维普资讯 http://www.cqvip.com

基于业务构件的快速可重构信息系统的框架研究

李绪蓉　凌兴宏　丁秋林
（南京航空航天大学信息科学与技术学院　南京210016）

Research on the Framework of RRIS Based on Business Components

LI Xu-Rong　LING Xing-Hong　DING Qiu-Lin
(College of Information Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

Abstract　Rapidly Reconfigurable Information System (RRIS)and its framework have been hotspots of research circles. The inherent characteristics of a framework offer a direct support for RRIS. Different software technologies will lead to different framework solutions and their reconfiguration mechanisms are very different. This paper gives a proper summary to the framework design of RRIS, and then, analyzes the strength and weakness between Object-Oriented enterprise framework and component-based enterprise framework. Moreover, the framework of RRIS based on business components is put forward and its building, functions, design and implementation are discussed in details.
Keywords　Reconfiguration mechanisms, RRIS, Business component, Framework, Software architecture, Enterprise framework

1　引言

系统可重构的思想蕴含着"适者生存"的"应变"的哲理。快速可重构性已成为智能系统的基本特性[13]，是衡量系统响应变化能力的重要指标，是企业赢得全球化市场竞争的关键因素。快速体现为尽量缩短新产品的上市时间，而系统重构体现为变化来临时系统中的子系统会重新组合而产生新功能、新过程或更高层的整体性[12]。快速可重构信息系统(Rapid Reconfigurable Information System, RRIS)及其框架研究已经成为业界的重要研究课题[15]。框架是半成品的应用程序，设计良好的框架应支持从系统分析、设计到代码级的重用。基于框架的开发可避免重复劳动，降低开发成本，提高软件生产率。重构应以重用为基础，框架的固有特性(模块化、可重用、可定制与可扩展)将对信息系统的快速重构提供直接的支持。不同的软件技术导致不同的框架解决方案，其重构机制也大不相同：OO框架技术利用对象的继承、组合(很少使用)以及设计模式等机制支持系统的重构，而构件化框架技术则通过构件在框架上的即插即用的方式来组建应用，即通过构件的演化、替换及重组等方式来支持重构。二者各有利弊，OO框架容易实现一些，而构件化框架比OO框架更灵活、演化性更好，代表未来的发展趋势。

业务构件技术是构件技术的最新发展，具有下面4个优点：促进了软件生产工业化；可取得生产率的突破；可减少业务技术隔阂；可有效地支持信息系统快速重构。业务构件可以很好地模拟业务过程并将其平滑地转变成软件实现，设计良好的业务构件可通过即插即用的方式，随业务变化而灵活重组。基于业务构件的RRIS框架研究(具有理论价值和现实意义)必将成为未来软件开发的重点。本文首先概要介绍框架有关概念，然后对RRIS框架的设计方案进行详细的比较分析，最后提出基于业务构件的RRIS的框架的建立过程。

2　框架概述

2.1　框架定义

框架的定义较多，反映出人们对框架内涵的不同理解。

李绪蓉　博士研究生，主要研究领域为分布式系统、系统集成与系统重构。凌兴宏　主要研究领域为Agent技术与系统集成。丁秋林　教授，博导，主要研究领域为CIMS技术与系统集成技术。

如果软件类型适合采用原型法和螺旋模型进行开发，以上过程也适用于演化型原型方法。但是如果软件本身的重点在用户界面和用户交互过程上，或者软件没有明显的关键功能，典型的如小型Web应用程序，多媒体教学系统等，在采用形式化方法上获利甚少；如果在快速开发平台上能够比较容易地产生部分功能的原型，那么这部分功能的开发也没有必要采用形式化方法。

在以上的开发过程中，对开发人员掌握形式化开发方法的要求相应降低了。由于整个开发过程采用了UML的过程框架，既可以采用螺旋模型控制整个开发过程，在每个螺旋过程中，又分成多个阶段。形式化方法在这些不同阶段中的应用层次完全不同，在问题定义与分析、可视化建模、代码生成、完善编码和测试等阶段的开发人员，只需要使用UML、面向对象语言及编程、RAD开发环境等开发技术，完全可以不涉及形式化技术。因此，软件开发过程中最重要的模型精化和模型检查过程采用形式化方法，加强了软件模型的精确程度，这也将提高软件的可靠性。

该方法的详细过程与规则和开发实例笔者将在后文中详述。

参考文献

1　周彦晖. 基于面向对象模型的形式化规约和FDOOM开发方法：[硕士论文]. 2002
2　The RAISE Method Group, George C, et al. The RAISE Development Method. TERMA Elecktronik AS, Denmark, 1999
3　Andrews J D, Groote J F, Middelburg C A. Semantics of Specification Languages. Workshops in Computing. Spring-Verlag, 1993
4　Bruun P M, et al. RAISE Tools Refernce Manual: [Technical Report LACOS/CRI/DOC/17]. CRI: Computer Resources International, 1995
5　Fowler M. UML Distilled (Second Edition). Addison-Wesley, 2000
6　Albir S. UML in a nutshell. O'Reilly, 1998
7　Rumbaugh B G, Jacobson J. The Unified Modeling Language User Guide. Addison-Wesley, 1999

维普资讯 http://www.cqvip.com

如:框架是一个系统的部分或全部的可重用设计,包括一组抽象的类库及其实例之间的互操作机制(设计角度的框架);框架是一个应用程序的骨架,该骨架可以由应用程序开发者所定制(功能角度的框架);框架是由多种模式组成的构造型(stereotyped)包,其中的体系结构模式为特定领域的应用提供了可扩展模板(基于 UML 的框架定义,强调了框架、模式与体系结构之间的关系)[4]。

总之,框架是一个可重用的(reusable)、半成品的(semi-complete)应用程序,通过对它的特化或定制可生成最终的应用程序[15]。

2.2 框架分类

根据框架所涉及的范围或领域,框架可分为3类[15]:

①系统基础结构框架,负责操作系统、通信协议、用户接口和 DBMS 等基础开发;

②中间件集成框架,主要用来集成分布式的应用和构件,使其工作在无缝集成的分布式开发和运行环境中;

③企业应用框架,是一种面向对象、基于构件和面向客户应用的框架。

按照扩展或定制框架所使用的技术的不同,框架又分为:

①白盒框架(white-box framework),主要通过继承关系来构造框架,要求父类的内部细节对子类可见;

②黑盒框架(black-box framework),主要通过组合来构造框架,要求被组合的对象具有良好定义的接口,而且用户只需知道被组合对象的接口而不必关心其具体实现细节。

2.3 框架与其它重用技术之比较

·框架、体系结构与领域工程

领域工程旨在重用,它为一组相似或相近系统的应用工程建立基本能力和必备基础,覆盖了建立可重用的软件构件的所有活动,是面向构件的理念工程。在领域工程指导下,可得到领域模型(问题域)、特定领域体系结构(DSSA)和领域构件等一系列可重用产品。

体系结构用于描述构件间的组合及其交互关系。

框架是体系结构的具体实现(DSSA 的实例化),更侧重于解域,旨在可重用与特化。

框架=体系结构+实现+文档[11]

·框架与设计模式

设计模式记录了软件开发过程中获得的成功经验(成功的设计和体系结构),描述了环境中不断出现的问题及其解决方案。复用这些解决方案,可以减少重复劳动、提高效率、降低开发成本。模式不仅可复用,而且支持变化,因而增加了设计的灵活性。设计模式和面向对象框架的设计技术密不可分,甚至可以认为框架就是各种各样的设计模式和构件的有机组合。能否在框架中组入多种软件模式,成为衡量框架成熟度的技术标志[7]。

3 RRIS 框架的设计

3.1 RRIS 框架设计的指导思想

变化引起重构,带动了 RRIS 的研究。在 RRIS 框架的设计中,应处处运用可重构的指导思想。笔者认为可采取"纵向分层、横向分割"的指导思想。

·纵向分层——层次化思想 RRIS 框架的设计应采用层次化思想,自下而上将 RRIS 框架划分3个层次,即基础结构框架、中间件集成框架和企业应用框架(参见2.2节),每一层专注于自己的问题域。这种"进行不同层次的分解和抽象,分而治之"的解决方案,使框架便于重用、扩展和维护。

·横向分割——共性和可变性分析 企业应用框架中的业务逻辑具有潜在的易变性,可采取横向分割的办法,将可变性剥离出来,借用 OO 程序设计的指导原则(发现变化,封装变化),优先使用组合而不是继承[13]。在领域工程的指导下,进行共性(凝固点)与可变性(热点)分析并将其合理地划分出来。具体步骤如下:

建立特定应用模型→建立领域模型(在领域工程的指导下)→进行共性与可变性的确定与分析(SCV 方法[15])→凝固点与热点的设计(上述过程是循环迭代的)。

凝固点与热点的设计方法不同,将导致不同的框架类型(黑盒框架或白盒框架)。

3.2 企业应用框架的设计方案之比较

目前基础结构框架和中间件集成框架相对成熟,本文重点研究企业用框架。

·企业框架的基本特征 企业应用框架的基本特征[15]为:成熟的运行功能;对可定制性和可扩展性的有力支持,企业应用框架必须提供足够多的"热点"("扩展点")来满足框架领域内不同具体应用的特定需求;框架的核心结构要保持稳定,稳定的框架核心与可定制、可扩展的热点的结合,为基于框架的应用提供了一个灵活和可扩展的体系结构及其实现;为分布式对象提供集成化的模型及其可伸缩性;为不同应用框架之间以及和遗留系统之间的集成提供解决方案;平台独立和可移植性;完善的框架文档;对 Web 和 E-business 的支持。

上述特征也是衡量企业框架好坏的标准,本文将参照此标准来设计 RRIS 的企业框架。

·OO 企业框架与构件化企业框架之比较 企业框架的设计技术目前主要有两种:OO 技术与构件化技术,对应有 OO 企业应用框架[5,6,11,15]和构件化的企业应用框架[3,10]。表1详细地对这两种解决方案进行了比较。

·第三种解决方案——"黑白"混合式企业应用框架的可行性分析 分析表1可见:OO 企业框架的优点(缺点)恰是构件化企业框架的缺点(优点),两者互补。虽然理想的企业框架应该是构件化的黑盒框架,但实际中多数都是基于 OO 的白盒框架。笔者认为较现实的解决方案应该采取折中的办法,即以构件化的黑盒框架为主、基于 OO 的白盒框架为辅的"黑白"混合式企业应用框架。

表1 OO 企业框架与构件化企业框架

	OO企业框架	构件化企业框架
扩展框架的方法/机制	侧重于类继承的白盒机制	侧重于构件组合的黑盒机制
热点(可变性)的设计	基于设计模式的热点子系统	适应性热点构件
凝固点的设计	模板方法/抽象类	领域构件库
框架核心与热点间的耦合度	高	低
优点	类继承是在编译时刻静态定义的,可直接对代码进行修改,以适应变化。	强调封装,只能通过接口访问,接口与实现的分离,只要接口类型一致,运行时刻构件可相互替代。
能否在运行时刻添加或修改框架的功能?	否	是
存在的问题	脆弱的基类问题:①无法在运行时刻改变从父类继承的实现;②子类暴露其父类的实现细节,破坏了封装;③父类、子类之间耦合紧密,一方变化必波及另一方。	①构件不易修改,很难演化;②市场上能提供的可用构件很少或没有;③构件间的交互/集成问题较复杂,很难掌握。

维普资讯 http://www.cqvip.com

4 基于业务构件的RRIS框架的建立

基于业务构件的RRIS框架，兼顾了业务构件技术的优势及框架设计方案的可行性，本节讨论有关细节。

4.1 业务构件技术

业务构件(W. Koyacyuski 定义[1])是业务对象的软件实现，可由若干软构件复合而成。它表达了自治的业务概念，是信息系统的构成元素。下面从两方面介绍业务构件技术：业务构件系统及其体系结构。

·业务构件系统 为了适应变化，提高信息系统的演化能力，在构件开发的各个阶段，应尽量避免构件间存在太多的交叉关联，因而有必要对所开发的业务构件系统[10](也称业务构件集，是一组通过协作来发布系统功能的业务构件)进行层次划分。业务构件系统可分为3层：业务过程构件、业务实体构件和实用构件(如图1)。业务过程构件代表领域本身的业务活动，如发票管理、订单管理等；业务实体构件表达了业务过程要运用的业务概念，包括数据、存储对象以及有关服务；实用构件可由任何更高级别的业务构件使用，如地址簿。

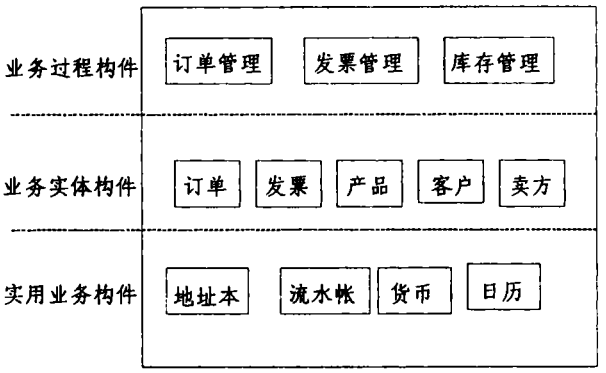

图1 业务构件系统

·业务构件系统的层次体系结构 业务构件系统是一组构件集合，从分布式应用的角度，可将它们布置在不同的逻辑层上协同作用。一个分布式的业务构件系统可分为4个逻辑层次[10]：用户层、工作空间层、企业层、资源层，它们与其物理结构并非直接对应，可根据需要灵活配置。

4.2 基于业务构件的RRIS框架的建立及其功能

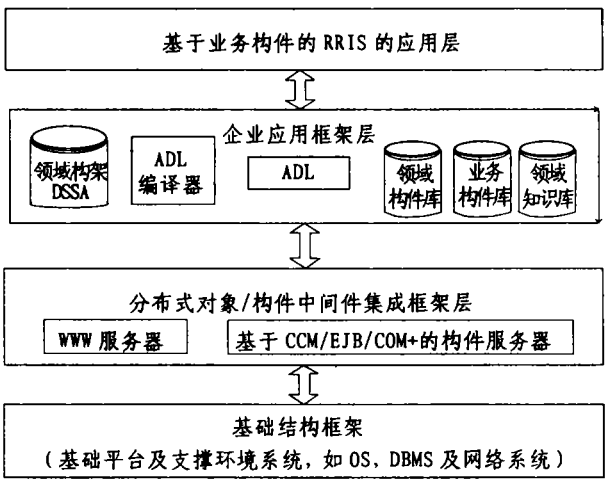

图2 基于业务构件的RRIS的体系结构框架

根据3.1节的"纵向分层、横向分割"的指导思想并利用业务构件技术，笔者提出基于业务构件的RRIS框架，如图2。框架是系统开发环境的一部分，应包括：服务、工具、方法及可重用构件库及框架的层间交互等。

基于业务构件的RRIS框架具有下列功能：

1. 分布式应用/构件在框架上的即插即用；
2. 构件互操作；
3. 企业框架与中间件框架的集成；
4. 业务解决方案与Web应用(Internet/Intranet)的集成。

4.3 混合式企业框架的设计

可在领域工程和重构工程的指导下完成稳定的核心结构的设计和反映变化的热点的设计。

·领域工程与重构工程相结合的企业框架设计策略 领域工程包括领域分析、领域设计、领域实现3个阶段，产生领域模型、领域构架、领域构件等一系列可重用产品。领域工程侧重于问题域，而框架更侧重于解域(DSSA的实例化)，因而在企业框架层分别建立了领域构架库和领域构件库。从业务构件开发的角度，笔者认为领域构件是领域内公共的业务构件，是业务构件的子集。除上述领域构架库/构件库，还需建立特定应用的业务构件库，以适应具体的应用需求。框架的稳定的核心结构可以通过领域构架来组织，用领域构件库及特定的业务构件库中的构件来搭建。

开发可重构的信息系统应注意处处运用可重构的设计思想，既要注意规范设计，又要考虑冗余设计：规范设计尽量应用标准化构件技术(如标准构件模型CCM/EJB/COM+)来设计各功能子系统并使其构件化，标准化程度越高，互换性越好，可重构能力越强；冗余设计提供一定的备用功能和可扩展功能(如热点功能分析)，而对热点设计的不同解决方案决定了企业框架的风格：白盒框架、黑盒框架或"黑白"混合框架。热点处反映了企业业务过程的变化，而变化是形形色色的。变化不同，应对策略亦不同。设计模式提供了23种基本的模式[14]，可满足大部分的变化需求。设计模式的关键目标在于支持变化，本文倾向于用设计模式来设计热点部分(基于设计模式的热点子系统，存放在领域知识库中)，如用State设计模式来解决订单状态的不确定问题；用基于钩方法的模板方法(Template method，行为型设计模式)来解决订单中的信用检查策略的多变性(通过OO的多态机制实现热点的动态绑定)。因为变化太复杂，用构件的方法来实现难度太大。当然也不排除将一些有规律可循的变化做成热点构件，存放在领域知识库中，最终走向全黑盒的理想框架。热点构件的设计可参考下述方案：类属构件模型[8]类似参数化模板，提供了一种柔性的机制以适应变化，它将构件集合中的共性抽取出来，把差异部分/可变化的部分用参数的形式表示；基于动态构件框架的构件演化方法[9]。

综上所述，稳定的核心结构的设计可采用构件化方法(黑盒)；而反映变化的热点部分的设计，主要采用基于设计模式的热点子系统(白盒)，也可设计一些热点构件(黑盒)。存在的问题是：这两种机制如何糅合到一起？框架是可重用的，框架的重用方式分为两种：白盒和黑盒。可重用的含义是在扩展框架的设计时，不同的应用框架通过适配器(Adpter)可以合成更大的框架。黑盒重用和白盒重用是一个交替的过程，可借助于白盒重用对框架进行扩展，随着越来越多的设计被抽象、提炼而形成构件库，白盒重用会逐步过渡到黑盒重用。

·软件体系结构是系统重构的蓝图 软件体系结构是软

维普资讯 http://www.cqvip.com

件更高层次的抽象，主要研究的是软件系统的结构模型、风格和样本，注重全局而不是具体细节。软件体系结构和构架的含义略有不同：构架是基于构件的软件体系结构（可看成是大粒度的构件，是构件化的体系结构），显然软件体系结构的构成元素并不限于构件。体系结构的定义并不统一，其中以 Garlan 和 Shaw 的定义[14]较为流行，即所谓的3C 模型：

体系结构 ＝构件(component)＋连接器(connector)＋约束(constraint)

体系结构描述了构件间的组合及其交互关系。构件是局部，体系结构是整体，经过体系结构的整体布局才能形成应用风格。体系结构的优势：体系结构蓝图可供系统分析，判断是否符合应用需求；可指导设计与实现；便于交流和改进；便于系统的装配。

体系结构的描述方法也不统一，主要有框线图法和形式化的 ADL（体系结构描述语言）[6]。框线图法直观但缺少 ADL 的严谨性；ADL 种类繁多，各有特点，它们从不同角度诠释了对体系结构的理解，描述语法不同且互不兼容。ADL 包括两部分：

ADL：：＝{语法分析，编译器}

语法分析用于保证体系描述格式正确，编译器用于保证语义正确。

·小结　领域工程和重构工程相结合的框架设计策略，构造了快速可重构的应用工程。根据特定的应用系统需求，在体系结构的指导下，从上述三种库（领域构架/构件库、业务构件库和领域知识库）中选用合适的构架/构件/热点子系统（热点构件）并对其进行重组，生成满足用户需求的应用系统。当变化来临时，修改可变部分（热点子系统/构件），从而进行系统的动态集成和重构。

4.4　企业应用框架与中间件集成框架的集成

·基于构件模型的中间件集成框架的作用　从分布式对象计算发展而来的基于构件模型的中间件集成框架，以 CORBA 的 CCM、COM＋和 EJB（三大构件标准）为代表。鉴于微软的 COM＋太依赖于其平台，这里侧重于讨论 CCM 与 EJB。在 CCM 形成之前 EJB 就早已存在，而 CCM（被誉为未来构件模型的典范）在汲取 EJB 优点的基础上，建立了比 EJB 更加完善的规范。CCM 与 EJB 有许多相似之处：都是服务器端的框架，都提供服务器端的容器模型以支持构件的运行，并通过 HOME（构件宿主）管理构件的生命周期。两者的不同之处：CCM 具有语言无关性，而 EJB 仅限于 Java 语言；CCM 提供的功能部件较 EJB 更完备；CCM 构件模型的开放性较 EJB 好；而 CCM 的支撑平台的成熟性比 EJB 差。

构件模型思想旨在：复用、高层开发、简化开发过程、降低开发费用、提高产品质量等。构件模型定义了构件的基本体系结构、构件界面的结构及与其他构件及容器相互作用的机制等。利用构件模型规范，“构件”开发人员开发构件（实现应用系统逻辑的构件），而“系统”开发人员将这些构件组合成应用系统，充分体现了行业分工的思想（恰当的人做恰当的事，提高效率）。

基于构件模型的中间件集成框架，将构件模型的开发同中间件技术联系起来。企业级应用系统的中间件以其复杂性（涉及应用逻辑、并发性、伸缩性及不兼容系统的组合等问题）而著称，构件模型（服务器端的框架）解决了中间件开发的复杂性问题，它使得开发人员可集中于应用系统的逻辑部分，而不用处理同步、可伸缩性、事务集成、网络、分布式对象框架等一些分布式应用系统中存在的复杂的细节问题。

基于构件模型的中间件集成框架承上启下：向下，由构件服务器负责与分布式计算及操作系统有关的底层细节（如通讯协议、多线程、负载平衡等）；向上，由容器提供构件的生存环境和各种服务（如资源管理、事务支持、并发性管理及安全性管理等），容器和构件服务器共同组成了构件的运行环境。

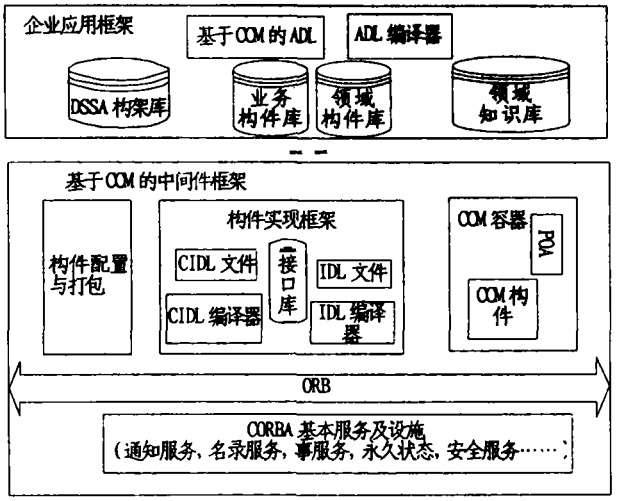

图3　企业应用框架与中间件框架的集成

·企业应用框架与中间件框架的集成　企业应用框架与中间件框架的集成是一个复杂的问题。这里以 CCM 为例进行说明，如图3。

(1)CCM 模型的演化。软件技术的发展促进了构件内涵及构件模型的演化。很长一段时间（直到 CORBA3.0才提出了 CCM— CORBA Component Model）都没有出现真正的用于描述构件及其装配关系的构件模型。这之前的 CORBA 模型，本质上是分布式对象模型（基于对象模型的构件开发，工作量很大）。没有标准的构件模型，就没有真正的即插即用构件。CCM 提供了标准的构件描述、管理及配置方案，可使开发者以更小的工作量应用构件和集成实现服务。

(2)CCM 对于 CORBA 的扩展。CCM 对 CORBA 的扩展主要体现在对 ORB 和 IDL 的扩展上。

CCM 规范扩展了 ORB 的功能，简化了构件框架的实现工作。这些扩展并没有直接影响构件开发者开发构件，而是帮助开发者提高构件的性能。CCM 规范引入了本地接口（local）的概念，用于在 IDL 文件中定义只限本地使用的接口，这样消除了 IDL 描述接口的语义模糊性，提高了构件工作性能。

接口库是 CORBA 的一个标准机制，允许应用在运行期发现接口的 IDL 定义和元数据。CCM 规范扩展了接口库，以支持构件端口机制的定义和提供遍历构件所有接口的操作。

(3)CCM 的核心内容。CCM 定义了抽象模型、构件实现框架模型、容器模型、构件配置及打包模型。抽象模型提供了用于描述构件的接口和特性的方法——CIDL（构件实现定义语言）；构件实现框架模型主要完成构件实现的部分自动化，CIDL 文件通过 CIDL 编译器生成构件框架（扩展该框架即可完整地实现一个构件）及构件描述（将用于构件配置模型）；容器模型提供构件运行的服务器端环境，管理和创建构件实例，并为构件行为加载服务（事务服务、安全服务、事件服务和永久性服务）；构件配置及打包模型和抽象模型、容器模型相互协作，共同构成完整的分布式企业服务器体系结构，完成构件的连接及实现构件的动态集成。

维普资讯 http://www.cqvip.com

(4)企业应用框架与中间件框架集成的关键。①按照CORBA标准,采用IDL/CIDL对构件进行描述,设计领域构架、领域构件、业务构件及领域知识库中反映热点变化的热点构件和基于设计模式的热点子系统,并将其纳入接口库中,统一到CORBA环境下,便于统一管理;

②针对ADL现状(没有基于构件标准的ADL),自行设计基于CCM的ADL,以便充分利用CORBA机制的优点;

③设计ADL编译器,经过ADL编译器即可将ADL变成可执行系统。

总之,体系结构是构件的管理者、组织者和领导者,它就像乐队指挥,用它那充满魔力的手,将乐队的成员——构件调动起来,按业务流程运转,完成系统的各个功能,实现系统的重构和集成。

结论 本文从代表构件技术最新发展方向的业务构件入手来研究RRIS框架,角度新颖。对RRIS框架的整体设计采用"纵向分层,横向分割"的指导思想,不同级别的抽象可以化解问题的复杂性。对企业应用框架的设计是本文的关键,"黑白"混合式的设计策略,兼顾理想和现实两方面的考虑,是一种较实际的解决方案。框架是半成品的应用程序,其设计与实现是比较复杂的,限于篇幅,本文归纳性地阐述了框架的整体与部分的关系以及框架层间的集成。

参考文献

1 Kozaczynski W. Architecture framework for business components. In: [C] 5th Intl. Conf. on Software Reuse, computer socity press, 1998. 300～307

2 Baster G, Konana P, Scott J E. Business Components: A Case Study Of Bankers Trust. Communications of the ACM, 2001, 44 (5)

3 Fingar P. Component-based Frameworks For E-Commerce. Communications of the ACM, 2000, 43(10)

4 Kobryn C. Modeling Components and Frameworks With UML. Communications of the ACM, 2000, 43(10)

5 Chan S M, Lammers T L. Reusing a Distributed Object Domain Framework. In: [C] 5th Intl. Conf. on Software Reuse, computer socity press, 1998

6 Lu Jian, Li Yingjun, Ma Xiaoxing, Tao Xianping. A Hierarchical Framework For Parallel Seismic Applications. Communications of the ACM, 2000, 43(10)

7 何克清,应时,田中茂,等.业务应用软件框架的一种分析方法.软件学报,12(7)

8 李景峰,刘西洋,陈平.一种可重用构件模型——类属构件.计算机科学,1999,26(9):83～86,80

9 符进强,汪洋,钱乐秋.基于动态构件框架的构件演化.计算机科学,2001,28(1)

10 刘晓铭,刘积仁,李华天.构件化领域框架设计与实现.计算机研究与发展,1999,36(2)

11 Butler G. Object-Oriented Frameworks. http://www.cs.concordia.ca/～faculty/gregb

12 万麟瑞,李绪蓉.系统集成方法学研究.计算机学报,1999,22(10)

13 李群明,张士康,王成恩,宋国宁.可重构的企业管理信息系统. http://sy.863cims.net/lab2

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

15 沈延森.快速可重构信息系统极其关键技术研究:[南京航空航天大学博士学位论文]. 2001.5

(上接第102页)

25 Karimi J, Zand M K. Asset-based system and software system development-a frame-based approach. Information and Software Technology, 1998, 40: 69～78

26 Knight K. Unification: a multidisciplinary survey. ACM Computing Surveys, 1989, 21(1): 93～124

27 Krueger C W. Software Reuse. ACM Comp. Surv., 1992, 24 (2): 131～183

28 刘叙华.基于归结方法的自动推理.北京:科学出版社,1994

29 Mili H, Ah-Ki E, Godin R, Mcheick H. Another nail to the coffin of faceted controlled-vocabulary component classification and retrieval. In: Proc. of Symposium on Software Reusability'97 (SSR'97), ACM Software Engineering Notes, 1997, 22(3): 89～98

30 Mili H, Mili F, Mili A. Reusing software: issues and research directions. IEEE Trans. Soft. Eng., 1995, 21(6): 528～561

31 Mili H, Marcotte O, Kabbaj A. Intelligent component Retrieval for software reuse. In: Proc. of 3rd Maghrebian conference on artificial intelligence and software engineering. Rabat, Morocco, 1994. 101～114

32 Mili R, Mili A, Mittarmeir R T. Storing and retrieving software components: a refinement based system. IEEE Trans. Soft. Eng., 1997, 23(7): 445～459

33 Mili R, Raymond J. Towards a formal framework for software reuse. Inform. Sci., 1998, 110: 135～149

34 Minsky M. A framework for representing knowledge. In: Psychology of Computer Vision (Winston P H eds.), McGraw-Hill, 1975

35 Nishida F, Takamatsu S, Fujita Y, Tani T. Semi-automatic program construction from specifications using library modules. IEEE Trans. Software Engineering, 1991, 17(9): 853～871

36 Park Y. Software retrieval by samples using concept analysis. The Journal of systems and Software, 2000, 54: 179～183

37 Rine D C, Sonnemann R M. Investments in reusable software. A study of software reuse investment success factors. The Journal of systems and Software, 1998, 41: 17～32

38 Waldmann U. Semantics of order-sorted specifications. Theoretical Computer Science, 1992, 94: 1～35

39 王家兵,徐正权,王能超.RLD演绎及子句蕴涵与子句包含关系的非等价性.计算机研究与发展,已录用

40 Wing J M. Writing larch interface language specifications. ACM Trans. Programming Languages and Systems, 1987, 9(1): 1～24

41 杨芙清.软件复用及相关技术.计算机科学,1999,26(5):1～4

42 Ye Yunwen, Fischer G, Reeves B. Integrating active information delivery and reuse repository systems. In: Proc. of the ACM SIGSOFT Eighth Intl. Symposium on Foundations of Software Engineering, San Diego, Calif., ACM Soft. Eng. Note., 2000, 25 (6): 60～68

43 Zaremski A M, Wing J M. Signature matching: A tool for using software libraries. ACM Trans. Soft. Eng. Meth., 1995, 4(2): 146～170

44 Zaremski A M, Wing J M. Specification matching for software components. ACM Trans. Soft. Eng. Meth., 1997, 6(4): 333～369