基于服务连接和消息连接软件构架的研究

任洪敏 朱 承 钱乐秋

(复旦大学 计算机科学系 软件工程实验室，上海 200433)

摘 要：软件构架是解决复杂大型软件开发面临的困难、提高软件质量和软件生产效率的有效方法，是软件复用和基于构件软件工程研究的重要领域.通过对软件构架风格特点的讨论和对构件之间交互方式的分析，论文提出了基于服务连接和消息连接的软件构架风格，开发了一个支持该构架风格的构件生产和组装平台.该构架风格具有以下特点：(1)支持多种可变性机制；(2)构件之间灵活和显式的服务连接和消息连接机制；(3)支持构件合成；(4)构件接口分组，成为端口.

关 键 词：软件构架；构架风格；构件合成；CASE 工具

中图分类号：TP311 **文献标识码**：A **文 章 编 号**：1000-1220(2003)04-0744-05

Research on Software Architecture Based on Service Connection and Message Connection

REN Hong-min, ZHU Cheng, QIAN Le-qiu

(*Department of Computer Science and Technology Fudan University , Shanghai* 200433,*China*)

Abstract: Development based on software architecture is one of the most effective solutions to improve software quality and productivity, and minimize the difficulties of developing large and complex systems. Therefore, software architecture is one of the key research areas of software reuse and component-based software engineering. Based on the analysis of characteristic properties of good architecture styles and the illumination of various forms of interaction between components, an architecture style based on service and message connection is introduced in this paper. An accompanying component production and assembly platform is developed for support and verification purposes. Briefly, this architecture style distinguishes itself by the following features: (1) support of diverse variability mechanisms, (2) explicit and flexible service and message connection mechanisms between components, (3) support of component composition, (4) grouping of component interfaces to ports.

Key words: software architecture; architecture style; component composition; CASE tool

1 引 言

软件构架着眼于软件系统的全局组织形式，在高层次上把握系统各部分之间的内在联系，从整体的角度去理解和分析整个系统的行为和特性，解决当前开发复杂大型软件系统所存在的困难，提高软件质量和效率，降低软件开发成本[1].软件构架技术及作为其支撑工具的构件生产和组装平台，是软件产业工程化和工业化的核心技术[2].

国内外的众多研究机构和学者对软件构架技术和其支撑工具进行了广泛的研究，他们研制和开发的系统各具特点，并应用于不同领域.如北京大学基于“构件—构架”软件工业化技术，研制开发了应用系统集成组装环境“青鸟 III 系统”[3].吉林大学提出了基于角色模型的软件构架风格和基于主动连接件的软件构架风格[4,5].中国科学院软件研究所用时序逻辑语言描述软件构架和开发了相关可视化支撑工具[6,7].Richard N . Tayler 等研制开发了用于 GUI 软件开发的基于消息总线的 C2 构架风格[8]，Mazy Shaw 等开发了利用枚举方式定义构件和连接器类型的 Unicon 系统[9]，J. Magee 等研制了用于分布式开发环境的 Darwin 系统[10].其它国际上有影响的软件构架研究包括Wright[11]、ACME[12]、Rapide[13]等.

构架模型的三个要素是构件、连接器和构架配置，即构架的拓扑结构[14].一个好的构架模型应该具备下述特点：

(1) 构件模型应支持多种可变性机制，因而相应的CASE工具应该支持构件生产时构件的泛化和组装时的客户化.

(2) 应该全面支持构件的合成，包括功能的合成、可变性机制的合成、实现的合成.

(3) 构件功能接口元素，如构件发出的通知消息或提供的服务操作，应当分组.当构件复杂时，提供数十个功能接口元素，如不分组，则变得复杂，不利于构件的理解、组装和复用.

(4) 连接器既要有丰富的语义表达能力，能够表达构件之间的交互关系，又要能够自动生成和复用.否则不利于软件系统的组装和构件的合成.

收稿日期：2001-07-17 基金项目：上海市教委重点学科建设项目资助. 作者简介：任洪敏，博士研究生，主要研究领域为软件复用.朱 承，博士研究生，主要研究领域为软件复用.钱乐秋，教授，博士生导师，主要研究领域为软件自动化，软件复用，软件自动测试.

我们吸收前述研究工作的优点,并注意到它们或者它们中的部分,对上述特点支持的不足,提出了一个基于服务连接和消息连接的软件构架模型 SASCMC(software architecture based on service connection and message connection),开发了一个支持 SASCMC 构架风格的构件生产和组装平台原形. SASCMC 构架模型实现了上述(1)～(3)项要求. 连接器的研究是构架研究的一个难点,SASCMC 支持现实世界和软件世界中实体交互的两种最常见的方式即消息连接和服务连接,基本满足了第(4)项要求.

2　服务连接和消息连接

服务连接和消息连接,是现实世界和软件世界中实体联系和交互的两种重要方式.

2.1　服务连接

服务连接是一个构件提供某项服务,其它构件请求和应用该项服务,并且在该项服务执行完毕之后,请求构件才能继续执行. 这种请求和提供服务模式是构件之间交互的最常见模式,COM 构件的交互采用该种模式[15]. 服务连接的特点是:

(1) 反映交互实体间紧密耦合的关系,服务请求必须存在对应的服务提供.

(2) 服务请求和提供具有同步关系,服务请求者必须等待该项服务执行完毕.

(3) 服务请求者和服务提供者通常是多对一的关系,即多个请求者可以请求一个服务提供者提供服务.

(4) 服务的请求与提供交互中,信息能够在交互的双方之间,双向流动.

2.2　消息连接

消息连接,也称为隐式调用(Implicit Invocation)或事件广播(Event Dissemination)交互机制[1],是指一个构件因其自身内部状态发生变化,它向外界发出一个通知消息,宣告自己的变化信息,对该消息感兴趣的构件可自行响应和处理该消息. 这种消息的通知和响应模式是构件之间交互的另一种常见模式,C2 构架风格采用该种交互模式[8],其它应用包括数据库系统和 CASE 工具集成[16]. 消息连接的特点是:

(1) 反映实体松散耦合的关系,一个通知消息可能不存在对应的响应行为.

(2) 消息通知和响应具有异步关系. 消息通知者不等待消息响应者执行.

(3) 消息通知者和消息响应者通常是一对多的关系,即一个通知消息可有多个响应者.

(4) 消息的通知与响应交互中,信息只能由消息的发送者向消息的接受者单向传递.

消息连接的好处在于系统易于扩充和进化,弱点在于对系统执行缺乏控制,服务连接和消息连接,相互补充[1],SASCMC 支持这两种交互模式.

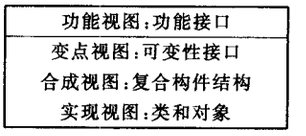
3　SASCMC 构架模型

我们从构架风格的三个要素即构件、连接器和构架配置[14]三个方面阐述 SASCMC 构架模型.

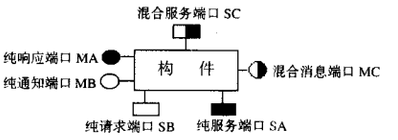
3.1　构件模型

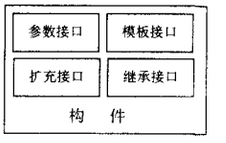
构件是相对独立的功能计算单位. 采用不同的软件开发方法,在不同的软件抽象层次上,构件的形态和表示等也不相同,如从软件开发过程看,有分析构件、设计构件、代码构件和测试构件等. 从实用性和完整性考虑,SASCMC 仅考虑符合 OO 风范的源代码级和目标代码级构件.

SASCMC 建立一个多视图构件模型,易于理解和复用(如图 1). 功能视图,提供构件的功能,构件通过功能接口连接,彼此协作,完成需求的功能. 变点视图,提供构件的可变点,支持构件的泛化和客户化. 合成视图,提供复合构件的内部结构及其外部视图和内部组成元素间的关系,具体见第 4 节构件合成. 实现视图,提供构件的具体实现,是符合 OO 风范的类和对象.



(a) 构件的多视图模型





(b)构件的功能接口　　(c)构件的可变性接口

图 1　SASCMC 构件的模型

Fig. 1　SASCMC component module

3.1.1　*构件功能视图*　构件的功能接口是构件提供给外界的功能视图,构件通过各自的功能接口彼此交互和通信,协同完成系统要求的工作. SASCMC 构件的功能分为两类,即服务接口和消息接口,以支持服务连接和消息连接. SASCMC 构件把服务接口和消息接口根据功能内聚原则,进行分组,每组即为一个端口. 分组的好处在于,一方面能够让我们不再只关注构件单个的操作和消息,而关注功能内聚的操作和消息组之间的交互模式,具有更高的抽象程度,另一方面在于构件之间通过端口进行连接,结构清晰.

SASCMC 构件的端口相应地分为两类,一类是服务端口,一类是消息端口(见图 1(b)). 一个构件具备零个或多个服务端口,每个服务端口是紧密耦合的一组相关操作的集合. 服务端口中的操作具有 PROVIDE 属性或 REQUEST 属性,具有 PROVIDE 属性的操作表示该操作是构件向外提供的服务,具有 REQUEST 属性的操作表示该操作是构件为了完成自己的功能,需要向外界其它协作构件请求提供的服务. 仅由具有 PROVIDE 属性的提供服务操作构成的端口,称为纯服务端口,构件表示图形中用填充的矩形表示. 仅由具有 REQUEST 属性的请求服务操作构成的端口,称为纯请求端口,构件表示图形中用未填充的矩形表示. 一个服务端口既有具有 PROVIDE 属性的提供服务操作,又有具有 REQUEST 属性的请求服务操作,则称为混合服务端口,构件表示图形中用

一半填充的矩形表示.

一个构件能够具备零个或多个消息端口,每个消息端口是一组紧密耦合的消息的集合.消息端口中的消息具有 NOTIFY 或 RESPONSE 属性,具有 NOTIFY 属性的消息表示是该构件向外发出的消息,具有 RESPONSE 属性的消息表示是该构件能够响应的消息.仅由具有 RESPONSE 属性的响应消息构成的端口,称为纯响应端口,构件表示图形中用填充的圆形表示.仅由具有 NOTIFY 属性的通知消息构成的端口,称为纯通知端口,构件表示图形中用未填充的圆形表示.一个消息端口既有具有 RESPONSE 属性的响应消息,又有具有 NOTIFY 属性的通知消息,则称为混合消息端口,构件表示图形中用一半填充的圆形表示.

为了构件灵活地合成和组装,SASCMC 系统支持动态构件端口.构件服务端口和消息端口拥有的操作和消息由构件生产者初步确定,但构件的复用者能够对其重新调整,比如把一个端口中的操作和消息移动到另一个端口中,端口的合并,端口的再分割,即把一个端口分为几个端口.根据后面介绍的端口连接机制和服务请求提供交互、消息通知响应交互的语义,一个提供服务操作可以包容于构件的多个服务端口中,但一个请求服务操作只能包含于构件的一个服务端口中,一个通知消息和响应消息皆可包含于多个消息端口中.

3.1.2 *变点视图* 同一般软件产品相比,构件是可复用的软件产品资源,它的生产和维护需要更多的费用,甚至数倍到十倍的费用.故构件必须具备高度抽象,应尽可能地通用化,并且构件复用人员根据特定软件开发需要,易于对其进行客户化,从而增大构件的复用机会,降低应用系统开发和维护成本.构件的生产者在构件生产时利用可变性分析机制对其进行泛化,得出构件的变点,并把变点通过可变性接口提供给构件复用者,让构件复用者在构件复用时对其进行客户化,以满足复用者的特定需要和具体应用环境.SASCMC 系统支持四种可变性机制[17].

第一种可变性机制是参数化,用于小的变化点,通常是基本数据类型.第二种可变性机制是模板化,用于构件数据类型的参数化.第三种可变性机制是继承,该机制可以重载构件的接口方法.第四种可变性机制是扩展,扩展点可有多个供选择的扩展体,扩展体通常是一个简单构件,具有封装的数据和能够提供多个操作服务.

3.2 连接器模型

连接器是软件构架中一个非常重要的概念,它表示构件之间的交互方式.连接器可能具有非常灵活的表现形式,如控制流、信息流、数据流、过程调用、消息机制、动态数据结构、管道等或者更一般的协议.SASCMC 系统克服 C2 等系统支持单一连接器、结构简单的不足,支持三种连接器,它们具有不同的语义,支持构件间最常见的交互方式,利用它们能构造结构复杂的系统,满足很多应用领域的需求.且这三种连接器皆有独立的、可复用的实现代码,易于系统跟踪和构件组装.

3.2.1 *扩展连接器* 可变性接口中,扩展机制既是一种可变性机制,实质也是一种连接机制.作为变体的构件通过扩展点,插入另一个构件,实现两个构件的连接,变体构件向变点所在构件提供功能和扩展其功能.但变体构件通常功能单一,局部于变点所在构件,不影响系统全局拓扑结构.

3.2.2 *服务连接器* 服务连接器连接构件的服务端口,对两个端口中的请求服务操作和提供服务操作进行第三方绑定[18],从而让服务请求者和服务提供者彼此独立,易于替换和进化,并且整个系统结构清晰明了,易于理解.

服务连接器同时解决请求服务操作和提供服务操作规约失配问题,担当适配器角色,转换不一致的参数类型、调整不一致的参数顺序、为未提供的参数提供缺省值等.

3.2.3 *消息总线连接器* 消息总线连接多个构件的消息端口,进行消息转发,实现隐式过程调用机制,同时解决通知消息和响应消息操作规约失配问题.

本系统实现了四种消息总线.第一种是非过滤器总线,该总线直接把接收到的消息转发到连接其上的构件.第二种是过滤器总线,该总线能够根据设定的条件,对不满足条件的消息进行过滤.第三种是优先级消息总线,该总线依据一定的优先级原则把消息转发到相应的构件.最后一种类型的消息总线是消息池,它能够根据一个开关,忽略送到它的消息,用于隔离子系统和系统演化.

3.3 构架配置模型

构架配置是构件通过连接器连接形成的软件拓扑结构及约束.构架配置需要决定连接的构件是否恰当、接口是否匹配、构件间的通信方式、连接语义和配置约束等方面的问题.SASCMC 系统利用其连接器,提供两种构架间的连接方式.

3.3.1 *服务端口连接* 服务端口的连接是指一个构件的服务端口与另一个构件的服务端口通过服务连接器进行连接.其实质是两个服务端口中所有具备 REQUEST 属性的请求服务操作与其相连端口中相应的具有 PROVIDE 属性的提供服务操作两两连接,表示请求的服务由相应的提供服务操作提供,两者协作完成工作.服务端口连接时提供服务操作和请求服务操作参数个数、参数类型等接口失配问题,由服务连接器自动或者在构架设计师的指导下解决.

请求服务操作和提供服务操作两者勿需按名匹配,可由构架设计师在图形界面上调整两个端口中服务操作的次序,从而让它们依次一一对应,或者由构架设计师明确指定一个请求服务操作与相连端口中哪一个提供服务操作相连.在相互连接的两个服务端口中,其中任一个端口中的请求服务操作必定在另一个端口中存在对应的提供服务操作与其相连,但反之并不一定成立,即两个端口中都可存在对方不需要的提供服务操作.混合服务端口只能与一个相应的服务端口连接,它们间的耦合关系紧密.纯服务端口可与多个相应的服务端口相连,构成客户—服务器模式,向多个端口提供服务,它们之间是一种较为松散的耦合关系.纯请求端口只能与一个相应的服务端口相连.

3.3.2 *消息端口连接* 多个构件的消息端口能够通过消息总线连接器进行连接.表示其中一个消息端口中发出的通知消息,经过消息总线的过滤策略裁决,或者经由构架设计师指定,由连接在同一条消息总线上的零个或多个其它构件的消息端口中相应的响应消息进行响应.一个构架中可以存在多

条消息总线,一个构件的多个消息端口可以同时连接在不同的消息总线之上.不同的消息总线之间也可以进行连接,由一个消息总线转发到另一个消息总线上的通知消息和其相应的响应消息由构架设计师指定.消息端口连接时通知消息和响应消息参数个数、参数类型等接口失配问题,由消息总线连接器自动或者在构架设计师的指导下解决.

消息端口中既可以存在着没有其它构件响应的通知消息,也可存在着没有相应通知消息的响应消息.构件之间通过消息端口进行连接,是一种松散的耦合关系.

SASCMC 构架风格为给构架设计师提供充分的灵活性和方便,未对构件之间的连接和由此形成的系统拓扑逻辑结构的形态做过多的限定,但良好的设计,应使系统具有优美的拓扑逻辑结构,符合常见构架设计模式,如层次结构、或星形结构.构架配置模型参见图 2.

4　构件合成

构件合成指由多个基本构件或子复合构件通过一定的连接方式组合而成一个单独的构件,称为复合构件.通过显式地表示复杂的结构和行为,或把其抽象为一个单独的构件,构件合成机制允许构架设计师在不同的抽象和详细级别上描述软件系统,因此应是构架建模机制的一个重要组成部分.

复合构件内部组成元素之间的结构亦是一个构架,SASCMC 系统采用统一方式对复合构件结构和构架进行描述,即复合构件内部元素之间的连接方式遵循构架模型中构件间的连接方式.并且复合构件和基本构件都遵循相同的构件模型.故我们从构件模型的三个视图即功能视图、变点视图、实现视图三个方面描述 SASCMC 构架风格严密的构件合成机制.

4.1　功能接口合成

根据构架配置模型中端口连接的原则和约束,复合构件功能接口合成方式及其原则如下:

(1) 纯服务端口,无论其与内部其它构件的服务端口进行连接与否,皆可直接输出作为该复合构件的服务端口,但不必一定输出.

(2) 混合服务端口和纯请求端口,如未与内部其它构件的端口进行连接,则其一定要输出作为该复合构件的一个服务端口,但如果它们已与复合构件内部的其它端口进行了连接,则其不能输出.

(3) 消息端口,无论其与内部其它构件的消息端口是否通过消息总线进行连接,皆可直接输出作为该复合构件的消息端口,但不必一定输出.

(4) 按照上述原则输出的功能端口,既可以直接输出作为复合构件的功能端口,但它们也可被构架设计师、构件生产者利用合并、分割、裁减等方法,重新分组,形成复合构件一个或多个新的端口.所谓裁减,就是指内部构件的某些提供服务操作、通知消息和响应消息不输出到复合构件的功能端口之中,请求服务操作不能被裁减.

举例说明 SASCMC 构架模型中构件连接方式和复合构件功能接口合成方式(见图 2).图中外层矩形表示是一个复合构件,图中三条虚线表示复合构件的消息端口 MC 是由三个内部构件 C、内部构件 D、内部构件 E 的三个相应消息端口复合裁减而成.

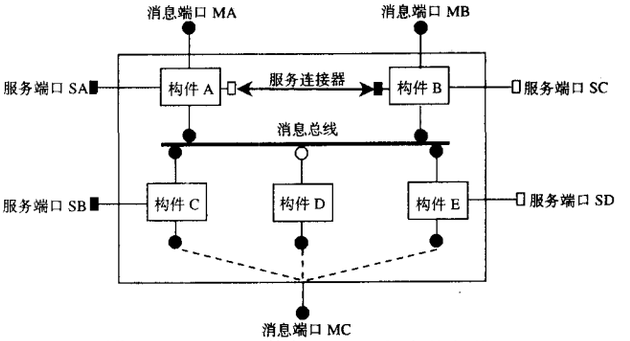


图 2　SASCMC 构架配置和复合构件模型

Fig. 2　SASCMC architecture configuration and composition component

4.2　可变性端口的合成

所有内部构件的可变性端口中所有的可变点,除了构架设计师、构件生产者已经为其指定了实例化参数,进行了客户化之外,皆应输出作为该复合构件的可变点.当复合构件进行客户化时,复合构件把外界提供给自己的变体,提供给相应的内部构件的可变点,对其进行客户化.

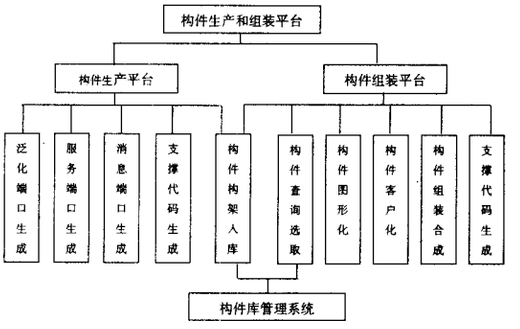


图 3　构件生产和组装平台系统结构

Fig. 3　Accompanying component production and assembly architecture

4.3　实现合成

因为遵循面向对象风范,故利用对象包容机制实现复合构件的实现合成.实现时,为复合构件自动生成一个类,内部构件定义为复合构件类的成员变量,成为复合构件的内部对象,所有复合构件的接口操作皆委托给内部对象.

5　构件生产和组装平台设计和实现

为了支撑 SASCMC 构架风格模型,我们设计和开发了一个相应的构件生产和组装平台原型和一个应用实现系统,对其进行支持和验证.该平台方便构件生产者生产符合 SASCMC 风格的可复用构件,方便构架设计师进行系统构架设计.构件生产者能够通过图形界面、鼠标点击、简单编辑,运

用该平台内部支持的代码自动生成机制，把一个普通的类簇转变为符合 SASCMC 构件模型的构件，支持服务端口、消息端口和可变性接口. 构架设计师能够运用构件组装机制、构件合成机制、连接器代码自动生成机制、约束检测机制，通过可视化界面，设计和生成应用系统. 构件生产和组装平台生成的基本构件、复合构件和系统构架，经过打包，提供构件相关属性和刻面描述，存入构件库，供复用.

构件生产和组装平台的系统整体结构见图 3.

6 结 论

软件构架技术和作为其支撑工具的构件生产和组装 CASE 工具，是软件工程界研究的一个热点，是基于构件软件工程和软件复用研究中的一个关键技术. 论文讨论了一个好的构架风格应该具备的四个特点，仔细分析了服务连接和消息连接的特性，提出了一个新的构架风格，即基于服务连接和消息连接软件构架 SASCMC，并开发了一个遵循该风格的构件生产和组装平台原型.

SASCMC 构架风格提供可供追踪的消息总线连接器和服务连接器，支持现实世界和软件世界中实体交互的两种常见模式：支持多到一通信机制的服务连接器和支持一到多通信机制的消息连接，具有强的建模能力. 消息总线连接器和服务连接器同时解决构架失配问题和支持构件交互双方独立，从而易于构件的复用. SASCMC 构架风格支持多种可变性机制，让其具有更强的可复用性.

SASCMC 构架风格支持构件服务接口和消息接口分组，并且允许根据需要，动态调整分组，从而让构架设计师关注功能内聚的操作组和消息组之间的交互模式，不再只是关注构件单个的操作和消息，因而具有更高的抽象程度，同时构件之间通过端口进行连接，而不是通过许多单个操作和消息进行连接，因而结构清晰. SASCMC 还支持构件合成、支持面向对象风范，提供了严格的构件合成和组装机制. 利用 SASCMC，能够开发具有复杂结构的应用系统，适合多种应用领域软件的开发. 当然，SASCMC 也存在不足. 今后，我们将在下面两个方面继续展开工作：

(1) 利用 SASCMC，开发构件和应用系统，积累更多实践经验，完善 SASCMC 模型，如支持动态构架建模.

(2) 引入形式化方法，严格描述 SASCMC 构架风格，支持更多约束检测和相关特性分析.

Reference:

1 Shaw M, Garlan D. Software architecture: perspectives on an emerging discipline[M]. Prentice Hall, Inc., Simon & Schuster Bejing Office, Tsinghua University Press, 1996

2 Yang Fu-qing, Mei Hong, Li Ke-qin. Software reuse and software component technology [J]. ACTA ELECTRONIC SINICA. 1999.27(2):68～75

3 Yang Fu-qing, Mei Hong, Li Ke-qin et al. An introduction to JB3 system supporting component reuse [J]. Computer Science. 1999.26(5):50～55

4 Feng Tie, Zhang Jia-chen, Chen Wei, Jin Chun-zhao. Software architecture specification based on a framework and role type[J]. Journal of Software. 1999. No.

5 Zhang Jia-chen, Feng Tie, Chen Wei, Jin Chun-zhao. Active-connector-based software architecture and its description method [J]. Journal of Software. 2000.11(8):1047～1052

6 Luo Hua-jun, Tang Zhisong, Zheng Jiandan. Visual architecture description language XYZ/ADL[J]. Journal of Software. 2000.11(8): 1024～1029

7 Jiao Wen-pin, Shi Zhong-zhi. Formalizing architecture styles with XYZ/E[J]. Journal of software. 2000,11(3):410～415

8 Talor R N, Medvidovic N, Anderson K M et al. A component- and message-based architectural style for GUI software [J]. IEEE Transactions on Software Engineering. 1996. 22(6):390～406

9 Shaw M, DeLine R, Klen D V et al. Abstractions for software architecture and tools to support them[J]. IEEE Transactions on Software Engineering. 1995. 21(4):314～355

10 Magee J, Kramer J. Dynamic structure in software architectures [A]. In: Kaise G E ed. Proccdings of the ACM SIGSOFT'96: the 4th Symposium on the Foundations of Software Engineering. New York: ACM Press, 1996. 3～14

11 Allen R. A formal approach to software architecture [D]. PhD thesis, Carnegie Mellon Univ., CMU Technical Report CMU-CS-97-114, May 1997

12 Garlan D, Monroe R, Wile D. ACME: an architectural interconnection language [R]. Technical Report, CMU-CS-95-219, Carnegie Mellon University, 1995

13 Luckham D C, Vera J. An event-based architecture definition language[J]. IEEE Transactions on Software Engineering. 1995. 21(9):717～734

14 Nenad Medvidovic, Richard N. Taylor. A classification and comparison framework for software architecture description languages [J]. IEEE Transactions on Software Engineering. 2000. 26(1):70～93

15 Eddon G, and Eddon H. Inside distributed COM[M]. Microsoft Press. 1998

16 Steve P. Reiss. Connecting tools using message passing in the Field Environment. IEEE Software, 1990, 7(4):57～66

17 Ivar Jacobson, Martin Griss et al. Software reuse[M]. ACM press, 1977

18 Crane S, Dulay N et al. Configuration management for distributed software services[A]. In Y. Raynaud, A. sethi, F. Faure-Vincent, edtors. Integrated Network Management IV[M]. Chapman and Hall, 1995. 29～42.

附中文参考文献：

2 杨芙清，梅宏，李克勤. 软件复用与软件构件技术[J]. 电子学报. 1999.27(2):68～75

3 杨芙清，梅宏，李克勤，袁望洪，吴穹. 支持构件复用的青鸟 III 型系统概述[J]. 计算机科学. 1999.26(5):50～55

4 冯 铁，张家晨，陈伟，金淳兆. 基于框架和角色模型的软件体系结构规约[J]. 计算机学报. 2000.11(8):1078～1086

5 张家晨，冯铁，陈伟，金淳兆. 基于主动连接件的软件体系结构及其描述方法[J]. 计算机学报. 2000.11(8):1047～1052

6 骆华俊，唐稚松，郑建丹. 可视化体系结构描述语言 XYZ/ADL [J]. 计算机学报，2000，11(8)：1024～1029

7 焦文品，史忠植. 用 XYZ/E 形式化体系结构风格[J]. 计算机学报. 2000.11(3):410～415