基于层次消息总线的软件体系结构描述语言

张世琨,王立福,常 欣,杨芙清
(北京大学计算机科学技术系,北京 100871)

摘 要: 以青鸟软件生产线的实践为背景,提出了基于层次消息总线的软件体系结构风格 JB/ HMB,并设计了相应的体系结构描述语言 JB/ SADL,该语言支持从接口、静态结构和动态行为三个方面刻画构件. 采用 JB/ SADL,可以方便地进行软件体系结构的构造、细化和验证,并具有快速生成原型的能力,还支持代码框架的自动生成和系统体系结构的动态演化.

关键词: 软件体系结构; 层次消息总线; 体系结构描述语言
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2001) 05-0581-04

Hierarchical Message Bus-Based Software Architecture Description Language

ZHANG Shi-kun, WANG Li-fu, CHANG Xin, YANG Fu-qing
(Department of Computer Science and Technology, Peking University, Beijing 100871, China)

Abstract: Based on the practice of Jade Bird software production line, we propose a software architectural style based on hierarchical message bus named JB/ HMB, and design a corresponding architecture description language JB/ SADL. JB/ SADL describes a component from aspects of interface, static structure and dynamic behavior. Using JB/ SADL, we can construct, refine and verify the software architecture conveniently, which provides basis for the dynamic simulating execution, automatic generation of code framework and run-time evolution.

Key words: software architecture; hierarchical message bus; architecture description language

1 引言

随着软件系统规模和复杂性的增长,系统总体结构设计的重要性已远远超过了特定算法和数据结构的选择,良好的软件体系结构是保证系统成功的关键. 软件体系结构目前已成为软件工程研究的热点之一. 体系结构的设计位于系统开发的前期,在这一层次上的设计主要包括以下内容:系统中构件的描述,构件之间的交互,指导构件交互的风格,以及施加在风格上的约束等[1].

尽管良好的体系结构设计对保证系统的成功变得越来越重要,而且存在着许多有用的体系结构风格,但不可忽视的一个事实是:在目前大多数的工程实践中,(1) 体系结构的设计是建立在直觉和经验之上、而非坚实的工程原则之上;(2) 体系结构描述都是非形式化的和随意的,经常采用框线图(box-and-line diagram)加文字注释的方法:图中的方框表示构件,箭头线表示这些构件之间的联系. 这样的图表达了系统的高层概貌,标识了系统中主要的构件和数据/ 控制流,但通常对数据在计算/ 处理中所起的作用以及构件之间交互的细节缺乏规范的描述. 针对这些问题,研究人员正在寻求比较形式化的方法描述系统的体系结构,并且已经设计出了一些体系结构描述语言,典型的如 Acme[2],C2 SADL[3,4],Rapide[5,6],Unicon,Wright[1]等.

青鸟工程在"九五"期间,对基于构件 - 构架[1] 的软件工业化生产技术进行了研究,并实现了青鸟软件生产线系统[7]. 以青鸟软件生产线的实践为背景,提出了基于层次消息总线的软件体系结构风格(Jade Bird Hierarchical Message Bus - Based Style,简称 JB/ HMB 风格),并设计了相应的体系结构描述语言 JB/ SADL,开发了支持软件体系结构设计的辅助工具集,还研究了基于 JB/ HMB 风格进行应用系统开发的过程框架[8]. 本文重点讨论支持 JB/ HMB 风格的软件体系结构描述语言 JB/ SADL.

2 基于层次消息总线的体系结构风格 JB/ HMB

JB/ HMB 风格基于层次消息总线、支持构件的分布和并发,构件之间通过消息总线进行通讯,如图 1 所示. 消息总线是系统的连接件,负责消息的分派、传递和过滤,以及处理结果的返回;各个构件挂接在消息总线上,向总线登记感兴趣的消息类型;构件根据需要发出消息,由消息总线负责将该消息分派到系统中所有对此消息感兴趣的构件,消息是构件之间通讯的唯一方式;构件接收到消息后,根据自身状态对消息进行响应,并通过总线返回处理结果. 由于构件通过总线进行连

收稿日期:2000-10-11;修回日期:2000-12-10
基金项目:国家"九五"重点科技攻关项目基金(No. 98-780-01)

1 在本文中,"构架"和"体系结构"两个词是同义的,可以相互替换,都对应英文单词 architecture.

接,并不要求各个构件具有相同的地址空间或局限在一台机器上.该风格可以较好地刻划分布式并发系统,以及基于 CORBA、COM 和 EJB 规范的系统.

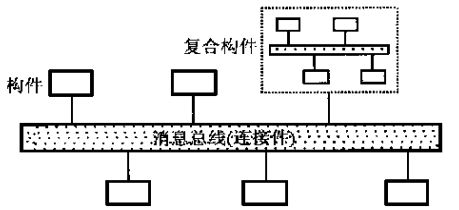

图 1 JB/ HMB 风格的系统示意图

如图 1 所示,系统中的复杂构件可以分解为比较简单的子构件,这些子构件通过局部消息总线进行连接,这种复杂的构件称为复合构件.如果子构件仍然比较复杂,还可以进一步分解.如此分解下去,整个系统形成了树状的拓扑结构,树结构的末端结点称为叶结点,它们是系统中的原子构件,不再包含子构件,原子构件的内部可以采用不同于 JB/ HMB 的风格,例如数据流风格、面向对象风格、管道 - 过滤器风格等[1],这些属于构件的内部实现细节,但要把它们集成到 JB/ HMB 风格的系统中,必须满足 JB/ HMB 风格的构件模型的要求,主要是在接口规约方面的要求.另外,整个系统也可以作为一个构件,通过更高层的消息总线,集成到更大的系统中.于是,可以采用统一的方式刻划整个系统和组成系统的单个构件.

3 JB/ SADL 概述

针对上述的 JB/ HMB 风格,设计了青鸟软件体系结构描述语言(Jade Bird Software Architecture Description Language,简称 JB/ SADL),支持对符合 JB/ HMB 风格的系统进行规范化描述,并可在工具的支持下,进行系统性质分析、动态模拟运行、动态演化和代码框架自动生成[8].

在设计 JB/ SADL 时,考虑到以下几个基本原则:

* 对系统和组成系统的构件进行统一描述;

* 构件的规约和实现相分离;

* 接口定义和构件规约相分离:构件和接口之间是多对多的关系,即一个构件可以提供一个或多个接口,而不同的构件又可以提供同样的接口;

* 构件规约是无嵌套的:类似于 C 语言中的函数定义,而不象 Pascal 中的子程序嵌套定义,以此提高构件的独立性和可复用性.

采用 JB/ SADL 描述一个系统的体系结构,包括一组接口、构件规约和构件实现,如图 2 所示.把作为系统的那个构件用 system as component - name 标识出来,即系统和构件采用了统一的描述方式.

```
architecture : : =
    {
      interface-spec                  / / 接口定义
      | component-spec                / / 构件规约
      | component-impl                / / 构件实现
    }
    system as component-name ;        / / 系统声明
```

图 2 JB/ SADL 的总体结构

4 构件规约

在 JB/ SADL 中,构件是系统的基本构造单元,具有较好的自包含性.每个构件对外提供了一个或多个不同的接口,接口定义了一组发送和接收的消息集合,刻画了构件对环境提供的服务和向环境要求的服务.此外,在构件规约中,还刻画了构件的静态结构和动态行为,如图 3 所示.

```
component - spec : : =
    component name provides interface-list is      / / 接口部分
      [structure-part]                             / / 结构部分
      [behavior-part]                              / / 行为部分
    end component ;
```

图 3 JB/ SADL 的构件规约框架

上述的构件规约同样适用于刻画整个系统.JB/ HMB 风格的系统是由构件通过消息总线连接而成的,其中的每个复合构件都是一个子系统,由更低层的构件通过局部消息总线连接而成;同时,系统也可以看作是一个大的复合构件,集成到更大的系统中.JB/ SADL 支持用统一的方法对构件和系统进行描述.在这个意义上,系统和构件只有层次上的差异,同我们考虑问题时所处的上下文相关,这符合人们认识客观世界的一般规律.

4.1 构件接口

在 JB/ SADL 中,一个构件可以提供一个或多个接口,构件接口定义如图 4 所示.接口定义给出了构件接收和发送的消息集合.在接口定义时,可以直接继承多个接口.在一般程序设计语言中,过程、函数或对象接口只定义了其对外提供的服务,而 JB/ SADL 的构件接口定义中还包括了要求的外部环境提供的服务,体现了互联接口的通讯完整性[9].

```
interface-spec : : =
    interface name[extends interface-list]is      / / 接口之间的继承
    send :
        {message-spec}                            / / 发出的消息集合
    receive :
        {message-spec }                           / / 接收的消息集合
    end interface ;
interface-list : : =
    interface-name { ,interface-name }
message-spec : : =
    [sync| asyn] message-name([parameter-list]) ;
parameter-list : : =
    parameter{ ,parameter}
parameter : : =
    direction [type] parameter-name
direction : : =
    in| out| in out
```

图 4 JB/ SADL 的构件接口定义

构件之间通过发送消息进行交互,消息可以是同步消息或异步消息,同步消息的发送者必须等待消息的处理结果返回后,才可继续运行;异步消息的发送者不必等待结果的返回,就可以继续运行,这样发送者和接收者可以并发运行.

消息中的参数分为三种:in、out 和 in out,分别代表输入、

输出和输入输出,in 参数是随消息一起发出的只读参数,out 参数保存了消息处理的结果,in out 参数同时具有两者的特点,缺省情况下是 in 参数.指明参数的传递方向不仅关系到不同的参数的处理方式,在分布式环境下,还可以减少网络上的数据传输量.对 in 参数而言,只需把数据从服务请求方传到服务提供方,out 参数则只需把结果数据从服务提供方传到服务请求方,只有 in out 参数需要在两个方向上传递数据.

由上可以看出,JB/SADL 提供的接口定义具有以下一些特点:

*支持互联接口,可以充分地表达构件和环境之间的关系.通过接口定义,表达了构件同环境的所有交互信息,减少了构件使用时对环境依赖的不确定性,提高了构件的复用潜力.

*方便构件集成.构件在实现时,如果需要外部环境提供的服务,只需使用在接口处定义的请求服务.在构件集成时,通过把构件请求的服务同其它构件(即环境)提供的服务匹配起来即可.

*支持对构件之间的同步和异步消息通讯的表达.

4.2　构件结构

在 JB/SADL 中,复合构件是由低层的构件通过局部消息总线连接而成的,其结构定义如图 5 所示.

```
reference :
    component-list ;                          // 引用的构件
consistof :
    {component-name instance-list ;}          // 构件实例声明
alias :
    {instance-name. message-name to new-name ;} // 消息过滤表
registry :
    {(bus-message ,instance-name) ;}          // 构件 - 消息响应登记表
```

图 5　JB/SADL 的构件结构定义

在上述构件结构定义中,

(1) reference 部分指明了所有引用到的子构件.

(2) consistof 部分指出了实际组成该复合构件的所有子构件实例,一个复合构件可能包含多个同类型的子构件实例.

(3) 由于不同来源的构件事先并不知道各自的接口,因此可能同一消息在不同构件中使用了不同的名字,或不同的消息使用了相同的名字,造成构件集成时的冲突和不匹配. alias 部分对构件发出和接收的消息进行简单地换名或阻塞某个消息,当新名字是 null 时,表示阻塞相应的消息.

(4) registry 部分登记了每个子构件实例感兴趣的消息,形成构件 - 消息响应登记表.对每个登记项中的 bus-message,构件实例 instance-name 中应有一个接收消息与此对应,即在消息名称、参数个数和类型、以及返回值上要一致.这里的 bus-messages 指的是复合构件接收的消息集合同所有子构件发送的消息集合的并集.

4.3　构件行为

软件系统以及组成系统的构件从本质上来说,都是信息的变换装置,其行为可以用带输出的有限状态机刻画.在 JB/SADL 中,采用 Mealy 机刻画构件的行为,一个 Mealy 机包括一组有穷的状态集合、状态之间的变迁和在变迁时发生的动作[10],采用五元组 (source-state, message, condition, method, target-state) 表示.其中,source-state 和 target-state 分别表示源状态和目标状态,为了实现从源状态到目标状态的转换,需要接收特定消息 message,满足特定条件 condition,并且执行特定方法 method.

构件行为定义部分刻画了构件对外界消息的响应,同构件接口定义和静态结构定义部分一起完整地对构件进行了规约,为构件实现或在可复用构件库中查找符合规约的构件提供了依据.

5　构件实现

为了支持系统的动态模拟运行和代码框架的自动生成,JB/SADL 还提供了描述构件实现部分的机制.在构件实现体中,包括属性声明和方法实现体.在 4.3 节采用五元组表示的状态变迁中,当构件接收到某个消息、且满足一定条件时,执行一个方法,这就隐含意味着方法的参数、返回值和消息的参数、返回值完全一致,即隐式地声明了方法.如果同一方法对多个不同的消息进行响应,那么这些消息的参数必须一致.

可以选择一种现成的面向对象编程语言刻画构件的实现,如 C++、Java 等,不过动态执行需要相应的编译器(或解释器)以及运行环境,而且具体实现的细节是后续开发阶段的任务,因此目前采用一种语法比较简单的伪码来刻画方法的实现.

语句的选择主要考虑如何描述构件之间的交互,而非每个构件的实现细节,用来刻画系统如何通过构件接口体现构件行为.围绕着消息发送语句(刻画构件之间的交互),还包括了一些其它的语句类型:赋值语句、条件语句、循环语句和执行语句,执行语句代表了所有其它不能用前面几种语句表达的情况,执行结果就是输出其后的正文串所表示的相关动作.限于篇幅,这里省略了具体语句的语法,可参见文[8]中的相关内容.

6　结论和进一步的工作

通过以上对 JB/SADL 特征的介绍和实例研究,可以看出,该语言较好地描述了符合 JB/HMB 风格的系统,并为系统静态性质分析、模拟运行、动态演化、代码框架生成和系统的集成组装提供了基础.目前已经开发了支持青鸟体系结构设计的辅助工具集 JB/SAKIT,包括建模工具、语法扫描器、静态性质分析工具、动态模拟运行工具和 CORBA 代码框架生成器[8].进一步的工作是针对不同的应用领域,对 JB/SADL 进行扩充,例如针对实时应用系统,在语言中加入对时间的描述,增强 JB/SADL 在特定领域内系统建模的能力.

参考文献:

[1] Mary Shaw and David Garlan. Software Architecture: Perspectives on an Emerging Discipline [M]. Prentice Hall, 1996.

[2] David Garlan, Robert Monroe, and David Wile. Acme: An architecture description interchange language [A]. In Proceedings of CASCON'97 [C], November 1997: 169 - 183.

高频地波雷达 OSMAR2000 通过验收

武汉大学电子信息学学院

武汉大学高频地波雷达课题组经过 2 年多的艰苦努力,自行研制了两台高频地波雷达 OSMAR2000.完成了国家 863 计划海洋领域的重大项目"高频地波雷达海洋环境监测技术"的研究项目,于 2000 年底通过国家科技部的验收.该项目研制的高频地波雷达 OSMAR 2000 采用一发八收相控天线,在中国大陆沿海首次实现双站岸基雷达监测海洋表面动力学参数,得到了海洋表面矢量流图.与传统海洋测量仪器的对比验证试验表明,OSMAR2000 探测海流和海面风向的距离超过 200 公里,探测浪高的距离达 150 公里.其整体性能已达到国外同类雷达先进水平,成为我国 863 计划海洋领域的一个"标志性成果".

高频地波雷达是近三十年来逐步发展起来的用于探测海洋表明动力学参数的一种新工具.它可以用来探测海洋表面的流场、风场、浪高等多种海洋动力学参数.与传统及现代的其它探测仪器相比,岸基高频地波雷达有其独特的优越性:一次探测面积大、实时性好、不受恶劣气候及海况的影响.

OSMAR2000 的硬件系统主要包括天线与馈电系统、八通道接收机、频率合成器、同步控制器、发射通道、固态发射机、数据采集器、信号处理机等组成.软件系统则是由控制与数据采集软件、海洋表面径向流提取软件、矢量流合成软件、风浪场反演软件及通信软件组成的软件包.

OSMAR2000 工作频率为 6～9MHz,采用线性调频中断连续波(FMICW)体制.天线阵以二元八木天线为基本阵列单元构成八单元的直线天线阵.使用位于天线阵中间的第 4 号或第 5 号天线作为发射天线,八根天线全部作接收天线,从而构成所谓"一发八收,收发共用"的工作模式.发射波束宽度约 120 度,采用数字波束形成(DBF)技术实现 15 度分辨率的波束扫描,利用"多重信号分类"(MUSIC)和"最小方差法"(MVM)算法实现回波到达方向的超分辨率计算.

利用一部 OSMAR2000 可以探测海洋径向表面流,利用相隔一定距离的二部 OSMAR2000 可以探测海洋矢量表面流.

(吴世才 柯亨玉供稿)

[3] Nenad Medvidovic. ADLs and dynamic architecture change [A]. In Proceedings of the Second International Software Architecture Workshop (ISAW-2) [C], San Fransisco, CA, October 1996: 24 - 27.

[4] Richard Taylor, Nenad Medvidovic, Kenneth Anderson, E. James Whitehead Jr., Jason Robbins, Kari Nies, Peyman Oreizy, and Deborah Dubrow. A component-and message-based architectural style for GUI software [J]. IEEE Transactions on Software Engineering, June 1996: 390 - 406.

[5] David Luckham, John Kenney, Larry Augustin, James Vera, Doug Bryan, and Walter Mann. Specification and analysis of system architecture using rapide [J]. IEEE Transactions on Software Engineering, Special Issue on Software Architecture, April 1995, 21(4): 336 - 355.

[6] David Luckham and James Vera. An event-based architecture definition language [J]. IEEE Transactions on Software Engineering, Special Issue on Software Architecture, April 1995, 21(4): 717 - 734.

[7] 杨芙清. 杨芙清文集 [M], 北京大学出版社, 1998, 5.

[8] Shikun Zhang. A study on hierarchical message bus based software architecture [D]. Doctoral Dissertation. Department of Computer Science and Technology, Peking University, June 2000.

[9] David Luckham, James Vera, and Sigurd Meldal. Three concepts of system architecture [R]. CSL-TR-95-674, Stanford University, July 1995.

[10] John Hopcroft, and Jeffery Ullman. Introduction to Automata Theory, Languages, and Computation [M]. Addison-Wesley Publishing Company, 1979.

[11] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice [M]. Addison-Wesley Publishing Company, 1997.

作者简介:

张世琨 博士,北京大学计算机科学技术系讲师.目前的主要研究领域为软件工程,软件体系结构.

王立福 博士,北京大学计算机科学技术系教授,博士生导师.目前的主要研究领域为软件工程,信息安全.