ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.1, January 2010, pp.68−83
doi: 10.3724/SP.J.1001.2010.03536

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于组件模型分析的组件容器产品线体系结构*

刘国梁[1,2,3+], 魏 峻[1], 冯玉琳[1,2]

[1](中国科学院 软件研究所 软件工程技术中心,北京 100190)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100049)

Container Product Line Architecture Based on Component Model Analysis

LIU Guo-Liang[1,2,3+], WEI Jun[1], FENG Yu-Lin[1,2]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: glliu@otcaix.iscas.ac.cn

Liu GL, Wei J, Feng YL. Container product line architecture based on component model analysis. *Journal of Software*, 2010,21(1):68−83. http://www.jos.org.cn/1000-9825/3536.htm

Abstract: Component containers play a key role as the infrastructure of component-based distributed applications at deployment and running time. In recent years, various kinds of component models are emerging and evolving, this brings great challenges to the development component container. Product line engineering is one of the most promising techniques to improve the quality and productivity of software. Study on product line architecture (PLA) for component containers is the most important, and of great help to improve the reusability of architectural design. Since component models are cornerstone of container design, an analyzing framework of component models is proposed integrated with domain analysis. This paper builds the domain model of component container by establishing mapping between component model elements and domain requirement elements. Based on the design principles of component container and variability encapsulation rules, this paper proposes a component container PLA, named PLACE, which meets domain requirements of component container by introducing optionality, hierarchical module structuring and decision model. PLACE is also applied to the development of several component containers on ONCE platform, which proved the effectiveness of this approach.

Key words: component model; component container; product line architecture; product line engineering

摘 要: 组件容器为组件提供部署和运行环境,是基于组件分布式应用开发的核心.近年来分布式组件的多样化和快速演化对组件容器的开发方法提出了挑战.产品线工程是基于公共的核心资产开发特定领域内软件产品系列的软件工程方法,产品线体系结构是其中最重要的部分.进行组件容器产品线体系结构的研究能够提高组件容器的

* Supported by the National Natural Science Foundation of China under Grant Nos.60673112, 90718033 (国家自然科学基金); the National Basic Research Program of China under Grant No.2009CB320704 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01Z19B, 2007AA010301 (国家高技术研究发展计划(863))

Received 2008-07-04; Accepted 2008-11-21

结构复用性,获得更高的生产效率和质量.由于组件模型是组件容器设计的基础,在领域分析阶段引入组件模型分析,提出了组件模型分析框架,通过组件模型元素到领域需求元素的映射,建立组件容器领域模型.提出了组件容器设计的基本原则,并根据变化性封装原则,提出了组件容器产品线体系结构 PLACE,通过引入可选属性、模块层次结构和决策模型,实现组件容器的领域需求.PLACE 产品线体系结构已在网驰平台的多个组件容器设计中得到应用.

关键词: 组件模型;组件容器;产品线体系结构;产品线工程

中图法分类号: TP311 **文献标识码:** A

基于组件的开发方法(component-based development,简称CBD)要求通过组合已有的组件构造新的软件系统,以降低开发成本,促进软件复用[1].组件是组合的单元,使用接口描述其提供的服务契约,并明确定义对环境的要求,可以独立部署并由第三方进行组合[2].

组件容器为组件提供部署和运行环境,是基于组件分布式应用开发的核心.组件容器负责在运行时创建和管理组件实例,自动为组件分配系统资源,维护组件间交互,透明地拦截客户请求.组件容器中封装的底层功能以系统服务的形式提供给组件,包括事务、安全、日志等.

近年来,随着 Internet 规模的增长,分布式组件技术快速发展,新的组件模型不断涌现,现有的组件模型也持续更新.组件模型的多样化和快速演化对组件容器的开发方法提出了挑战.组件容器体系结构成为分布式组件运行支撑系统设计的热点和难点问题.

产品线工程是基于公共的核心资产开发特定领域内软件产品系列的软件工程方法[3],近年来广泛应用于大规模软件系统的开发.产品线工程包含领域分析、领域设计和领域实现 3 个阶段[4].领域分析阶段的成果是领域模型,在一般需求模型的基础上,增加了对领域共性和变化性的描述,从而为领域内新系统的开发提供可复用的软件需求规约,并指导领域设计和领域实现阶段的可复用软件资产的生产.产品线体系结构(product line architecture,简称PLA)[5]是领域设计阶段的成果,是产品线工程最重要的核心资产.PLA通过对领域需求的支持和共性需求的实现,指导领域内的系统化复用.组件容器PLA研究能够提高组件容器的结构复用性,获得更高的生产效率和质量,是改进组件容器开发过程的合适途径.

组件容器的领域分析必须建立在对组件模型深入分析的基础上.组件模型是组件和组件应用设计开发遵循的标准和规范,符合组件模型规范是组件与其他软件单元的主要区别之一[6],组件容器为符合特定组件模型规范的组件提供部署和运行环境[7].因此,组件模型是组件容器需求的主要来源.但组件模型是对组件的约束,与组件容器需求在构成内容和描述方式上存在较大差异.因此,需要解决组件模型到领域模型的转换问题.为此,我们提出了组件模型分析框架,将组件模型细分为模型元素,结合基于原子需求的领域需求描述方法,给出组件模型到领域模型的映射关系,并在此基础上建立组件容器领域模型.

组件容器 PLA 设计需要符合可复用、可追溯和可扩展 3 个原则,以提高复用频率、缩短设计周期、延长使用寿命,达到提高组件容器结构复用性的目的.本文在领域模型的基础上,按照变化性封装原则,提出了组件容器产品线体系结构 PLACE,介绍了其模块结构和变化性管理机制.PLACE 已在实际组件容器设计中得到应用.

本文第 1 节讲述如何将组件模型分析和领域建模方法相结合,建立组件容器领域模型.第 2 节给出组件容器 PLA 的设计原则和 PLACE 的体系结构与变化性管理机制.第 3 节基于实际组件容器设计案例,展示基于 PLACE 的组件容器的设计过程.第 4 节是对相关工作的简要介绍.第 5 节对全文进行总结.

1 组件容器领域分析

1.1 组件模型分析框架

组件模型是组件和组件应用设计开发遵循的标准和规范[6],定义了什么是组件,如何构造组件,如何组合或组装、如何部署组件等内容[1].我们提出了基于组件模型元素的组件模型分析框架,将组件模型划分为语法模

型、交互模型和部署模型.

语法模型定义组件的开发语言和语法约束,给出了“如何构造组件”.交互模型定义组件在容器管理下提供服务和相互协作的方式,规定了“如何使用组件”和“如何组合”.部署模型定义容器提供的资源和绑定到组件的方式,说明了“如何部署”.这 3 个模型又划分为若干子模型和模型元素.见表 1.

Table 1 Analyzing framework of component models

表 1 组件模型分析框架

	Submodel	Model elements
Syntax model	Interface syntax	Interface definition language, interface constraint
	Implementation syntax	Implementation language, implementation constraint
	Metainfo syntax	Metainfo description language, metainfo constraint
Interaction model	Client interaction	Naming protocol, client protocol, reentrant, QoS support
	Component interaction	Interaction constraint, communication mechanism
	Container interaction	Persistence, lifecycle event, thread management, resource management
Deployment model	Packaging	Packaging specification
	Deployment description	Deployment description language, deployment description constraint, resource type

下面详细说明组件模型分析框架的各组成部分:

语法模型(syntax model)包括组件各构成部分的定义语言和语法约束.组件由接口、实现和元信息描述构成,其中接口可能有一个或多个,又可分为提供接口和需求接口,分别定义组件提供的服务和需要运行环境提供的服务.按照定义对象,语法模型分为 3 个子模型:接口语法模型(interface syntax)、实现语法模型(implementation syntax)和元信息语法模型(metainfo syntax).

在接口语法模型中,接口定义语言(interface definition language)一般有编程语言(Java)或可映射到编程语言的 IDL(WSDL,OMG IDL,Microsoft IDL 等).接口语法约束描述组件定义接口的语言单元,与接口定义对应.

在实现语法模型中,部分组件模型限定了实现语言(implementation language),有些则是语言独立的.但无论哪一种,其实现语言均采用常用的编程语言,即Java,C,C++或.NET[8].按照语言的不同,实现对应的语言单元可能是Java类、C模块、C++类等.

组件元信息描述作为对接口描述的补充,给出接口定义语言无法描述的行为、同步和QoS属性[9].但目前大部分的组件模型均只支持语法描述[8,9].因此,在元信息语法模型中,元信息描述(metainfo description)中只包括了接口与实现的对应关系、组件定位标识等补充信息,并经常与部署资源描述在同一个文件中加以定义.

交互模型(interaction model)包括组件与客户端之间、组件与组件之间以及组件与容器之间的交互方式,其中前两种交互均由容器管理并提供实现机制.交互模型包含客户端交互模型(client interaction)、组件交互模型(component interaction)和容器交互模型(container interaction)这 3 个子模型.

客户端交互模型包含了组件的查找协议(naming protocol)、通信协议(client protocol)、重入控制(reentrant)和提供的 QoS 保障(QoS support).不同的通信协议隐含了不同的属性,如 HTTP 协议是基于文本和同步请求/响应范型的无状态协议.常见的 QoS 保障有事务、安全等.

组件交互模型包括交互约束(interaction constraint)和通信方式(communication mechanism).交互约束限制了可访问组件的类型和可见范围,通信方式分为方法调用、消息通知和共享数据空间等.组件也可作为客户端访问其他组件,这时不受组件间交互约束的限制.

容器交互模型中包括组件的状态持久化(persistence)、生命周期事件通知(lifecycle event)、线程管理(thread management)和资源管理(resource management)等.组件与容器的关系类似于进程与操作系统的关系,容器管理组件运行所需要的各种资源.

部署模型(deployment model)包括部署单元的结构规范和描述方式.部署模型包含子模型:组装(packaging)模型和部署描述(deployment description)模型,分别描述了部署文件结构和部署描述信息的定义方式.

组装模型即打包规范(packaging specification),描述组件部署时的文件格式,常见的如 JAR,DLL,ZIP 等格式或使用文件系统的目录格式.

部署描述模型包括部署信息描述语言(deployment description language)、描述约束(description constraint)和资源类型(resource type),目前独立的部署信息描述文件大多采用 XML 语言.描述约束是指部署信息描述中的内容组织方式和数据类型等,组件模型的描述约束均各不相同.部署描述给出了组件需要的资源类型,部署者将组件绑定到容器提供的资源实例中,如数据源、环境变量、对其他组件的引用等.资源类型定义了组件容器中提供给组件的资源的种类.

1.2 组件容器的共性需求分析

领域需求的描述有多种方式,如UML用例[10]、特征模型[11,12]等.本文采用DREAM[13]使用的领域需求描述方法.将领域需求分为功能需求和非功能需求.功能需求由原子需求(primitive requirement,简称R)构成,定义了系统所应具有的功能.PR定义为构成系统外部可见的单个动作的一组操作.在外部参与者的视角中,PR是需求不可再分的最小单元,具有与事务类似的原子性.在低一级的PR元素(PRelement)层次,需求变化性通过变化点(variation point,简称VP)来描述.PR元素是PR中可以产生变化点的各个方面,变化点通过PR元素与PR相关联.非功能需求定义为对功能需求的约束,由于组件模型内只包含组件功能的规范,故本文不考虑非功能需求.

组件容器为符合模型的组件提供部署和运行环境.因此,领域共性需求包括组件部署期需求和组件运行期需求.在组件部署期,组件或组件应用部署在容器中,成为组件实例.在组件运行期,容器提供组件访问和组件间交互机制,管理组件实例的生命周期并提供组件运行所需资源.

用例与 PR 的关系非常密切.通过组件容器的需求,首先构造组件容器的领域用例.

在UML规范[14]中,用例定义为系统完成的一组操作,产生用户或参与者要求的结果.与组件容器交互的参与者共有 5 个:部署者(deployer)、组件客户端(component client)、容器管理员(administrator)、组件实例(component instance)和外部资源(resources).分析参与者与组件容器的交互,得到领域用例模型,如图 1 所示.其中 4 个参与者是用例发起者,分别是部署者(Deploy部署用例与Undeploy反部署用例)、组件客户端(Component Access组件访问用例)、组件实例(AccessContext访问上下文用例与ComponentInteract组件交互用例)和容器管理员(Manage管理用例).用例共包括上述 6 个用例和 2 个抽象用例(LifecycleManage生命期管理用例,Resource Access资源访问用例).

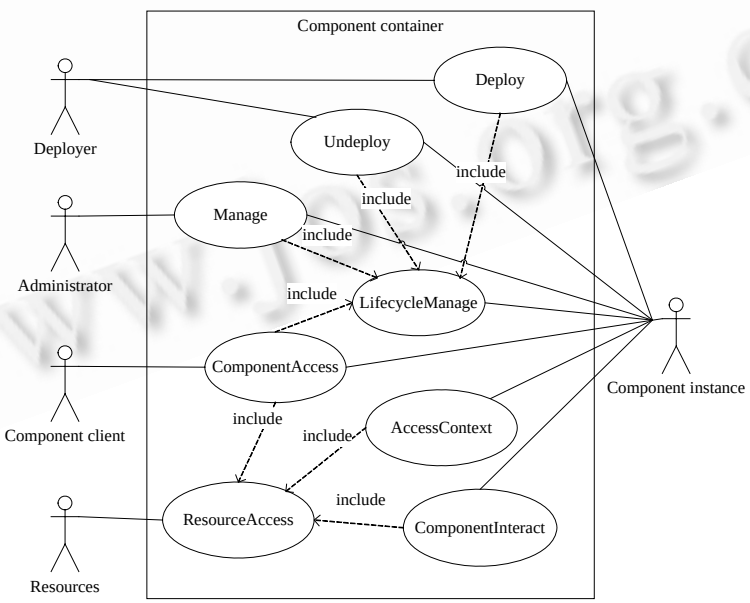

Fig.1 Domain usecase model for component container model

图 1 组件容器领域用例模型

将用例按照外部参与者的视角划分为原子步骤,得到对应的 PR.例如组件访问(ComponentAccess)用例表示客户端通过容器访问组件,共有两个参与者:组件客户端和组件实例.从外部参与者的视角,组件访问分为两个原子步骤,即组件客户端发送请求到组件实例以及组件实例返回结果到组件客户端,容器内部的中间步骤(请求解析、查找组件等)对外部来说不可见.故该用例可以分为两个 PR:处理远程请求(PR3)和返回远程应答(PR4).在组件访问过程中可能会访问资源和上下文.由于这些操作已由单独的用例表示,在分析组件访问用例时不包括这些步骤.

将图 1 中的用例划分为 PR,得到组件容器领域的 PR 列表见表 2.

Table 2 Common PR list of component container domain

表 2 组件容器领域公共 PR 列表

PR index	Description	Corresponding usecase
PR1	Deploy	Deploy
PR2	Undeploy	Undeploy
PR3	Handle remote request	ComponentAccess
PR4	Return response to remote client	ComponentAccess
PR5	Inter-Component request	ComponentInteract
PR6	Inter-Component response	ComponentInteract
PR7	Request context	AccessContext
PR8	Lifecycle management	LifecycleManage
PR9	Resource access	ResourceAccess
PR10	Container management	Manage

1.3 基于组件模型分析的容器领域建模

1.3.1 需求细化与共性需求建模

PR 可进一步细分为 PR 元素.PR 元素描述 PR 的静态结构和动态行为的各个方面,是描述领域需求变化性的基本单元.PR 元素分为两类:静态 PR 元素和行为 PR 元素,分别定义结构和行为的某个方面.PR 代表了组件容器的一段处理流程,组件容器对 PR 的支持受到若干组件模型元素的约束.根据与 PR 有约束关系的组件模型元素,可以分析 PR 流程的变化性,并得到 PR 对应的 PR 元素列表.

在组件模型分析框架中,交互模型描述各实体的交互行为规范,而语法模型和部署模型描述各实体的语法结构规范.因此,大部分交互模型的模型元素映射到行为 PR 元素,而与数据结构相关的规范,包括语法模型、部署模型和交互模型的一部分(如通信协议的消息格式,生命期事件通知的类型等),则映射到静态 PR 元素.

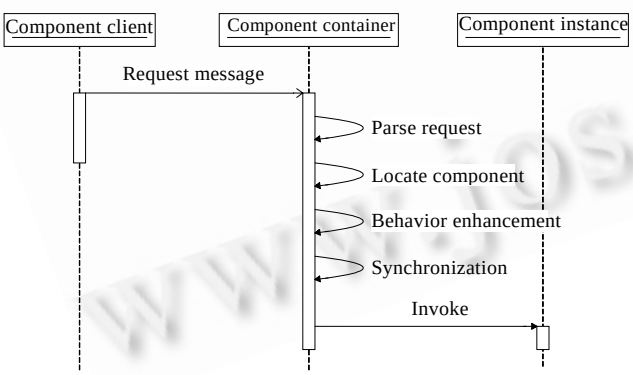

Fig.2 Sequence diagram of PR3: RemoteRequestProcess

图 2 PR3 远程请求处理流程图

以 PR3“远程请求处理”为例,其流程图如图 2 所示.PR3 是客户端与组件交互流程中的一部分,受到客户端交互子模型的约束.该子模型内的 4 个模型元素:查找协议、通信协议、QoS 保障和重入控制,分别映射到 PR3 的 4 个行为 PR 元素.与 PR3 相关的数据结构规范是输入的客户端通信协议消息和输出的组件调用参数,分别与模型元素通信协议和接口约束(interface constraint)相关,映射到 PR3 的两个静态 PR 元素.从模型元素到 PR3 的 PR 元素的映射如图 3 所示.

按照上述映射方法,我们从与 PR 有约束关系的组件模型元素派生出 PR 元素列表.PR 与其 PR 元素构成了组件容器领域共性需求的模型,如图 4 所示,其中展开了 PR1 和 PR3 的 PR 元素列表.

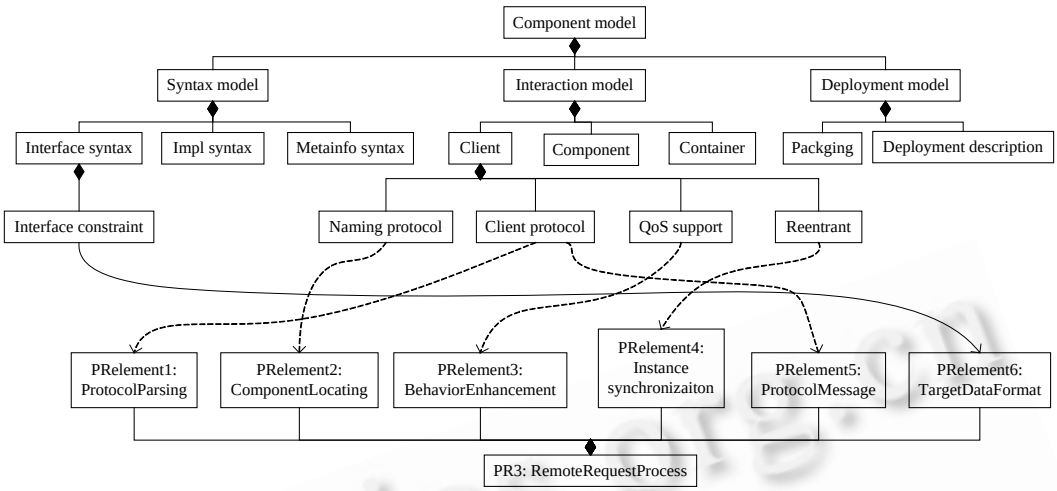

Fig.3 Mapping component model elements to PRelements

图 3 从组件模型元素到 PR 元素的映射

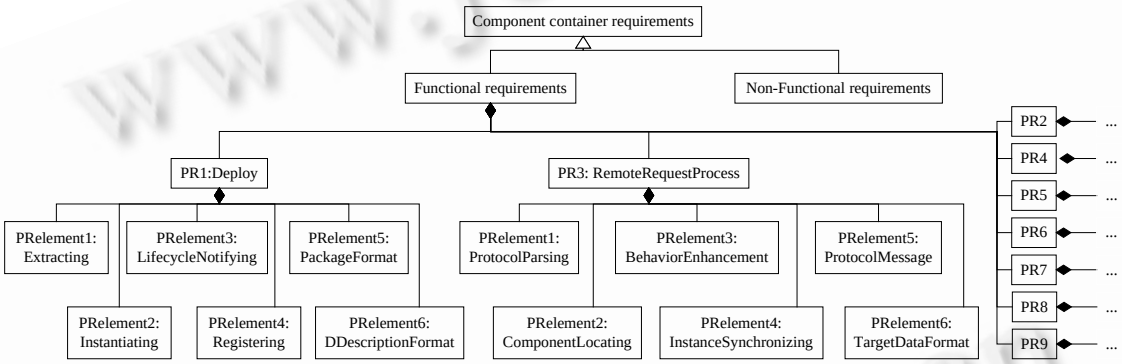

Fig.4 Domain model of component container

图 4 组件容器领域模型

1.3.2 领域成员的选择与变化性需求建模

领域需求的变化性用变化点来描述,PR 通过 PR 元素与若干变化点相关联.

变化点由其 3 个属性加以描述:类型、数量和可选项.可选项是变化点可能的实现方式.数量则表示变化点中可选项的数量范围.变化点的类型分为控制、计算、外部计算和数据.变化点的类型与 PR 元素类型相关:控制类型(模块间交互)和计算类型(模块功能)的变化点关联到行为 PR 元素.数据类型的变化点关联到静态 PR 元素.

产品线中包含的领域成员决定了变化性需求.因此在变化性建模之前,先要确定产品线中包含哪些组件容器.分布式组件容器种类众多,无法全部纳入领域建模的范畴.因此,选择有代表性的组件容器进行变化性建模.

明确领域成员后,按照组件模型分析框架对每个成员对应的组件模型进行分析,将模型元素的所有取值合并为取值集合.变化点的属性可以从关联的模型元素的取值集合中获得.数量属性由单个模型中包含取值的最大和最小数量来决定,可选项属性则包含所有可能的取值.

例如,我们选择 EJB 和 BPEL 组件进行变化性建模,其中 EJB 包括会话 EJB、消息驱动 EJB 和实体 EJB 等.通过对两个组件模型的分析,其模型元素 Client Protocol 对应的取值分别为 RMI-IIOP,CORBA-IIOP,Messaging(基于消息)协议和 SOAP/HTTP 协议.故 Client Protocol 的取值集合为{RMI-IIOP|CORBA-IIOP|Messaging,SOAP/HTTP}.Client Protocol 对应到两个 PR 元素(如图 3 所示):ProtocolPasing 和 ProtocolMessage,

分别处理协议的同/异步行为和消息格式.由模型元素和取值集合可得 ProtocolParsing 的变化点为控制类型,数量为[1..*n*],其可选项为(synchronous,asynchronous);而 ProtocolMessage 的变化点为数据类型,数量为[1..*n*](支持一种以上消息格式),可选项为(RMI-IIOP,CORBA-IIOP,Messaging,SOAP/HTTP).

1.3.3 PLACE 领域模型

网驰平台(ONCE)[15]是中国科学院软件研究所研发的一个软件基础架构平台,由基础运行支撑平台、数据集成平台、服务集成平台、流程集成平台及信息门户这 5 部分组成,其中涉及的组件模型有Servlet[16],EJB[17],Web服务,Portlet[18],BPEL[19]等.我们以这 5 种组件模型为基础建立组件容器产品线PLACE(product line architecture for component container environment).

按照第 1.1 节中给出的组件模型分析框架,我们对这 5 种组件模型进行了分析、比较,并选取了与 PR3 相关的部分列在表 3.

然后由第 1.3.2 节提到的变化性需求建模方法,基于表 3 得到与各 PR 元素关联的变化点,如图 5 所示,图中只展开了 PR3 部分.变化点的数量与可选项属性来自表 3 中对应模型元素的取值.PR3 的文本表述如图 6 所示.

Table 3 Comparison of component models

表 3 组件模型比较

Model elements	EJB	Servlet	Portlet	Web service	BPEL
Client protocol	RMI/CORBA-IIOP Messaging	HTTP HTTPS	HTTP HTTPS	SOAP/HTTP	SOAP/HTTP
	Synchronous asynchronous	Synchronous	Synchronous	Synchronous asynchronous	Synchronous asynchronous
Naming protocol	JNDI name	URL name	URL name	WS-Addressing	BPEL (partnerLink/ portType/method)
QoS support	Transaction session security monitoring	Security session logging monitoring	Security windows state logging	WS-*	Monitoring transaction logging
Reentrant	Reentrant queue-required	Reentrant	Reentrant	Queue-Required	Queue-Required
Interface constraint	Java method: Business interface	Java method: Servlet	Java method: Portlet	WSDL port	WSDL port
Others	…				

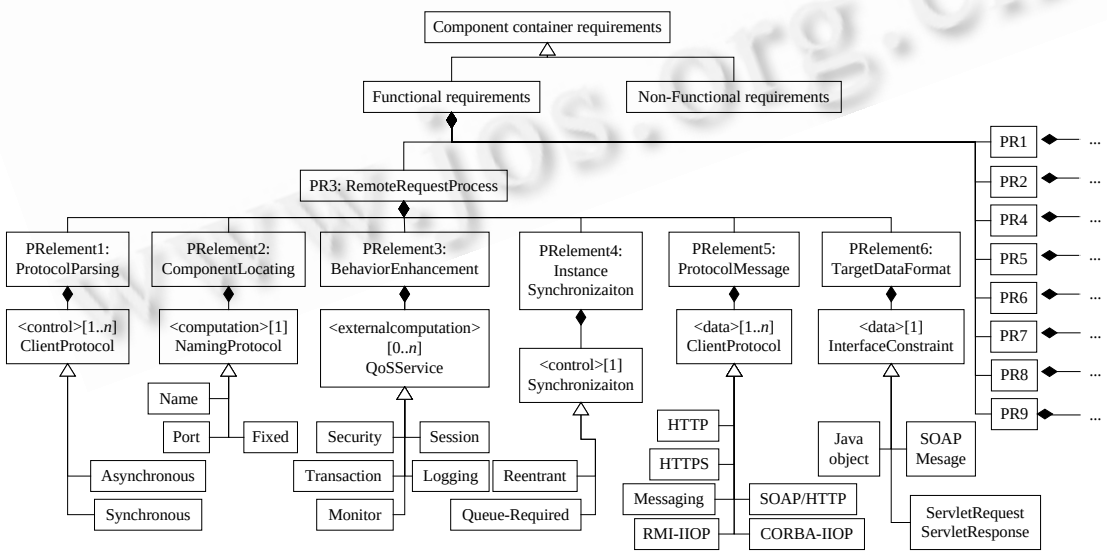

Fig.5 Domain model of PLACE

图 5 PLACE 领域模型

PR3: RemoteRequestProcess
Behavior PRelement:

- PRelement 1: Parse remote message
  - Variation point type: Control
  - Variation point cardinality: [1..*n*]
  - Variants: Syn, Asyn
- PRelement 2: Locate component
  - Variation point type: Computation
  - Variation point cardinality: [1]
  - Variants: Naming protocol
- PRelement 3: Behavior enhancement
  - Variation point type: External computation
  - Variation point cardinality: [0..*n*]
  - Variants: Transaction, session, security, logging, monitor
- PRelement 4: Instance synchronization
  - Variation point type: Control
  - Variation point cardinality: [1]
  - Variants: Reentrant, queue-required

Static PRelement:

- PRelement 5: Protocol message
  - Variation point type: Data
  - Variation point cardinality: [1..*n*]
  - Variants: Network protocols
- PRelement 6: Target data format
  - Variation point type: Data
  - Variation point cardinality: [1]
  - Variants: Component parameter type

Fig.6 A example of PR specification
图 6 PR 规范举例

2 组件容器产品线体系结构 PLACE

2.1 组件容器PLA设计原则

组件容器 PLA 研究的主要目的是提高组件容器的结构复用性,我们从提高复用频率、缩短设计周期、延长使用寿命等角度出发,提出了组件容器 PLA 设计应符合可复用、可追溯和可扩展这 3 个设计原则.

可复用是组件容器 PLA 设计的首要原则,目的是促进领域内不同类型组件容器间的复用.为此,PLA 的体系结构设计以领域模型为指导.其顶层模块以共性需求为基础进行设计,而将变化性需求封装在模块内部来实现.由于变化性体现的是领域成员的差异,通过变化性封装,可以提高 PLA 结构复用性和稳定性.

可追溯是通过建立需求到体系结构设计的链接,将需求变化性与设计变化性间的约束关系以决策模型的形式文档化,缩小需求与实现间的距离.支持可追溯性可提高设计效率,是需求和设计的一致性检查的基础,也是实现设计过程流程化和自动化的前提.

可扩展是指通过对共性需求的把握和对未来需求的预期,延长 PLA 使用寿命,从而降低产品平均投入,节约开发成本.我们提高可扩展性的手段是在领域分析阶段引入组件模型分析,从而保证了领域模型反映出来的大部分组件容器的共性需求.在组件模型分析中综合已有组件模型分析方法,为预期的需求也提供了一定的支持.

2.2 PLACE的体系结构

根据可复用性原则,PLACE 的顶层模块以领域共性需求为基础.PR 描述共性需求的处理流程,故 PLACE 的模块划分以 PR 为指导.

模块划分的指导原则是变化性封装原则.首先将 PR 的行为 PR 元素分配给不同的模块实现.以 PR1“部署”为例,其序列图如图 7 所示,对应到 4 个行为 PR 元素,如图 4 所示.在 PLACE 体系结构设计中分别由 4 个不同的模块分别实现,分别是部署器(deployer)、实例管理器(instance manager)、组件管理器(component manager)和组件注册(registry).模块间的交互按照 PR1 等共性需求设计,而将部署元信息、组件实现、生命期事件和查找协议等变化性封装在各模块内部.

第 2 步是将与同一种数据相关的功能模块合并,以

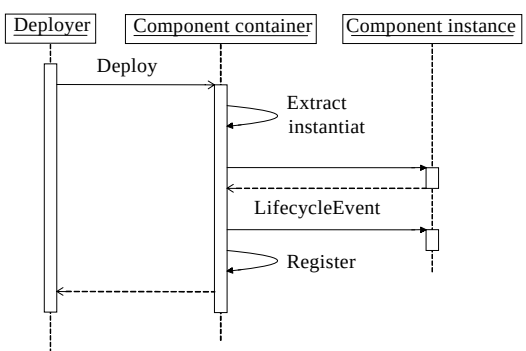

Fig.7 Sequence diagram of PR1: Deploy
图 7 PR1 部署的序列图

封装数据类型的变化性.如在 PR1 部署、PR2 反部署与 PR3 远程请求处理中,组件注册、组件注销和组件查找功能均与组件查找协议和组件接口定义相关,故合并到 Registry 组件注册模块,将注册、注销和查找功能包含在其模块提供接口中.

我们遵照以上模块划分和归并的分析方法,以图 5 所示的PLACE领域模型为基础,给出PLACE的总体结构如图 8 所示.图中所用图例是在Koala[20]和文献[21]中所用图例的基础上发展而来的,其中PLACE包含的体系结构元素有 3 种:模块(方框)、接口(模块边缘上的三角形,方向的内外分别表示需求接口和提供接口)和调用关系(接口间的连线).我们为体系结构元素引入了可选属性,分为必选元素和可选元素两类.其中必选元素存在于所有由PLACE派生的产品体系结构中,可选元素则只存在于部分派生的产品体系结构中.图中虚线的模块、接口和调用关系表示可选元素,实线的则表示必选元素.

下面对 PLACE 中各顶层模块的功能和相互关系加以简要介绍.

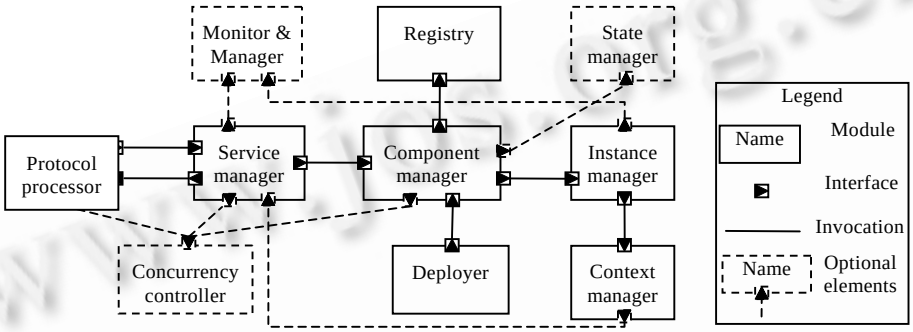

Fig.8 Static structure of PLACE

图 8 PLACE 的静态结构

协议处理器(protocol processor)负责与客户端通信用的网络协议的解析与构造.它把请求消息解析为内存对象,传递给服务管理器(service manager);也负责用服务管理器的返回对象构造应答消息,发送给客户端.协议解析有时需要并发完成,这时协议处理器会调用并发控制器(concurrency controller)进行并发处理.

服务管理器负责容器服务的载入、启动和停止等管理,还负责请求到容器服务的分发.请求由协议处理器解析为对象后,首先由服务管理器中的服务进行处理,然后传递给组件管理器(component manager).服务管理器有时也需要调用并发控制器对容器服务提供并发支持.服务管理器还需要为监控管理器(monitor and manager)提供运行状态信息.

组件管理器集中了与组件元信息和生命周期相关的组件运行时管理、组件配置、生命期管理、生命期事件通知和复合组件管理等功能.在组件部署时,部署器调用组件管理器完成部署操作.在处理请求时,组件管理器分别调用组件注册(registry)、状态管理器(state manager)、实例管理器(instance manager)进行组件查找、组件状态的载入和持久化操作、组件实例调度等.组件管理器还可能调用并发控制器为各子模块提供并发支持.

上下文管理器(context manager)除了向实例管理器中的组件实例提供环境信息以外,还接收和处理组件实例对运行服务的调用,并调用服务管理器完成处理动作.

监控管理器负责收集容器的运行时信息及动态调整容器配置,根据目前实际应用中监控管理器相关的信息种类,其相关模块包括服务管理器和实例管理器.

2.2.1 模块的层次结构

PLACE 的模块按照层次结构组织,即模块可以划分为子模块,其子模块的接口绑定到父模块的接口,包括需求接口和提供接口.子模块同样具有可选属性,其可选属性相对于父模块而定义.例如,必选模块 ModuleB 的父模块是可选模块 ModuleA,这表示 ModuleB 与 ModuleA 同时被选择,或同时不被选择.层次结构可以嵌套.

下面以服务管理器模块为例加以说明,如图 9 所示.

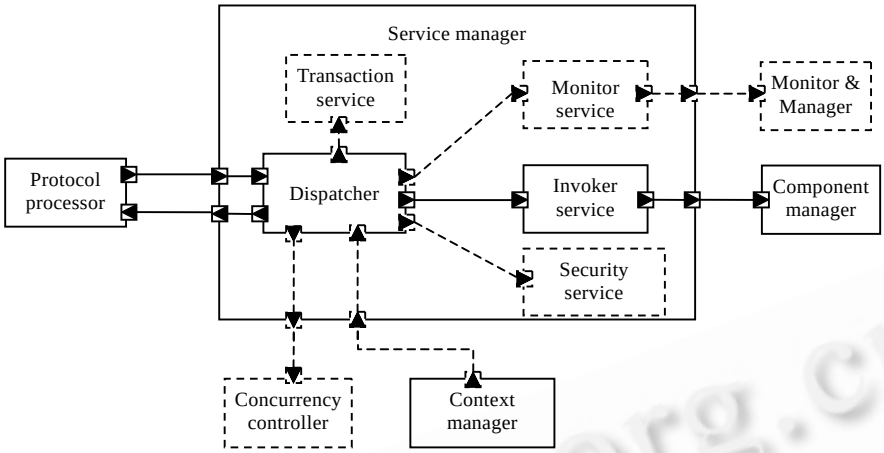

Fig.9 Hierarchy structure of service manager module in PLACE

图 9 PLACE 中服务管理器模块的层次结构

服务管理器模块包括两个必选子模块和若干可选子模块.必选子模块包括分发器(dispatcher)和组件调用服务(invoker service).其中分发器决定调用的服务及其顺序,组件调用服务负责调用组件管理器进行后续操作.为简单起见,图中并未显示服务管理器所有的可选子模块,只包含了事务服务(transaction service)、监控服务(monitor service)和安全服务(security service)模块,其他可选的服务还有会话服务(session service)、日志服务(logging service)等.

服务管理器的两个可选接口分别是调用并发控制器的需求接口和被上下文管理器调用的提供接口.这两个接口均绑定到分发器子模块,分别用于多个容器服务的并发运行支持和通过上下文向组件提供事务、安全、会话等上下文信息.

2.2.2 模块一览表

以上给出了 PLACE 中模块的图形化表示,在实际使用中我们使用模块一览表来记录 PLACE 中的模块及其可选属性和相互关系.表 4 是模块一览表的节选,包括了服务管理器模块和相关的部分模块.

Table 4 Excerpt from module list of PLACE

表 4 PLACE 模块一览表节选

Module index	Module name	Optionality	Invoking module	Invoker	Parent module
M1	ProtocolProcessor	M (mandatory)	M2, M8	M2	
M2	ServiceManager	M	M1, M3, M8, M10	M1, M6	
M2.1	Dispatcher	M	M1, M8, M2.2, M2.3, M2.4, M2.5	M1, M6	M2
M2.2	InvokerService	M	M3	M2.1	M2
M2.3	TransactionService	O (optional)		M2.1	M2
M2.4	MonitorService	O	M10	M2.1	M2
M2.5	SecurityService	O		M2.1	M2
M3	ComponentManager	M	M4, M5, M8, M9	M2, M7	
M4	Registry	M		M3	
M5	InstanceManager	M	M6, M10	M3	
M6	ContextManager	M	M2	M5	
M7	Deployer	M	M3		
M8	ConcurrencyController	O		M1, M2, M3	
M9	StateManager	O		M3	
M10	Monitor&Manager	O		M2, M5	

2.3 PLACE的变化性管理

PLA除了包含一般体系结构的元素外,还包含从PLA映射到产品系列中每个产品的方式,即体系结构的变化性[5,22].PLA主要考虑产品结构设计时的变化性绑定,对应链接时、运行时等其他绑定时间的绑定机制[23,24]不是PLA设计的重点.

PLACE 对体系结构变化性的支持体现在以下 3 个方面:

1) 为体系结构元素赋予可选属性,其目的是使 PLACE 可以按照特定的规则派生出不同的产品体系结构.
2) 根据可追溯性原则,建立需求到设计的决策模型.在决策模型中记录了需求变化性和体系结构变化性的对应关系和约束条件,从而可以根据产品需求确定相应的模块设计.
3) 根据需求变化性的属性,在模块中选择不同的体系结构变化性实现机制.

可选属性在第 2.3.2 节已经做过介绍,下面我们分别介绍其他两方面的内容.

2.3.1 决策模型

决策模型以变化点为单位,描述了变化点与模块的对应关系以及变化点间的约束条件.

变化点与 PLACE 模块间的对应关系有两种:封装关系和映射关系.封装关系将变化点映射为模块,表示该模块将变化点封装为模块的内部实现,保持模块对外接口的稳定.每个变化点有且只有一个封装模块.映射关系将变化点的可选项映射到可选模块,表示变化点绑定到可选项时对应的实现模块.约束条件描述变化点之间的逻辑约束关系.

表 5 是决策模型的例子,其中的变化点选自 PR3 的描述,如图 6 所示.VP1 是通信协议的同步属性支持,由图 6 可知,其数量属性是[1..*n*],即同步消息解析、异步消息解析或二者同时支持.

VP1 的封装模块是 M1 协议处理器模块,表示 VP1 在该模块内部实现,需求变化不会对其外部接口(与服务管理器和并发控制器连接)产生影响.映射模块一栏给出了面对 VP1 不同的绑定时的模块列表.其中 M1.2 和 M1.3 分别是同步和异步消息监听器,在二者同时支持时,则需要 M1.4(同步行为适配器)模块.

约束条件一般采用逻辑表达式描述,也可以采用自然语言,这是因为决策模型的使用者是组件容器产品的架构师和开发人员.其中 VPName.Variant 是一个布尔表达式,取值为 true 表示变化点绑定到该可选项,比如 VP3.SOAP/HTTP 表示组件容器需要支持基于 HTTP 的 SOAP 协议.VPName.size 表示变化点中被选择的可选项数目,如某 Servlet 容器需要同时支持 HTTP 和 HTTPS 协议,这时 VP3.size 为 2.

变化点间的约束条件源自组件模型元素间的约束关系.来源之一是模型元素与 PR 元素间的映射关系.如 VP1 与 VP3 相关的 PR 元素来自同一个模型元素,因此,当 VP3 的协议格式与 VP1 的同步行为间存在约束关系时,如表 5 第 1 行 VP1 所示.另一来源是组件模型元素在技术上的兼容性,例如 WS-Addressing 只与 SOAP 协议兼容,故在 VP2 与 VP3 间存在如表 5 第 2 行 VP2 所示的约束关系.

Table 5 Example of decision model

表 5 决策模型示例

Index	Description	Variants	Encap. module	Constraint	Mapping module
VP1	Parse remote message	Syn, Asyn	M1	If (VP3.Messaging) Then (VP1.asyn)	Syn: M1.2 Asyn: M1.3 Syn & Asyn: M1.2, M1.3, M1.4

VP2	Locate component	URL
		JNDI
		WS-Addressing
			M4	If (VP3.RMI-IIOP)
				Then (VP2.JNDI);
				If (VP3.SOAP/HTTP)
				Then (VP2.ws-addressing)
					URL: M4.1
					JNDI: M4.2
					WS-Addressing: M4.3

VP3	Protocol message	HTTP HTTPS RMI-IIOP CORBA-IIOP SOAP/HTTP Messaging	M1		VP3.size>1: M1.1 HTTP: M1.5 HTTPS: M1.6 RMI-IIOP: M1.7 CORBA-IIOP: M1.8 SOAP/HTTP: M1.9 Messaging: M1.10

2.3.2 模块设计

PLACE模块需要在实现变化性需求的同时,保持对外接口的稳定性.因此,在模块设计中我们采用了多种设计模式,并结合了面向对象的设计中的主要设计思想,如继承和重载等.设计模式[25−27]用于为设计中重复出现的问题提供解决方案,并且可以改善设计的某些质量属性,如可扩展性、关注点分离等.

模块设计方案主要受变化点的类型和数量这两个属性影响.

从变化点的类型来看,对于数据类型的变化点,一般采用针对数据访问变化性的实现机制,如类继承、Facade 模式等;控制类型的变化点则采用行为解耦相关的机制,如方法重载、Proxy 模式等;计算类型采用流程解耦的机制,如 Interceptor 模式、Chain of Responsibility 模式等;外部计算类型的变化点除采用计算类型的设计机制外,还会使用 Bridge 模式等支持与外界模块交互.

关于变化点的数量属性,主要考虑其上限,上限可能的值为 1 或者 *n*.上限为 1 表示多个选项之间存在互斥关系,类继承、方法重载、Strategy 模式等适用于这种情况;上限为 *n* 表示需要同时支持多个可选项,对于静态变化点需要提供公共数据结构,对于动态变化点则需要建立统一行为模型,适用的实现方法包括 Adaptor 模式、Facade 模式、Interceptor 模式或 Factory 模式等.

例如 PLACE 中协议处理器模块封装了表 5 中的 VP1 和 VP3,其类型分别为控制类型和数据类型,数量均为[1..*n*].其中我们使用了 Proxy 模式和 Facade 模式,Proxy 模式连接同步或异步消息监听器与消息解析器,Facade 模式则用于向服务管理器模块传递统一格式的参数.

3 实例研究

网驰平台(ONCE)[15]采用组件化软件工程设计方法,以组件容器产品线体系结构PLACE为核心.在平台的中间件系统中,实现有Servlet容器、EJB容器、Web服务容器、Portlet容器、BPEL容器等组件容器.

Servlet组件位于J2EE应用服务器 3 层结构中的表示层,使用Java语言、按照Servlet规范[16]开发,负责生成用户界面代码、处理用户交互.Servlet技术是J2EE中其他表示层技术,如JSP,JSF和Portlet等的基础.下面简要介绍基于PLACE的Servlet容器的设计过程.

在网驰平台开发中,为提高易用性,Servlet 容器还增加了如下自选需求:

1) 热部署功能(hot deployment).在运行时完成 Servlet 应用(WAR 文件)的部署和反部署.
2) Web 管理控制台(Web administrative console).通过 Web 界面对 Servlet 容器及容器中运行的组件及实例进行监控和管理以及远程部署功能.

网驰平台的Servlet容器符合Servlet 2.5[16]规范,使用组件模型分析框架对Servlet 2.5 规范进行分析,其结果见表 6.

Table 6 Analysis of Servlet component model

表 6 Servlet 组件模型分析

Model elements	Servlet CM	Model elements	Servlet CM
Interface def. lang.	Java	Interact. constraint	Same Webapp, or crossContext
Interface constraint	javax.servlet.Servlet	Comm. mechanism	Method invocation (RequestDispatcher)
Implementation lang.	Java	Persistence	Session state
Implementation constraint	Implementing interface javax.servlet.Servlet	Lifecycle event	Method invocation (init, destroy, listeners)
Metainfo desc. lang.	XML	Thread management	Required
Metainfo constraint	Web-app_2_5.xsd	Resource management	Static files, Java libraries, etc.
Naming protocol	URL	Packaging spec.	WAR
Client protocol	HTTP, HTTPS	Deployment desc. lang.	XML
QoS support	Security, session, log, monitor	Deployment constraint	web-app_2_5.xsd
Reentrant	Reentrant, queue-required (SingleThreadMode)	Resource type	EJB-Ref, Data source, message destination, variables

在 Servlet 组件模型分析的基础上,根据 PLACE 的决策模型,实现上述需求的 Servlet 容器体系结构如图 10 所示.图 10 所示的顶层模块体系结构与 PLACE 相比,其变化性体现为:

裁剪:由于 Servlet 组件不需要对中间状态进行管理,所以从 PLA 中去掉了状态管理器模块和相关的调用关系.由于在协议处理器部分进行线程调度,故服务管理器和组件管理器与并发控制器之间的调用关系被裁去.由于不提供对容器服务的监控,故监控管理器与服务管理器的调用关系被裁去.裁剪的部分在图中不再出现.

增加:增加的部分包括远程管理控制台(remote management console)、热部署器(hot deployer)和远程部署器(remote deployer)以及相应的接口和调用关系.由图中阴影的模块和粗线的调用关系来表示.

绑定:按照表 6 的 Servlet 组件模型元素的取值,通过决策模型可以确定 PLACE 中可选模块的绑定方式.以协议处理器模块为例,VP1 绑定为 synchronous,VP2 绑定为 URL,VP3 绑定为 HTTP 和 HTTPS.根据决策模型,M1 协议处理器模块包括了 M1.1 协议适配器、M1.2 同步监听器、M1.5 HTTP 协议处理器和 M1.6 HTTPS 协议处理器.图 11 展示了 M1 模块内部的层次结构.其中 M1.2 模块监听 80 和 8441 端口,并将监听到的连接请求转交给 M1.5 或 M1.6 模块进行解析,经过 M1.1 包装为公共格式后交给 M2(服务管理器)进行后续处理.M1.1 绑定到 M1 模块的需求接口,调用 M8 模块提供并发处理支持.

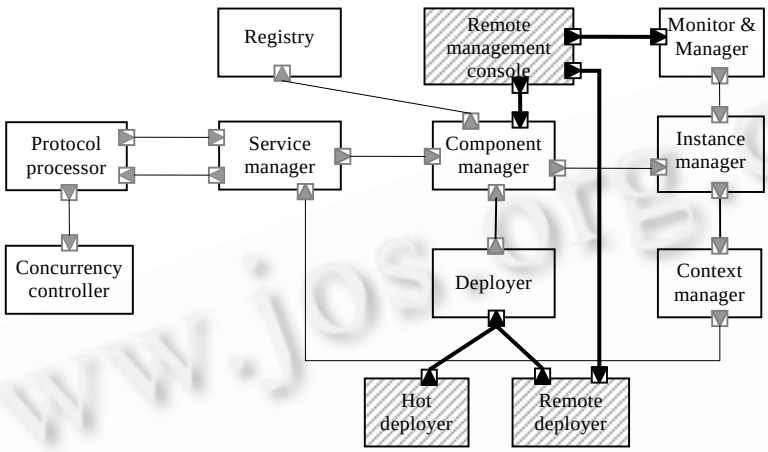

Fig.10 Architecture of Servlet container in ONCE

图 10 网驰平台 Servlet 容器体系结构

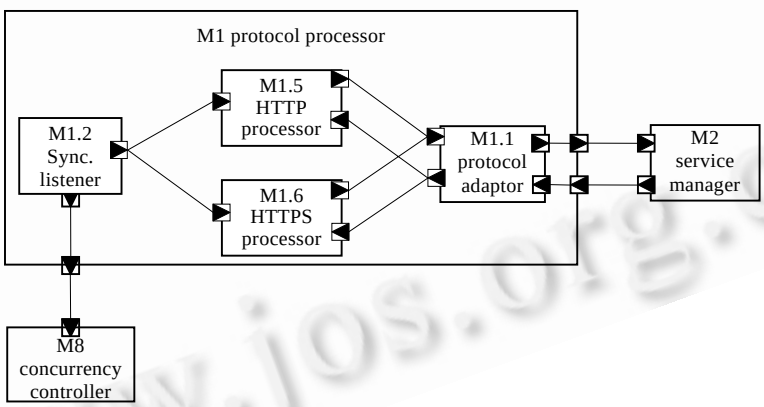

Fig.11 Structure of protocol processor module

图 11 协议处理器模块内部结构

在 M2 服务管理器、M3 组件管理器、M6 上下文管理器与 M7 部署器等模块中,按照决策模型确定模块结构后,还需按照组件模型实现各 QoS 服务、组件生命期事件通知、上下文支持和部署描述文件 web.xml 解析等 Servlet 特有功能.模块接口定义与模块间交互关系均从 PLACE 中复用.

3.1 讨 论

PLACE 中的主要模块在 ONCE 平台各个组件容器中的复用情况见表 7.

从表 7 中可以得到,PLACE 的必选模块(M1~M7)在每个组件容器中均得以复用,平均复用次数为 3.9 次,在 4 个产品构成的产品系列中复用率达到了 96%;可选模块(M8~M10)也出现在半数以上的组件容器中,平均复用次数为 2.8 次,产品系列内的复用率也达到了 63%.在整个组件容器产品系列中,复用 PLACE 模块的比例达到了

77%.

以上数据说明,基于PLACE进行组件容器的设计有效促进了结构复用,从而减少了设计过程中所需做出的设计决策数量,提高了生产开发效率.

基于兼容性测试,功能测试和性能测试的产品质量检验表明,基于PLACE开发的组件容器质量不低于原有版本的组件容器.

Table 7 Reusing statistics of PLACE modules

表 7 PLACE 主要模块复用情况表

Module index	Module name	Servlet container	EJB container	BPEL container	Portlet container
M1	ProtocolProcessor	√	M	√	√
M2	ServiceManager	√	√	√	√
M3	ComponentManager	√	√	√	√
M4	Registry	√	√	√	√
M5	InstanceManager	√	√	M	√
M6	ContextManager	√	√	√	√
M7	Deployer	√	√	√	√
M8	ConcurrencyController	√	M	—	—
M9	StateManager	—	√	√	√
M10	Monitor&Manager	√	√	√	√

Notes: "√" means directly reused, "M" means reused with modification, "—" means not reused

4 相关工作

4.1 组件模型分析方法

目前已有的组件模型分析方法主要有两类:以Lau[1]和Crnkovic[8]为代表的方法致力于建立组件模型分类准则,比较各组件模型的优劣,指导组件技术的发展;而CMU[6],Weinreich[7]等则从CBD的主要概念和技术入手,分析组件模型在CBD中起到的作用,列出其组成部分.本文提出的组件模型分析框架属于后一类.

Lau等人[1]认为软件组件模型定义了组件的3个主要方面:语义、语法和组合,并分别按照各方面对软件组件模型进行分类.组件语义定义为"组件是什么",组件语法定义为组件定义语言的类别,在组件组合方面,Lau提出了一个理想的组件开发生命周期,并按照组件模型在不同阶段对组合的支持情况将组件模型分为4个不同的类别.Lau[1]与Crnkovic[8]的工作目的是对组件模型的分类和分析,因此关注点涵盖开发过程、应用领域、质量属性等组件模型的外在属性,而对组件模型的构成只进行了粗粒度的分析,不适用于领域分析.

CMU[6]认为组件模型是组件开发者所需遵循的标准和约定.为了达到CBD的统一组合、质量保证和独立部署的目的,组件模型应包含组件类型、交互机制和资源绑定的规范.

组件的类型由组件接口所定义.在组件模型中,不同的组件类型规定了组件在系统中充当角色和能够参与的交互机制.交互机制描述了组件之间和组件与容器间的交互方式.资源绑定包括将组件与资源加以绑定的方法.资源可以是容器或者容器中部署的其他组件提供的服务.CMU[6]和Weinreich[7]的方法关注组件模型的构成元素.但这些工作对构成元素内包含的多个同类规范没有进一步划分.本文提出的组件模型分析框架则将组件容器划分为更细粒度的模型元素,以满足领域分析的需要.

4.2 领域分析和设计方法

目前大部分领域分析方法,如FODA[11],FeatuRSEB[28],FODM[4]等,均采用特征模型对领域需求进行刻画和组织.特征定义为软件系统中用户可见的、显著的或具有特色的方面,提供了一种需求的分割和组织方式,即以特征作为需求空间内的一阶实体,系统具有的特征及其相互关系构成了系统的需求空间[4].

特征概念涵盖的范围很广,从系统的整体功能到具体的领域技术和实现手段[12].但这也使特征不具有统一的定义和层次划分,其定义完全依赖于领域专家的判断,不利于建立与组件模型的映射关系.

Bosch在文献[5]中对软件体系结构和产品线体系结构(PLA)的设计方法进行了全面的介绍,包括成本分析、

特征规划、PLA设计等步骤.Matinlassi[29]对 5 种PLA设计方法也作了比较,指出每种方法都有不同的目的或观点,相互之间并不冲突.近年来,PLA的研究集中在PLA变化性的描述[22]、变化性管理[3,30]和实现机制[23]等方面.本文作者尚未发现有本文提出的组件容器PLA类同研究[29,31].

5 结束语

组件模型的多样化和快速演化要求组件容器的开发方法必须与软件复用技术相结合.产品线工程是解决组件容器面临问题的合适途径.进行组件容器 PLA 研究要提取组件模型中蕴含的需求信息并以此为基础进行PLA 设计.

本文提出的组件模型分析框架,弥补了现有组件模型分析方法抽象和粗粒度的不足.将组件模型分析框架与基于原子需求的领域建模方法相结合,通过组件模型元素和原子需求元素之间的映射关系,建立了组件容器领域模型.

本文提出组件容器设计需要符合可复用、可追溯和可扩展的 3 个原则,以求达到提高组件容器结构复用性的目的.在本文建立的组件容器领域模型的基础上,进一步根据变化性封装原则,本文提出了组件容器产品线体系结构 PLACE,并通过引入可选属性、层次结构和决策模型,实现组件容器的领域需求.PLACE 产品线体系结构在网驰平台的 Servlet,EJB 等组件容器设计过程中,展示了良好的抽象性和灵活性.

本文的方法还可以应用于其他各种分布式组件容器的需求分析和体系结构设计.下一步的研究工作包括组件模型的分析和评估、PLA 辅助设计工具以及组件容器非功能需求的支持等.

致谢 在此谨向为本文工作提供支持和建议的老师和同学表示感谢.

References:

[1] Lau KK, Wang Z. Software component models. IEEE Trans. on Software Engineering, 2007,33(10):709−724.

[2] Szyperski C. Component Software—Beyond Object-Oriented Programming. 2nd ed., Boston: Addison-Wesley, 2002.

[3] Clements P, Northrop L. Software Product Line: Practices and Patterns. Boston: Addison-Wesley Professional, 2002.

[4] Zhang W, Mei H. A feature-oriented domain model and its modeling process. Journal of Software, 2003,14(8):1345−1356 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1345.htm

[5] Bosch J. Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach. Reading: Addison-Wesley Professional, 2000.

[6] Bachmann F, Bass L, Buhman C, Comella-Dorda S, Long F, Robert J, Seacord R, Wallnau K. Volume II: Technical concepts of component-based software engineering, 2nd ed., Technical Report, CMU/SEI-2000-TR-008, Pittsburgh: Carnegie Mellon University, Software Engineering Institute, 2000.

[7] Heinemann GT, Councill WT. Component-Based Software Engineering: Putting the Pieces Together. Boston: Addison-Wesley, 2001.

[8] Crnkovic I, Chaudron M, Sentilles S, Vulgarakis A. A classification framework for component models. In: Proc. of the 7th Conf. on Software Engineering Research and Practice in Sweden (SERPS 2007). 2007. 3−12. http://crosbi.znanstvenici.hr/prikazi-rad?lang=EN&rad=393766

[9] Beugnard A, Jézéquel JM, Plouzeau N, Watkins D. Making components contract aware. IEEE Computer, 1999,32(7):38−45.

[10] Gomma H. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Massachusetts: Addison-Wesley, 2004.

[11] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-Oriented domain analysis (FODA) feasibility study. Technical Report, CMU/SEI-90-TR-21, Pittsburgh: Carnegie Mellon University, Software Engineering Institute, 1990. 1−52.

[12] Kang KC, Kim S, Lee J, Kim K, Shin E, Huh M. FORM: A feature-oriented reuse method with domain-specific architecture. Annals of Software Engineering, 1998,5(1):143−168.

[13] Moon M, Yeom K, Chae HS. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. IEEE Trans. on Software Engineering, 2005,31(7):551−569.

[14] OMG. OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2 (formal/2007-11-02). OMG, 2007. http://www.omg.org/spec/UML/2.1.2/

[15] Institute of Software, the Chinese Academy of Sciences. ONCE Platform (in Chinese with English abstract). http://www.once.org.cn

[16] Sun Microsystems. Java Servlet Technology. http://java.sun.com/products/servlet/

[17] Sun Microsystems. Enterprise JavaBeans Technology. http://java.sun.com/products/ejb/

[18] Java Community Process. JSR 168: Portlet Specification. 2003. http://jcp.org/en/jsr/detail?id=168

[19] OASIS. Web services business process execution language version 2.0. 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[20] Van Ommering R, vander Linden F, Kramer J, Magee J. The Koala component model for consumer electronics software. IEEE Computer, 2000,33(3):78−85.

[21] Hendrickson SA, van der Hoek A. Modeling product line architecture through change sets and relationships. In: Proc. of the 29th Int'l Conf. on Software Engineering (ICSE 2007). Minneapolis: IEEE Computer Society, 2007. 189−198.

[22] Sinnema M, Deelstra S. Classifying variability modeling techniques. Information and Software Technology, 2007,49(7):717−739.

[23] Svahnberg M, van Gurp J, Bosch J. A taxonomy of variability realization techniques. Software Practice and Experience, 2005, 35(8):705−754.

[24] Fan GC. Research on some key technologies for Web application servers [Ph.D. Thesis]. Beijing: Institute of Software, the Chinese Academy of Sciences, 2004 (in Chinese with English abstract).

[25] Schmidt D, Stal M, Rohnert H, Buschmann F. Pattern-Oriented Software Architecture-Patterns for Concurrent and Networked Objects. Chichester: John Wiley & Sons, 2000.

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

[27] Buschmann F, Henney K, Schmidt DC. Pattern-Oriented Software Architecture, Volume 4: Pattern Language for Distributed Computing. New York: John Wiley & Sons, 2007.

[28] Griss ML, Favaro J, d'Alessandro M. Integrating feature modeling with the RSEB. In: Devanbu P, Poulin JS, eds. Proc. of the 5th Int'l Conf. on Software Reuse (ICSR'98). Victoria: IEEE Computer Society, 1998. 76−85.

[29] Matinlassi M. Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA. In: Proc. of the 26th Int'l Conf. on Software Engineering (ICSE 2004). Edinburgh: IEEE Computer Society, 2004. 127−136.

[30] Sinnema M, Deelstra S, Nijhuis J, Bosch J. COVAMOF: A framework for modeling variability in software product families. In: Nord RL, ed. Proc. of the 3rd Int'l Software Product Lines Conf. (SPLC 2004). LNCS 3154, Springer-Verlag, 2004. 197−213.

[31] Product Line Hall of Fame. Software Engineering Institute, Carnegie Mellon. http://www.sei.cmu.edu/productlines/plp_hof.html

附中文参考文献:

[4] 张伟,梅宏.一种面向特征的领域模型及其建模过程.软件学报,2003,14(8):1345−1356. http://www.jos.org.cn/1000-9825/14/1345.htm

[15] 中国科学院软件研究所.网驰平台. http://www.once.com.cn

[24] 范国闯.Web 应用服务器关键技术研究[博士学位论文].北京:中国科学院软件研究所,2004.

刘国梁(1983－),男,辽宁喀左人,博士生,主要研究领域为软件工程,软件体系结构.

魏峻(1970－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为主要研究领域为软件工程,网络分布计算,服务计算.

冯玉琳(1942－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为组合式软件工程,网络分布计算,系统形式规范和模型验证.