基于构件的软件框架与角色扩展形态研究*

刘 瑜+, 张世琨, 王立福, 杨芙清

(北京大学 信息科学技术学院,北京 100871)

Component-Based Software Frameworks and Role Extension Form

LIU Yu+, ZHANG Shi-Kun, WANG Li-Fu, YANG Fu-Qing

(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: 86-10-62751186, Fax: 86-10-62751187, E-mail: liuyu@urban.pku.edu.cn

http://www.cs.pku.edu.cn

Received 2003-01-17; Accepted 2003-03-05

Liu Y, Zhang SK, Wang LF, Yang FQ. Component-Based software frameworks and role extension form. *Journal of Software*, 2003,14(8):1364~1370.

http://www.jos.org.cn/1000-9825/14/1364.htm

Abstract: Framework is an important approach to large-grained software reuse. Object-Oriented frameworks are widely used through instantiation, but some shortcomings of object-oriented framework were mentioned during related researches and practices. Component-Based software framework (CBSF), which belongs to black-box framework, is adopted to solve these problems. In CBSF, component is chosen to substitute for collaborated class to increase the granularity of framework's element, and hot spot is implemented by component interface calling and components composition. Role extension form of CBSF is discussed, in which role is a special component that need to be instantiated, and can be categorized into abstract component and template. During the CBSF develop process, architecture patterns, design patterns and parameterizations approaches are helpful to design role extension for different domain variability types. Some issues on composing application components based on the role extension form are also discussed in this paper.

Key words: component-based software framework; hot spot; role; variability

摘 要: 框架是实现大粒度软件复用的有效途径,面向对象框架可以通过实例化扩展实现具体的应用系统,但是,研究和实践表明,面向对象框架存在着一些问题.采用基于构件的框架(CBSF),通过用构件替代框架内部相互协作的类,以增大框架构成成分的粒度;同时利用构件的接口调用和组装实现框架扩展机制.主要探讨了角色扩展形态,角色是将要被进一步实例化的特殊构件,可分为抽象构件和模板.针对不同变化性类型,提出可以利用体系结构模式、设计模式或参数化这 3 种方式设计 CBSF 角色扩展,进而讨论了基于角色扩展形态的应用构件组装.

关键词: 基于构件的框架;扩展点;角色;变化性

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2001AA113171 (国家高技术研究发展计划(863))

第一作者简介: 刘瑜(1971－),男,山东诸城人,博士生,讲师,主要研究领域为基于构件的软件开发,软件工程技术.

中图法分类号: TP311 文献标识码: A

近年来,随着软件复用研究的深入,软件框架也日益受到人们的重视.框架是指一个可复用的、部分实现的软件制品,它能够被实例化扩展,以生成特定的应用.从领域工程的角度来看,框架是实例化的 DSSA (domain specific software architecture,特定于领域的软件体系结构),它反映了一个软件系统族的体系结构,并且提供了创建后者的基本构造单元,同时定义了针对特定的功能需要在何处进行调整和修改[1],即扩展点(hot spots).软件框架有助于实现领域内体系结构层次较大粒度的设计复用,提高应用开发中复用的比例,从而保证复用活动的成功率,降低应用开发的成本.

20 世纪 80 年代以来,许多学者围绕框架的定义、开发、实例化进行了大量研究,一些软件公司也实现了特定于领域的框架,如 Taligent 公司的 CommonPoint[2]和 IBM 针对商业领域的 San Francisco*等等.综合各种对于软件框架的定义和描述,可以发现框架具有以下几个特点:(1) 框架是面向特定领域的,它构成了软件产品线的核心资产;(2) 框架是 DSSA 的实例,具有部分实现的特性,它反映了产品线中应用的体系结构;(3) 框架由一组协作的成分构成;(4) 利用框架开发应用系统是通过扩展点的实例化过程实现的.

从复用的角度来划分框架,可以分为白盒框架(white-box framework)和黑盒框架(black-box framework)[3],从复用成本上来看,白盒框架高于黑盒框架,但是后者设计难度更大.本文采用基于构件的框架(component-based software framework,简称 CBSF)概念,就框架元模型、框架扩展形态、扩展形态与领域变化性的关系、构件组装等课题进行探讨,从而为开发框架提供有益的指导.

1 基于构件的软件框架

1.1 概念的提出

文献[4]中指出,软件框架在其生命周期开始时一般是白盒框架,随着对领域认识的不断加深,白盒框架将逐渐演化成为黑盒框架,从而提高对复用活动的支持.目前,实践中的白盒框架主要是面向对象框架(包括前述的 San Francisco 和 CommonPoint 框架),它是构成一类特定软件可复用设计的相互协作的一组类[3],包括具体类和抽象类,其实例化方式主要通过子类化抽象类以及对现有具体类方法的调用.随着 OO 框架的开发和使用,许多学者也意识到了面向对象框架的不足,主要集中于以下 3 点:首先是框架的“过度增殖问题”,通过实际的调查表明,在许多基于 OOF 实例化实现的软件系统中,是将变化性“硬编码”到框架中,这就造成了框架的大量“增殖”,使框架维护变得困难[5].其次,“脆弱的基类(fragile base class)”问题,面向对象框架的开发者并不清楚在框架被分发后,用户将进行何种扩展,当框架开发者修改框架基类时,就有可能损害用户所做的扩展.第 3,“隐式的体系结构(implicit architecture)”问题.当用户使用一个框架时,首先需要了解其体系结构,然而由于框架的实现表现为具体的、在代码层次提供复用的类,因此框架的体系结构被“隐没”在类的实现细节中而难以被识别[6].

上述 OO 框架的不足可以归结为两个方面的原因,即框架类的粒度过小和继承扩展机制.因此,本文通过用构件接口的调用来替代类方法的重载,将“继承”变为“组装”,形成基于构件的框架,使其支持黑盒复用方式.与面向对象框架相比,CBSF 更大的构件粒度及其基于组装的扩展机制,使得它更易于被复用.目前对于 CBSF 研究还相对较少,实践工作主要有 Avalon 项目**、概念知识处理框架 Tockit***以及交互训练框架[7]等等,它们都是基于 Java Bean 或 EJB 的 CBSF 框架.

CBSF 具有黑盒框架易于复用的优点,并避免了 OO 框架的一些不足.在基于 CBSF 的构件组装过程中,框架为构件提供了一致的可复用环境,从而降低了构件组装的难度.此外,因为基于构件的框架定义了构件的规约,

* http://www.ibm.com/software/ad/sanfran-cisco/

** http://jakarta.apache.org/avalon/

*** Plüschke A. Proposals for Tockit: A Component Based Framework for Conceptual Knowledge Processing. http://tockit.sourceforge.net. 2001.

并同时提供实现后者的模板,也使开发新构件变得容易.

1.2 基于构件的框架元模型

基于构件的框架由一组互相协作的构件组成,通过这些构件及其协作关系定义了应用系统的体系结构;同时框架通过扩展点组装用户开发的构件,以处理领域变化性.

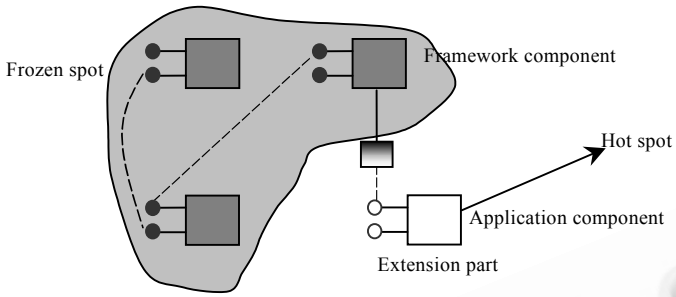

Fig.1 Structure of component based framework

图 1 基于构件的框架结构

如图 1 所示,框架内部的构件称为框架构件(framework components),它实现了领域共性.而在框架外部的、由用户定制的、待组装的构件称为应用构件(application compo-nents),它表现了具体应用系统的特性,即领域变化性;框架复用时需要根据应用的具体特点,组装不同的应用构件.

对于基于构件的框架而言,扩展点是框架中预先定义的一些"点",在框架复用中应用构件的组装需要基于扩展点进行.构造性和演化性是软件的两个本质特征****,作为一类重要的可复用软件制品,CBSF 的构造性表现为框架构件及其协作关系;而基于扩展点可以组装不同的应用构件以适应领域的变化性,则体现了框架对于软件演化特征的支持.

基于构件的框架可以定义为以下的五元组:

CBSF={framework components, connections, constrains, design patterns, hot spots}.

从结构上看,框架构件及其连接形成了框架的体系结构,框架体系结构在被实例化时决定了具体应用的总体结构.此外, CBSF 还包含了一系列约束(constrains),包括框架内部的控制流程、构件之间的协作关系以及框架对于扩展的限定等等,这些约束往往与领域有关.设计模式(design patterns)描述了针对特定问题,为了处理某种变化性而采取的构件局部协作方案,它既包括通用模式,也包括特定于领域的模式,反映了框架面向特定领域的特点;而扩展点则反映了框架部分实现和需要针对具体应用进行扩展的特性.综上所述,并参照文献[8]给出的 OOF 元模型定义,在此给出 CBSF 元模型,如图 2 所示.

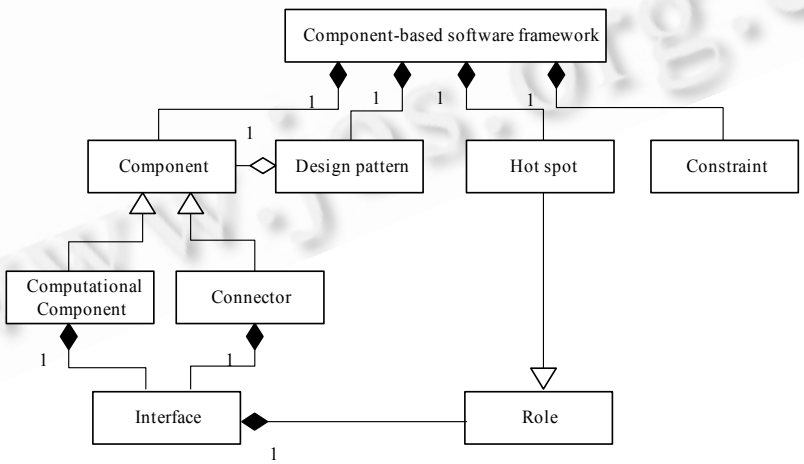

Fig.2 Meta-Model of component based software framework

图 2 基于构件的框架元模型

**** 杨芙清.软件产业面临新的发展机遇."99 年度推荐优秀软件产品活动推广应用总结大会"上的讲话.

2 CBSF 角色扩展形态

在采用基于构件的框架概念之后,需要解决的一个重要问题是设计并实现框架扩展点,以处理框架复用过程中的应用变化性.本文提出了扩展形态(extension forms)的概念,一种扩展形态是针对不同领域变化性类型和层次设计的构件(包括应用构件和框架构件)协作关系,它决定了扩展点的形式、扩展点和其他框架构件的协作关系、组装机制等方面的内容.下面以角色扩展形态为例,分别就 CBSF 扩展与变化性的关系、应用构件组装机制、CBSF 扩展设计等问题进行探讨.

2.1 基于构件的软件框架扩展和领域变化性

在基于框架进行应用开发时,开发者需要在框架的基础上,根据具体应用需求决定扩展部分,文献[6]中将扩展部分称为特定于应用的增量(application-specific increments,简称 ASIs),根据框架和 ASI 之间的调用方向,文献[9]中将框架区分为调用(calling)框架和被调用(called)框架,但是在实际应用中,两个方向的调用过程通常都存在,我们分别称为调用扩展(calling extension)和被调用扩展(called extension),前者等同于普通构件库的功能,而后者则提供了应用变化性的支持.

框架是针对特定领域的,所以它需要体现领域的共性,并支持领域内不同应用的变化性处理,即在框架复用中能够利用扩展点的定制机制,实现具体应用系统的不同特性.文献[10]描述了变化性的 5 个层次,即产品线层次、产品层次、构件层次、子构件层次和代码层次.因为 CBSF 体现了产品线中应用的共性(产品线层次),并且具有“黑盒”特征,即对使用者隐藏了具体实现(代码层次),所以 CBSF 设计中只需要关注其中的 3 个层次,即体系结构层次、构件层次和子构件层次.其中,处理体系结构层次的变化性需要考虑如何将构件匹配在一起、如何处理演化的接口等等;构件层次的变化性则需要考虑设计构件的接口;而子构件层次的变化性主要关注如何选择特定的构件实现以及构件内部功能的扩展和删除.分析上述的变化性分类可以发现,体系结构层次的变化性的实现主要考虑多个构件之间的连接关系;而构件层次和子构件层次的变化性的处理则取决于单个构件的接口定义及其具体实现,因此,在 CBSF 研究中,可以从结构和功能两个角度来探讨应用变化性,前者对应于构件之间的协作关系,而后者对应于单个构件.

从 CBSF 的角度考察结构变化性和功能变化性,可以看到框架定义了领域的共性特征,其中结构上的共性表现在框架构件之间的协作关系中,而功能共性体现于承担特定功能的计算构件中.在 CBSF 实例化过程中,框架通过调用扩展点组装应用构件,同一框架派生(实例化)的不同应用之间的功能差别体现在被组装的应用构件中,因此在基于 CBSF 开发应用时,功能变化性可以通过选择组装特定的应用构件来处理,而结构变化性则体现于调用扩展点的形态以及扩展点和其他框架构件的协作中.

2.2 CBSF角色扩展形态及组装机制

针对不同的领域变化性类型,CBSF 可以有多种扩展形态,如角色、插件、脚本等等.限于篇幅,本文主要针对角色(role)扩展形态进行描述.角色(role)扩展形态属于调用扩展,其结构如图 3 所示,其中角色是定义了接口以及调用规约而没有完全实现的构件.角色的概念提出较早,在信息安全领域基于角色的访问控制研究中,通过将用户和访问控制基于角色相关联,实现访问策略的精确定义[11];而在角色扩展形态定义的协作中,角色反映了一个应用构件在协作中所承担的责任,它起到了“占位符”的作用;在框架实现以及复用中,角色通常还实现以下 3 个方面的功能:1) 定位和查找应用构件;2) 检查应用构件接口是否符合框架定义的约束;3) 在框架构件对应用构件的调用中负责传送消息,并进行接口以及数据的转换.

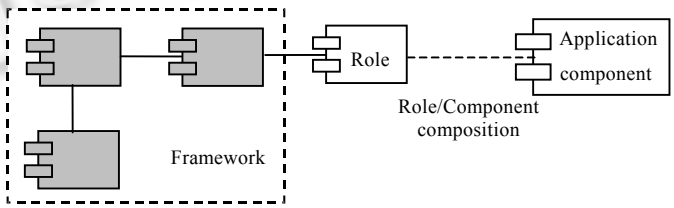

Fig.3 Calling extension form based on the role

图 3 基于角色的调用扩展形态

CBSF 角色构件可以分为两类,即具有部分实现特性的抽象构件[12]和模板构件,前者仅进行了接口定义,而

没有相应功能的具体实现;而模板构件包含实现算法,但是在使用时需要设置具体参数(包括数值参数和类型参数).角色构件需要进一步实例化,包括编写具体实现和指定运行时参数,这体现了框架的部分实现特性.

在角色扩展形态中,CBSF 利用角色构件定义了用户扩展部分与框架其他部分的协作关系,基于框架进行应用系统开发时,对于抽象构件,需要以某种方式实现与具体应用构件的组装,使得角色构件在框架协作中所承担的责任被应用构件所替代;而对于模板构件,则需要进行参数化,通常可以通过一个特定的接口实现该过程.角色/构件组装以及模板构件的参数化过程统称为角色的实例化过程,它支持框架的调用扩展机制.根据角色构件提供的具体功能,角色/构件组装可以采取以下 4 种方式:

1) 替代方式,其中角色构件仅定义应用接口,并在组装过程中被与之接口一致的具体应用构件所替代.替代方式是最为简单的组装途径,并可以通过构件部署实现.

2) 绑定方式,将角色构件和应用构件绑定形成复合构件,框架构件通过角色构件进行具体应用功能的调用,其中角色构件负责二者之间的消息传输.

3) 查找方式,角色构件定义了应用构件的接口,客户通过角色构件查找得到需要的应用构件.

4) 元构件方式,元构件是描述构件的构件,即构件使用者可以通过元构件接口获取构件的描述信息,包括接口定义、参数、约束等内容,该组装方式同时需要角色构件提供检查应用构件接口是否匹配的功能.

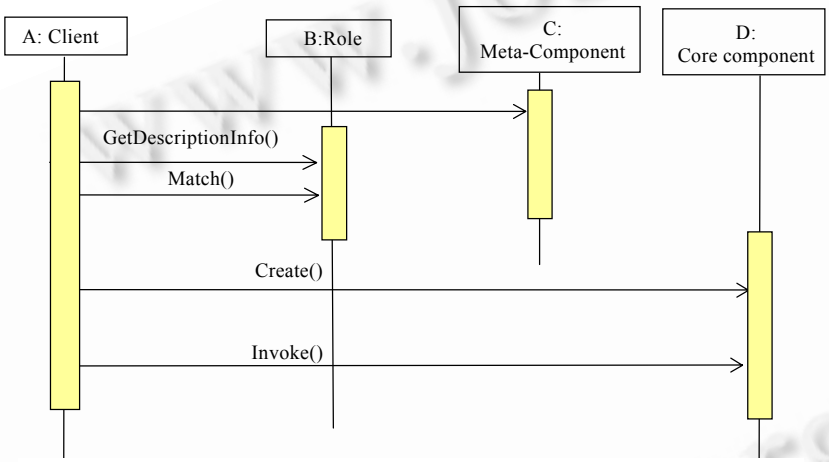

Fig.4 Calling process of role/component composition for meta-component

图 4 元构件角色/构件组装调用过程

在目前的一些主流构件模型中,如 DCOM,CORBA,EJB 等,都提供了对上述组装方式的支持,如 CORBA 的“桩”定义了与核心构件一致的接口,使得客户程序可以“位置透明”地访问核心构件;而 EJB 中的 JNDI 则支持基于命名进行构件的查找.

下面简单地以 DCOM 构件模型为例说明元构件组装机制.在 DCOM 中,可以通过 ITypeLib 和 ITypeInfo 接口得到构件的接口描述信息,即 ITypeInfo 接口起到了元构件的作用.在采用该方式进行构件组装时,角色被设计为元构件.它描述了可匹配应用构件的接口规约,通过它可以和待组装的应用构件进行匹配,并进行功能调用,从而完成组装.该组装过程的顺序如图 4 所示.

2.3 支持变化性处理的角色扩展形态设计途径

根据上面的讨论,CBSF 采用角色扩展形态时,角色构件和其他框架构件之间的连接关系规定了所支持的变化性类型,而角色扩展以及连接关系的设计,可以采用以下 3 个途径,即体系结构模式、设计模式和参数化构件.

1) 体系结构模式

一种体系结构模式定义了关于构件和连接件类型的术语以及一组约束它们组合方式的规定.比较典型的有管道-过滤器模式、客户-服务器模式、基于事件的隐式调用模式、层次系统、基于消息广播并面向图形用户界面的 Chiron2 模式等[13,14],每种模式都有其优点和缺点,适用于不同的应用设计.

在体系结构中,连接件规定了构件协作中计算构件所承担的责任以及参与约束不同的体系结构模式,决定了连接件的形式,如在客户端-服务器模式中连接件为远程过程调用,参与约束为 2,连接的构件所承担的角色分

别为调用者(caller)和被调用者(callee);而对于 C2 模式,连接件为消息总线,参与约束为 1…*n*,所承担的角色为消息发送/接收者.从 CBSF 扩展的角度来看,如果连接件规定的角色由角色构件承担,那么两者对接口的定义形式是一致的,因此体系结构模式以及连接件决定了角色构件的参与约束以及接口协议.在框架设计时,可以根据变化性的类型以及处理策略,选择合适的体系结构模式以定义 CBSF 的调用扩展.

2) 设计模式

在文献[15]中描述的 23 种面向对象设计模式可以分为类模式和对象模式.其中类模式处理类和子类之间的关系,这些关系通过继承而建立,是静态的,并且在编译时刻就已经确定下来;而对象模式则用于处理对象间的关系,这些关系在运行时刻是可以变化的,更具有动态性.由于构件难以实现继承关系,因此,只能使用类似于对象模式的模式.这些模式(除了 Singleton,Façade,Flyweight,Memento 模式以外)都提供了对于设计中某种变化性的解决方案,可以用于处理构件层次不同类型的变化性,实现 CBSF 扩展点.关于利用设计模式处理变化性,到目前为止已经有非常多的研究[16,17],在此不再赘述.

3) 参数化

参数化(parameterization)也是处理变化性的途径之一[18],它可以用于实现 CBSF 的模板构件,即在框架设计和实现时,保留一些待定的参数到实例化时通过实际的数值加以确定.参数化方式主要处理构件内部的、更小范围的变化性,包括运行参数、处理的数据类型等.与抽象构件的组装实例化方式不同,模板构件通过组装时的参数设置实现实例化,并且需要提供参数化接口.

从变化性层次的角度来比较这 3 种途径,其区别主要在于,体系结构模式途径主要用于处理全局性的,即产品层次的变化性;设计模式主要处理局部的构件层次的变化性;而参数化方式主要处理更加局部的子构件层次的变化性.

2.4 基于构件的软件框架应用实例

商业领域应用系统涉及不同业态的采购、销售、库存、物流管理,并且不同企业的业务流程也各不相同.在实践中完成的商业领域框架中,通过一系列构件,包括单据、账簿、核心商业过程(进货、销售、调货、仓储管理)等等,实现了商业领域的共性,并同时提供了 4 个方面的扩展,即基础数据的扩展、界面的扩展、系统功能的扩展以及业务过程的扩展,以支持用户基于该框架开发具体应用系统时的变化性处理.该框架与 San Francisco (SF)框架相比,具有以下优点[19]:1) SF 是基于继承的框架,开发者通过添加子类或者修改已有子类实现扩展,本文描述的商业领域框架是基于组装的;2) 要使用 SF,必须要编写实例化代码,并同框架代码一起编译;而商业 CBSF 框架则提供了扩展工具,直接支持构件的组装.

3 结 论

作为面向领域的大粒度可复用软件成分,框架是目前软件工程研究中的一个重要内容,从复用的角度来划分,框架可以分为白盒框架和黑盒框架,后者更易于使用,但是设计较为困难.本文描述的基于构件的框架属于黑盒框架,它避免了面向对象的白盒框架的一些不足.

开发和使用基于构件的软件框架的最大困难是如何定义其扩展点,并通过扩展点组装应用构件.通过参考面向对象框架的相关研究,并借鉴相关的模式,本文探讨了上述问题的解决方案,结果表明:1) 基于构件的框架包括两种扩展形式,即调用扩展和被调用扩展,其中被调用扩展类同于普通的构件库,提供了代码复用;而调用扩展支持框架复用过程中的变化性处理.2) 基于构件的框架主要通过构件接口实现其被调用扩展;而调用扩展可以有多种形态,角色构件是其中之一,它可以分为抽象构件和模板构件,在 CBSF 复用中,需要采用不同的方式对其进行实例化.3) 在进行框架设计时,可以根据不同的变化性类型,采用合适的体系结构模式、设计模式以及参数化方式,以设计框架的角色扩展形态.

本文只是对 CBSF 以及其扩展进行了初步的探讨,至于如何更加全面地描述各种框架扩展形态以及形式化地定义框架,并基于不同扩展形态实现框架的自动组装,还需要在进一步的工作中加以研究.

References:

[1] Buschmann F, Meumier R, Rohnert H, Sommerlad P, Stal M. Pattern-Oriented Software Architecture——A System of Patterns. Chichester: John Wiley & Sons Ltd., 1996.

[2] Cotter S, Potel M. Inside Taligent Technology. Addison-Wesley, 1995.

[3] Johnson RE, Foote B. Designing reusable classes. Journal of Object-Oriented Programming, 1988,1(2):22~35.

[4] Roberts D, Johnson R. Evolving frameworks: A pattern language for developing object-oriented frameworks. In: Martin DR, Buschmann F, eds. Pattern Languages of Program Design 3. Addison-Wesley, 1998. 471~486.

[5] Batory D, Smaragdakis Y. Object-Oriented frameworks and product-lines. In: Donohoe P, ed. Proceedings of the 1st Software Product Line Conference. Dordrecht: Kluwer, 2000. 227~247.

[6] Bosch J, Molin P, Mattsson M, Bengtsson P. Object-Oriented frameworks——problems and experiences. In: Fayad M, Schmidt D, Johnson RE, eds. Building application frameworks: object-oriented foundations of framework design. Chichester: John Wiley & Sons Ltd, 1999. 55~82.

[7] Dörner R, Grimm P, Abawi DF. Synergies between interactive training simulations and digital storytelling: A component-based framework. Computers & Graphics, 2002,(26):45~55.

[8] Lee S, Choi HS, Yang Y, Li S. XML based retrieval of object-oriented frameworks. IEEE International Conference on Systems, Man, and Cybernetics, 2000,4(10):2953~2958.

[9] Sparks S, Benner K, Faris C. Managing object-oriented framework reuse. IEEE Computer, 1996,29(9):52~61.

[10] Svahnberg M, Bosch J. Issues concerning variability in software product lines. In: Linden F, ed. Proceeding of the 3rd International Workshop on Software Architectures for Product Families. Berlin: Springer-Verlag, 2000. 146~157.

[11] Sandhu RS, Coyne EJ, Feinsein HL, Youman CE. Role-Based access control models. IEEE Computer, 1996,29(2):38~47.

[12] Zhang WJ. Research on software component model and corresponding techniques to support variability [Ph.D. Thesis]. Beijing: Peking University, 2001 (in Chinese with English abstract).

[13] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall, 1996.

[14] Taylor RN, Medvidovic N, Anderson KM, Whitehead Jr.EJ, Robbins JE. A component and message-based architectural style for GUI software. IEEE Software Engineering, 1996,22(6):390~406.

[15] Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[16] Keepence B, Mannion M. Using patterns to model variability in product families. IEEE Software, 1999,16(4):102~108.

[17] Chen ZL, Wang QX, Mei H, Yang FQ. Dealing with the variability in object-oriented domain design. Acta Electronica Sinica, 2001,29(11):1486~1490 (in Chinese with English abstract).

[18] Jacobson I, Griss M, Jonsson P. Software reuse: Architecture, process and organization for business success. Addison Wesley, 1997.

[19] Hu WH. Study and implementation of business domain software framework [MS. Thesis]. Beijing: Peking University, 2001 (in Chinese with English abstract).

附中文参考文献:

[12] 张文娟.支持变化性的软件构件模型及其相关技术研究[博士学位论文].北京:北京大学,2001.

[17] 陈兆良,王千祥,梅宏,杨芙清.面向对象领域设计中的变化性处理.电子学报,2001,29(11):1486~1490.

[19] 胡文蕙.商业领域软件框架的研究与实现[硕士学位论文].北京:北京大学,2001.