J, GHEMAWAT S, *et al*. Bigtable: distributed storage system for structured data[C]//Proc of Seattle. 2006.

[9] GHEMAWAT S, GOBIOFF H, LEUNG S T. The Google file system [J]. ACM SIGOPS Operating Systems Review, 2003, 37(5): 29-43.

[10] 陈述彭,鲁学军,周成虎. 地理信息系统导论[M]. 北京:科学出版社,2000.

[11] TUDRUJ M. Scalable computing practice and experience[M]. ISPDC, 2008.

[12] 于宇,唐晓嘉. 蒙太格 PTQ 系统的内涵逻辑[J]. 西南大学学报:社会科学版,2009,35(1):81-86.

[13] ALEXANDER C. The timeless way of building[M]. Oxford: Oxford University Press, 1979.

[14] OGC Reference model[M]. 2nd ed. Inc, Open Geospatial Consortium, 2008.

[15] CZAJKOWSKI K, FITZGERALD S, FOSTER I, *et al*. Grid information services for distributed resource sharing[C]//Proc of the 10th IEEE Int'l Symp on High Performance Distributed Computing. Los Alamitos: IEEE Computer Society, 2001.

[16] 徐贵红,张健. 语义网的一阶逻辑推理技术支持[J]. 软件学报,2008,19(12):3092-3099.

[17] FIELDING R T. Architectural styles and the design of network-based software architectures[D]. Irvine: University of California, 2000.

(上接第 3389 页)

```
〈aspect. component name = "SecurityComponent"〉
〈aspect. port name = "Security"〉
〈aspect〉
〈pointcut = "UserInfoDBComponent; UserInfoResponse; *; * | AccountDBComponent; AccountDBResponse; *; *" /〉
〈advice. role = "before" action = "Log()" /〉
〈advice. role = "before" action = "Transaction()" /〉
〈/aspect〉
〈/aspect. port〉
〈required. port name = "UserInfoRequest" /〉
〈/aspect. component〉
```

4 结束语

本文给出了一种面向方面软件体系结构模型,详细设计了它的三个基本构成单元模型,即构件、连接件和方面构件;最后通过一个网上支付实例验证了该模型有效性和实用性,为面向方面软件体系结构的实际应用奠定了一定的基础。笔者将继续完善该模型的相关理论,研究面向方面软件体系结构的工程化应用方法。

参考文献:

[1] FABRESSE L, DONY C, HUCHARD M. Foundations of a simple and unified component-oriented language[J]. Journal of Computer Languages, Systems & Structures, 2008, 34(2-3): 130-149.

[2] LIEBERHERR K, LORENZ D, MEZINI M. Programming with aspectual components, T R NU-CSS-99-01[R]. [S. l.]: Nouheastam University, 1999.

[3] PAWLAK R, SERNTURIER L, DUCHIEN L D, *et al*. JAC: an aspect-based distributed dynamic framework[J]. Software Practice and Experiences, 2004, 34(12): 1119-1148.

[4] 李千目. 软件体系结构设计[M]. 北京:清华大学出版社,2008.

[5] 马亮,孙春艳. 软件构件概念的变迁[J]. 计算机科学,2002,29(4):28-30.

[6] SZYPERSKI C, GRUNTZ D, MURER S. Component software: beyond object-oriented programming[M]. 2nd ed. [S. l.]: Addison-Wesley, 2002.

[7] LAU K K, WANG Z. Software component models[J]. IEEE Trans Soft Eng, 2007, 33(10): 709-724.

[8] SHAW M. Procedure calls are the assembly language of software interconnection: connectors deserve first-class status[C]//Proc of In ICSE Workshop on Studies of Software Design. 1993: 17-32.

[9] NAVASA A, PÉREZ M A, MURILLO J M, *et al*. Aspect oriented software architecture: a structural perspective[C]//Proc of Workshop on Early Aspects. 2002.