面向方面的中间件

刘敬勇[1,2]，张立臣[1]，钟 勇[2]
（1.广东工业大学 计算机学院，广东 广州 510006；
2.中国科学院 成都计算机应用研究所，四川 成都 610041）

摘 要：中间件应用领域的不断拓展，给中间件体系结构的设计带来了困难，中间件的体系结构必须在通用性与专用性之间寻找平衡。传统的中间件体系结构产生这些问题的一个基本原因是：使用垂直分解过程获得的软件分解模型不能同时模块化共存的正交设计需求。论述了针对专门领域的面向方面中间件开发的三种工具：AspectIX 和 QuO 处理 QoS 关注点，DIL 从功能代码中分离协议实现。在建造一个中间件系统时，使用这些工具可以获得较好的模块性、可配置性和代码的演化性。

关键词：面向方面；中间件；分布式系统；关注点分离
中图分类号：TP311.52 **文献标识码**：A **文章编号**：1673-629X(2008)08-0068-04

Aspect-Oriented Middleware

LIU Jing-yong[1,2], ZHANG Li-chen[1], ZHONG Yong[2]
(1. College of Computer, Guangdong University of Technology, Guangzhou 510006, China;
2. Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041, China)

Abstract: The widening of the application spectrum has brought some difficulties to design the architecture of middleware systems. The most prominent problem with middleware systems is that the architecture of middleware constantly struggles between generality and specialization. One of the fundamental problems in middleware architecture is that software decomposition models obtained using vertical decomposition procedures are incapable of simultaneously modularizing coexisting orthogonal design requirements. Survey three aspect oriented approaches to developing middleware. AspectIX and QuO address the QoS concern, and DIL separates the protocol implementation from the functional code. The use of these tools in building a system will likely result in greater modularity, configurability and evolvability of the code.

Key words: aspect-oriented; middleware; distributed systems; concern separation

0 引 言

中间件是介于操作系统和各种分布式应用程序之间的一个软件层，它建立分布式软件模块之间互操作的机制，屏蔽底层分布式环境的复杂性和异构性，为分布式应用程序的开发提供支持。近几年来，随着中间件技术的不断进步，像 Web Services，CORBA，J2EE 和 .NET不仅被用于传统的企业计算平台，而且对于一些在运行时有苛刻的资源限制并要求实时性、高性能、高可靠性的领域有着广泛的应用。例如中间件系统被应用在 Cisco ONS 15454 光传输平台上，用来处理硬件配置和管理软件与硬件驱动程序之间的通信。中间件系统也被美国海军作为连接潜艇作战控制系统的各个子单元的软件总线[1]。应用领域的扩大使得中间件系统的能力得到不断延伸。

1 传统中间件体系结构面临的问题

由于中间件应用领域的不断拓展，这给中间件体系结构的设计带来了困难，即中间件的体系结构必须在通用性与专用性之间寻找平衡。因为，一方面中间件供应商希望自己的中间件产品尽可能支持较多的应用领域，所以这些中间件产品往往提供了一个相对完整的特性集合，然而这导致用这些中间件开发的系统要求庞大的内存空间和足够的计算资源。另一方面，中间件设计师希望优化中间件体系结构，使它们在支

收稿日期：2007-11-09
基金项目：国家自然科学基金资助项目(60474072,60174050)；广东省自然科学基金资助项目(04009465,010059)；广东省高校自然科学研究项目(Z03024)
作者简介：刘敬勇(1975-)，男，博士研究生，讲师，研究方向为实时系统与软件工程；张立臣，博士，教授，研究方向为并行处理、分布式处理、实时系统；钟 勇，博士，研究员，研究方向为软件过程与技术方法。

持具有特定运行时要求,比如实时性、高可用性和高可靠性的应用领域时性能最佳。这样传统中间件体系结构的限制显现出来[2]:

(1) 通用性过强:很多中间件实现为了支持多种不同的目标平台,包含了大量的特性集合。然而,在实际应用中,对于特定领域的具体实例,很多特性并不是一直需要,有些特性甚至在应用程序的整个生命周期都毫无必要。当前的中间件体系结构缺乏在程序部署或运行时将自己裁剪至满足特定需要的有效方法。

(2)变化过于迅速:虽然网络通信模型是相对稳定的,中间件系统中的模型和抽象却很少保持不变,因为中间件体系结构需要不断地适应新的特殊领域的计算特性。比如,对象适配器是 CORBA 平台的一个负责管理服务和分派请求的构件。即使对象适配器扮演的角色自 CORBA 诞生以来就没有显著的变化,但它的体系结构却经过了一系列的发展:从基本的对象适配器到支持可移植性、可拦截性以及多线程。中间件抽象模型的发展是必要的也是不可避免的,但是,它同时也必须保持对建立在旧的中间件模型之上的应用程序的向后兼容性。

(3)同种技术,不同规范:为了适应来自不同应用领域的需求,中间件往往需要增生。比如,对象管理组织除了定义 CORBA 标准外,还定义了实时 CORBA 规范以处理执行的可预测性,容错 CORBA 规范以处理高可用性,高性能 CORBA 规范以处理数据和高密度计算的应用,以及用于嵌入式平台的最小 CORBA 规范。Java 平台也出现了在不同平台上使用不同的 Java 虚拟机而产生的相同问题,微软的.net 平台也有着相似的趋势。这些不同的规范使中间件供应商面临挑战,他们必须根据特定的规范重新架构系统,而这同时也让用户感到平台很难理解和使用。

传统的中间件体系结构产生这些问题的一个基本原因是:使用垂直分解过程获得的软件分解模型不能同时模块化共存的正交设计需求。这就需要寻找一种新的分解方法来解决这一问题。

2 面向方面技术应用于中间件系统

面向方面编程是一种基于关注分离的新技术,系统不同的关注能够分离并单独设计,可以解决面向对象编程难于解决的复杂问题。例如在以往的结构化程序设计和面向对象编程技术中,某些程序设计代码分散在系统各个模块中,从而导致系统难以开发和维护,面向方面编程技术能很好地解决这个问题,并能提高模块的重用性[3]。面向方面软件设计把系统建模分成两部分:核心组件和方面。面向方面建模技术允许系统开发者在系统设计时,从核心功能性需求中分离出不同的关注,例如实时性、安全性、日志、同步控制、调度、性能优化、通信管理、资源共享、分布式管理、错误和异常处理等[4]。同时通过支持方面的组合和绑定来实现系统的集成。关注分离可帮助改进系统的设计,开发者只需要实现单独的方面而不必考虑其它方面和系统的核心组件。面向方面的开发步骤分为三步[5]:

(1)方面的分解:分解需求提取出横切关注点和一般关注点,即把核心模块级关注点和系统级的横切关注点分离开来。

(2)关注点的实现:即各自独立地实现这些关注点。

(3)方面的织入:方面编织器通过创建一个方面来指定重组的规则。编织过程使用这些信息来构建最终的系统。

面向方面编程允许用不同的维度分解软件系统,软件开发者可以使用垂直分解过程建立中间件的基本分解模型,它最优化地提供透明的网络通信便利。然后,使用面向方面技术在不改变现存结构的情况下,将正交设计需求的实现添加到基本模型上。

3 开发面向方面中间件的几种工具

3.1 AspectIX 体系结构

AspectIX 是一个 CORBA 兼容的中间件体系结构,它通过使用片断对象模型使 QoS 代码从系统的功能代码中分离出来[6]。片断对象模型允许一个分布式对象被分割成若干个片断,片断之间能相互作用。多主机可以把片断看作是本地的,每个客户可以通过一个接口访问一个片断。AspectIX 体系结构的设计有两个重要的目的:通过关注点的分离使 QoS 相关的分布式系统的软件维护变得容易。通过让客户在运行时插入系统的 QoS 而获得更加动态、灵活的计算环境。

一个具有 AspectIX 的系统的典型运行时行为是这样的:当客户绑定到一个分布式对象,它在本地地址空间接受一个对象的片断。分布式对象的片断之间的通信对客户是不可见的,片断作为对整个对象的一个本地访问点,在该点上,客户可以检查对象支持哪一种服务质量配置。这可以通过一连串的请求获得:客户创建一些描述它自己需求的方面配置对象,然后将这种配置发送给对象。如果对象不能完成这些需求便可拒绝接受。在这种情况下,客户必须寻找另外合适的 QoS 配置。如果对象接受 QoS 需求,就必须完成它们。如果在运行期间对象发现自己不能完成需求,它必须通过回调接口和异常通知客户。在这种情况下,客户不仅得到一列失败配置,而且得到一列仍被支持

的可选配置。

能用一种通用的方式在对象的接口上配置与对象交互的非功能方面。例如实时方面，客户对调用对象能给出最大时间。一般而言，一个分布式对象甚至能同时支持几个方面，每个方面有唯一的类型和定义语义。只要AspectIX实现支持同样的方面类型，在AspectIX结构内就能支持可移植的应用。在一个片断的生命周期内方面可以被激活也可以被解除活动。方面的参数也可以被动态改变。改变方面配置可以导致一个片断自身被另一个更适合完成需求的实现取代。

AspectIX采纳了CORBA的IDL以支持多种语言。分布式对象的本地片断提供了一个用CORBA的IDL描述的接口。这个接口可以像在CORBA中一样被扩展和缩小。当一个片断被创建时，ORB在相应的语言映射里创建两个本地对象：一个片断接口和一个片段实现。

片断接口是一个普通的对象，它在开发过程中自动生成，它仅取决于分布式对象接口的IDL描述。片段接口把方法调用委托给片断实现，片段接口仅维持一个片断实现，并能在运行时隐藏片断实现的交换。每个方面被一个本地对象描述，本地对象可以使用由片断接口提供的普通接口配置来重新恢复。一个方面对象能分别被IDL或者PIDL(伪IDL)测定类型。方面对象提供方面明确的接口。方面配置可以被改变、设置和进行有效性检查。

改变方面配置可以导致当前片断实现被另一个更适合的新配置的片断实现替换。因而，分布式对象的程序员能采用对象片断的何种不同实现取决于方面配置。如果一个方面的实现需要另外的中间件服务或机制，ORB实现可以加载专门的模块。因此，AspectIX对方面的新实现开放甚至对预先不知道的新方面开放。

3.2 QuO体系结构

QuO开发工具集给软件设计者提供一种处理中间件的服务质量方面的机制[7]。这种机制将通知指定并连结到描述系统运行时潜在状态的操作区域。开发者不再仅根据中间件、网络或者应用程序思考问题，而且还可以根据它们之间的相互作用思考问题。这种方法鼓励用户从中间件或应用程序的观点描述系统执行时的潜在点。

QuO体系结具有模块化和可重用的特点。在分布式系统内管理QoS既需要应用软件的知识也需要底层物理设施的知识，在这种体系结构中，支持QoS关注点的可重用模块被称为qosket。每个qosket从contracts, sysconds, callbacks和delegates的集合中建造。

软件设计者必须从为qosket创建契约开始，每个契约定义了一套系统条件对象或系统条件，而且可以用回调形式包含对中间件的参考。契约设计者可以根据系统条件定义操作区域。系统条件本质上是原子值。从这些值中形成的断言可以激活区域。通常，应用程序(或代理)检查契约确定当前区域以便据此做出反应。

用于创建它们的代理和方法明显地使这种结构面向方面。QuO提出一个专门领域的语言给用户创建对象包裹或代理。用户可以在行为文件中写一些指令(或通知)然后它可以根据哪种应用程序方法被执行以及底层物理设施的当前操作区域来指定何时执行指令。在编译时，QuO方面编织器获得该文件而且创建被称为代理的对象包裹，对象包裹被自动插入应用程序代码内。代理拦截在行为文件中标识的方法的调用并用另外指定的通知执行这些方法。

3.3 DIL体系结构

创建DIL的目的是让应用代码从分布式系统的分布式代码中彻底分离，全面增加系统的灵活性、可维护性和可移植性[7]。关注点的分离在DIL体系结构内的两个层次中完成。首先，是如何通信与何时通信的分离，根据一个基于事件浸透的角色模型书写协议规范，即：一旦系统中的任何两个组件被指定扮演协议中的一部分角色，它们可以仅仅通过发送事件来通信。其次，满足不同需求的协议被相互分开定义。这样设计方面的好处是处理不同关注点的多种协议可以在应用程序上分层。这同时也能获得很大的可定制性，因为添加或移走一个层不会带来副作用。在这种模型的运作方式背后有两种重要的机制：角色和参与者成分以及协议成分。

在DIL中，协议的定义包括：参数、协议操作和角色定义。参数是一套全局值，它可以被协议中的所有参与者共享。在合成期间，对于将在协议中扮演角色的应用层对象，这是一种给协议代码反馈状态或信息的方式。例如，复制协议可以在协议中使用参数指明复制的数量。协议操作支配角色的安装以及其他对协议本身的全局行为。这些操作在给组件分配角色的协议实例上被调用，并将其有效地约束到应用层上。角色定义是与协议相联系的实际行为代码。为了在应用程序内将这些角色组成对象，使用了反射。反射以元对象的形式为对象提供了清楚的描述。在编译期间，用一组元对象表达的协议取代系统定义的元对象。

协议的成分通过同样的方式获得，对于每一个协议、一旦元对象被创建，元对象栈被用于在应用层对象

之上对协议分层。传给应用对象的消息被这个栈拦截,该消息能指导相应层的协议。

4 结束语

中间件平台提供的特点和服务使分布式应用的开发变得容易,然而,这样的特点常常与模型提供的核心功能横切和缠结[8]。即使某些中间件的服务作为组件提供(例如连通性),这些组件也可能被其他中间件的特点(例如安全、事务)横切。中间件的横切本质使得理解、分析和改变中间件的特点变得困难。文中总结了针对专门领域的面向方面中间件开发的三种工具:AspectIX 和 QuO 处理 QoS 关注点,DIL 从功能代码中分离协议实现。在建造一个中间件系统时,使用这些工具可获得较好的模块性、可配置性和代码演化性。

参考文献:

[1] Zhang C, Jacobsen H - A. Quantifying Aspects in Middleware Platforms[C]// In Proceedings 2nd International Conference on Aspect - Oriented Software Development(AOSD). [s. l.]:ACM Press,2003:130 - 139.

[2] Zhang C, Jacobsen H - A. Re - Factoring Middleware Systems: A Case Study[R]. Canada: University of Tortonto, Computer Systems Research Group, 2003.

[3] Loughran N, Parlavantzas N, Pinto M. Survey of Aspect - Oriented Middleware[C]// AOSD - Europe Project Deliverable. [s. l.]:[s. n.], 2005.

[4] Zakaria A, Hosny H, Zeid A. A UML Extension for Modeling Aspect - Oriented Systems[C]// Proceedings of International workshop on Aspect - oriented modeling with UML. Germany:[s. n.], 2002.

[5] Mezini E M, Alice M. Modularization of Middleware using Aspect - Oriented Programming [C] // 4th International Workshop on Software Engineering and Middleware. [s. l.]: Springer - Verlag,2005:47 - 63.

[6] Hauck F J, Becker U, Geier M. AspectIX: A Middleware for Aspect - Oriented Programming[C]// Workshop on Aspect - Oriented Programming (ECOOP'98). Brussels, Belgium:[s. n.],1998:426 - 427.

[7] McCormick E. A Survey of AOP Architectures for Middleware[EB/OL]. 2003. http://www. cs. ubc. ca/emccormi/projects. html.

[8] Hunleth F, Cytron R, Gill C. Building Customizable Middleware using Aspect Oriented Programming[C]// presented at OOPSLA 2001 Workshop on Advanced Separation of Concerns in Object - Oriented Systems. Tampa, Florida:[s. n.], 2001.

(上接第 67 页)

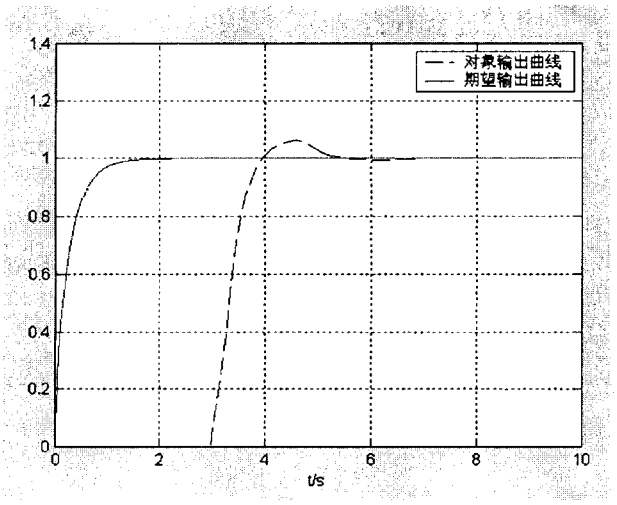

图 5　基于 BP 神经网络预测控制阶跃响应曲线

过神经网络预测控制算法仿真得出的结果。从图 5 中可以看出,对象的输出曲线在 5s 时就能与期望输出曲线吻合而且超调量小,比单纯使用预测算法(见图 3)的效果要好。

4 结束语

通过以上分析,人工神经网络作为一种拟智技术,对预测控制的发展有很好的促进作用。通过该算法在机器人模型上的仿真效果来看,神经网络技术的引入,一方面能加强和丰富预测控制的方法和手段,另一方面能提高预测控制的水平,因此,该方案切实可行。

参考文献:

[1] 余主正,杨马英.基于预测控制算法的网络控制[C]//第五届全球智能控制与自动化大会论文集,第十卷.杭州:浙江大学出版社,2004:15 - 19.

[2] 徐丽娜.神经网络控制系统[M].北京:电子工业出版社,2002:115 - 160.

[3] Farouq. Neural network model - based predictive control of liquid - liquid extraction contactors[J]. Chemical Engineering Science,2006,21(6):659 - 662.

[4] 陈　虹,史旺旺.基于因特网远程控制系统的改进广义预测控制方法[J].电机与控制学报,2005(6):566 - 570.

[5] Aoyama A. A Fuzzy Neural - network Approach for Nonlinear Process[J]. Engng Applic,1995(5):401 - 408.

[6] 杨　平,彭道刚.神经网络预测控制算法及其应用[J].控制工程,2003,9(2):10 - 14.

[7] 刘叔军,盖小华.MATLAB7.0 控制系统应用与实例[M].北京:机械工业出版社,2006.