基于业务构件的快速可重构信息系统研究

李绪蓉　丁秋林
（南京航空航天大学计算机系　南京210016）

Research on Business Component-Based Rapid Reconfigurable Information System

LI Xu-Rong　DING Qiou-Lin
(Department of Computer, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

Abstract　The present situation of research on Rapid Reconfigurable Information System (RRIS) is analyzed. And then, the advantages of BCRRIS (Business Component-based RRIS) are summarized. Moreover, strategies of BCRRIS are discussed briefly. The emphasis is paid on the key technologies of BCRRIS, including software architecture of BCRRIS, business components, component model (CORBA CCM) and a framework of BCRRIS.

Keywords　RRIS, BCRRIS, Business component, Component-based software architecture, Component model CORBA CCM, Software framework, Middleware technology, Design pattern

1　引言

系统可重构的思想蕴含着“适者生存”的“应变”的哲理，生活中的例子随处可见，如生物组织的再生、社会结构的变迁、人类的繁衍生息等。重构旨在适应变化，快速可重构性已成为智能系统的基本特性[2]，是衡量系统响应变化能力的重要指标，也是企业赢得全球化市场竞争的关键因素。快速可重构信息系统（下文简称RRIS——Rapid Reconfigurable Information System）已经成为业界的重要研究课题[1~4]。

目前支持系统可重构的理论有：“多代理”、“HOLON制造”、“生物制造”等理念[3]，而系统可重构的实现技术/机制则相对滞后。基于OO技术的可重构实现方法[4]，主要通过支持变化的设计模式（白盒机制，重构时需要了解有关代码）来实现重构，显然此法并不简便。文[3]提出基于组件装配的分布式可重构企业信息系统的重构方案，采用的是分布式对象技术，比OO方法前进了一步，但还不是真正意义上的构件/组件技术，本质上仍属白盒机制，仍要了解实现细节（文中通过修改构件的某一函数来实现某功能的重构）。系统重构的真正理想的解决方案应该是基于黑盒机制（无需了解实现细节）的即插即用构件的装配和重组。为此，本文提出基于业务构件的快速可重构信息系统（BCRRIS），采用业务构件技术来开发RRIS，以弥补现有可重构实现技术方面的不足。本文首先讨论基于业务构件开发RRIS的优点，然后探讨BCRRIS的重构策略，在此基础上重点研究BCRRIS的关键技术。

2　BCRRIS的优点

W. Koyacyuski[11]将业务构件定义为：是业务对象的软件实现，表达了自治的业务概念；是自治的、可复用的信息系统的元素，可由若干软构件合成。基于业务构件开发RRIS具有下述优点：

·业务构件技术促进了软件生产工业化　基于业务构件的开发方法代表着软件技术的最新发展方向，即第五代技术。CBSE（基于构件的软件工程）极大地变革了传统的开发模式[8]，旨在以即插即用构件的方式在框架上组建满足用户需求的应用系统，促进了软件行业的分工（构件生产者和构件装配者）。然而目前的构件技术，尚不能使软件的生产达到类似计算机硬件产业的工业化水平。业务构件，以统一的、独立自治的概念贯穿整个软件产品生命周期，每个业务构件都能独立于其它业务构件而存在，被另一构件代替时无需重新编译系统，从而实现真正意义上的即插即用。业务构件技术为软件生产工业化提供了理论与技术基础，促进了软件生产工业化。

·业务构件技术可取得生产率的突破　大多数IT组织运用对象技术，只能提高10%～30%的生产率，大部分原因在于缺乏软件构造自动化的能力以及没有成功地在业务应用中进行大规模的构件重用。只有大粒度的可重用构件（如业务构件），才能使新产品或改进产品快速而灵活地交付到市场上，才能取得生产率的突破。

·业务构件技术可减少业务技术隔阂　信息系统的开发人员（技术专家）缺少领域经验/专门技术，而业务用户缺少对系统进行维护的技术能力，因而造成业务技术隔阂。为了减少业务与技术隔阂人们提出许多方法，如业务对象和构件技术。业务构件将基于封装的构件技术与业务对象融为一体以期进一步减少业务技术隔阂[14]。

·业务构件技术有效地支持信息系统快速重构　上述三大特点奠定了有效地支持信息系统快速重构的基础。企业赢得全球化市场竞争的关键在于：快速（尽量缩短新产品的上市时间T）重构（原有产品/应用适应“变化”的快速演化能力）。业务构件是信息系统的元素，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。因此基于业务构件的信息系统具备良好的适应变化的演化能力。

3　BCRRIS可重构策略

3.1　信息系统重构范围的界定

本文以制造业信息系统为例来研究信息系统可重构问题。制造业信息系统的可重构能力涵盖面很广，包括所有的制造活动和过程，如组织结构重构（如虚拟企业）、业务过程重构（BPR）、功能应用软件重构（如物料需求计划、生产计划、财务

管理、质量管理、运输调度等)、产品重构、车间制造系统重构、信息平台重构等。为了简化问题的复杂性,本文将重构范围限定在:功能应用软件重构或业务过程重构。可将实际应用功能模块构件化,通过构件的演化[13]、更替来实现功能重构;通过构件(代表业务过程所需要的有关要素)的重用和重组来实现业务过程重构。

3.2 领域工程和重构工程相结合的重构策略

信息系统的快速重构应以重用为基础(避免从头开始),可采取领域工程和重构工程相结合的重构策略。

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。领域工程包括领域分析、领域设计、领域实现三个阶段,产生领域模型、领域构架、领域构件等一系列可重用产品,分别存放在领域模型库、领域构架库及领域构件库,作为可复用资源。

重构工程体现为变化来临时,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[5]。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,尽量应用标准化技术和有关协议(如标准构件模型 CCM/EJB)分别设计各功能子系统/模块并使其构件化,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,进行"热点"——易变化功能分析,形成热点构件存放在领域知识库中,保证一定的冗余度,提高系统的可伸缩性。热点构件的设计可参考下述几种方案:构件模板法(可规范化、可参数化和可实例化的模块语义结构或程序源代码);类属构件模型[15],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[13]。

除上述可复用产品库(领域工程的结果)及领域知识库(重构工程的结果),还需建立特定的业务构件库,以适应具体的应用需要。从业务构件开发的角度,这里有必要对领域构件与业务构件的关系进行说明:领域构件是领域内公共的业务构件,是业务构件的子集。这种领域工程和重构工程相结合的重构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,在上述三种库中选用合适的构架/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(热点构件),从而进行系统的动态集成和重构。

4 BCRRIS 关键技术

以即插即用业务构件的方式在框架上组建满足用户需求的应用系统,是 RRIS 的真正理想的解决方案,然而实现起来有很大难度。基于业务构件组装的 RRIS 的关键技术包括:基于构件的软件体系结构/构架、构件模型、框架、业务构件。如果把基于业务构件的软件体系结构比作是 RRIS 的装配蓝图,业务构件看成是 RRIS 的基本构造块,则框架就是实现即插即用业务构件的插槽,而构件模型(CMM/EJB)为构件组装提供标准化的管理及运行级的支持,为信息系统的重构提供了基础和保障,简化了框架的设计与实现。

4.1 构件模型标准

·软件技术的发展促进了构件内涵及构件模型的演化

构件的概念由来已久,其内涵随着软件技术的发展而演化(如表1)。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。

表1 软件技术的发展促进了构件内涵及构件模型的演化[9]

软件技术	构件内涵	构件交互	构件接口描述	构件模型	存在的问题
过程化设计	模块(过程和函数)	过程调用	×	×	集成关系固定在构件实现中,要了解实现细节,缺少灵活性。
OO 设计	对象	消息通讯	×	×	
分布式对象设计(如 CORBA2.X)	分布式对象	ORB 总线	IDL	对象模型	接口标准开始出现,接口与实现相分离,有一定的灵活性。但缺乏标准化的对象及对象生命周期管理,构件开发工作量很大。
构件化设计(如 CORBA3.0)	分布式构件:是对象的软件实现(以 DLL 形式存在)	ORB 总线(扩展了接口库,以支持构件的定义和遍历)	CIDL	构件模型(CCM)	CCM 提供了标准的构件描述、管理及配置方案,只是标准复杂,其实现尚不成熟。

·三种构件模型标准 目前业界存在三种构件模型标准,CCM、EJB 和 COM+。这里侧重于 CCM 与 EJB 的比较。在 CCM 形成之前 EJB 就早已存在,而 CCM 在汲取 EJB 优点的基础上,建立了比 EJB 更加完善的规范,被称之为未来构件模型的典范。CCM 与 EJB 有许多相似之处: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。

·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(构件实现定义语言 CIDL);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。

4.2 业务构件是系统重构的元素

业务构件是信息系统重构的元素。要做到业务构件在框架上的即插即用，促使系统的灵活演化。构件应具备四个本质属性[12]：①关联独立性：一构件无需了解与其交互的其它构件的情况即可自由地装配。②位置透明：可按不同的配置装配构件。③异构连接：应支持异构构件（构件运行在不同的技术体系结构支持环境中而导致异构）的合成。④演化能力：应允许构件很容易地增加、删除或替代。CCM 模型对上述属性提供基本的支持。

·业务构件系统　业务构件系统（也称业务构件集）[10]是一组通过协作来发布系统功能的业务构件。如果构件在任何开发阶段有太多的交叉关联，则系统很难演化。为此业务构件系统应该层次化（做到最少关联），可分为三层：业务过程构件、业务实体构件和实用构件（如图1）。业务过程（process）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（entity）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（repository objects）以及有关服务；实用构件（Utility），可由任何更高级别的业务构件使用，如地址簿。

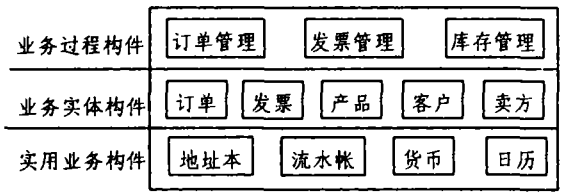

图1　业务构件系统

·业务构件系统的层次结构　一个分布式的基于业务构件的系统可分为四个逻辑层次[10]：用户层、工作空间层、企业层、资源层。这四个逻辑层与 C/S 物理结构并不是直接的对应关系，可根据需要灵活配置。业务构件系统是一组构件集合，它们可分布在不同的逻辑层上协同作用，详见文[7,11]。

·业务构件接口描述　可采用 CIDL 构件实现定义语言对业务构件接口进行描述，CIDL 对构件的属性、构件间的依赖关系及构件的生命周期管理做了详细的规定。

4.3 软件体系结构是系统重构的蓝图

·软件体系结构概述　软件体系结构（以下称 SA）是软件更高层次的抽象，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 SA），显然软件体系结构的构成元素并不限于构件。SA 的定义并不统一，不同定义反映了对软件体系结构的不同理解，其中以 Garlan 和 Shaw 的定义较为流行，即所谓的3C 模型：

SA = 构件（component）+ 连接器（connector）+ 约束（constraint）

对 SA 的表达与描述方法也不统一，主要有框线图法和体系结构描述语言 ADL[6]，用 ADL 与 UML 集成的方法来描述体系结构[14]也日渐流行。框线图法直观但缺少 ADL 形式化描述的严谨性；ADL 种类繁多，各有特点，它们从不同角度/侧面诠释了对于体系结构的理解，描述语法不同且互不兼容。

前面我们把软件体系结构比作 RRIS 的装配蓝图，因为体系结构定义了组成系统的构件及构件之间的相互作用关系。鉴于构件模型（如 CCM）为构件的描述、管理与组装提供了很好的思路和技术支持，本文将构件模型与体系结构的优点结合起来，提出基于构件模型的 BCRRIS 的框架体系结构（如图2），以减少开发难度和成本并尽量避免重复开发。

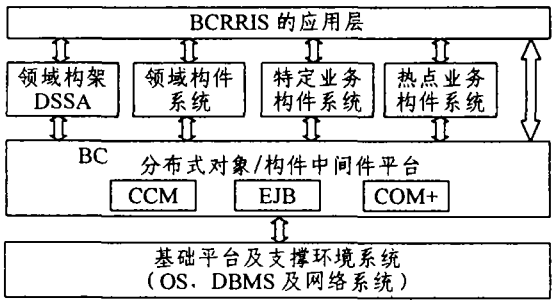

图2　基于构件模型的 BCRRIS 的框架体系结构

·BCRRIS 的体系结构的特点

（1）通过层次结构融合了不同体系结构风格（领域构架及业务构件系统可采用不同的体系结构风格），其中以层次化体系结构风格为主（优点：它支持不同抽象级别的设计，分解问题的复杂性；便于扩展和维护；有利于重用），四层结构分别为：基础平台及支撑环境层，分布式对象/构件中间件平台层，业务构件/构架系统层和 RRIS 应用层。

2）支持系统的快速可重构（参见业务构件/构架系统层）：当变化来临时，重用可复用部分（领域构架/构件），修改可变部分（热点业务构件），从而进行系统的动态集成和重构；或者通过构件/构架的重组来支持重构。

3）基于构件模型标准：根据标准的构件模型规范（如 CCM/EJB），利用其对构件的标准化描述、标准化管理和标准化配置，以提高构件的互换性及系统的可重构能力，同时利用构件模型平台（MICO/J2EE）提供的构件开发的部分自动化支持，从而简化构件的开发过程。

4.4 框架是即插即用构件的插槽

·框架概述　框架的概念。框架[4]（framework）是一组互相交互的、可重用、可定制、半成品的应用程序及软件构件的集合。开放的构架是指允许以"即插即用"的方式在框架上轻易地删除或替换构件。封闭的构架则没有这种灵活性。

框架的分类。根据框架所涉及的范围或领域，可分为三个层次：系统基础结构框架（对应于图2，层1），负责操作系统、通信协议、用户接口和 DBMS 等基础开发；中间件集成框架（对应于图2，层2），主要用来集成分布式的应用和构件，CCM 为构件的描述、管理与组装提供了技术及运行级的支持，增强了构件开发者开发业务应用能力，并使他们能够工作在一个无缝集成的分布式开发和运行环境中；企业应用框架（对应于图2，层3-4），是一种基于业务构件的、面向客户应用的框架。

BCRRIS 框架特点。基于业务构件的快速可重构集成框架具有如下几个特点：框架是面向特定领域的，如制造业信息系统；框架的构成元素是业务构件/构架；框架是可重用、可定制、可扩展、可快速重构的；框架提供可集成机制。其中，可重用性与可重构性是 BCRRIS 框架的两个最基本特点：框架的可扩展性和可维护性，以可重用为基础，以重构机制为支撑。

·框架的即插即用集成机制　以即插即用业务构件的方式在框架上组建满足用户需求的应用系统的难点和关键在于：构件的组装/集成。框架是实现即插即用构件的插槽，构件的组装应以软件体系结构为蓝图，以 CCM 构件模型及其支

持平台为支撑。图3给出基于CORBA软总线的构件组装模型。具体说明如下：

1）即插即用软总线技术　CORBA规范提供了即插即用软总线(ORB)技术，利用有关管理工具将有关领域构件/构架、业务构件及热点业务构件注册到ORB总线的接口库中，供访问和调用。

2）基于CCM的构件管理与装配　利用CCM构件模型的有关机制(如抽象模型、构件配置及打包模型及容器模型相互协作)完成构件的连接装配，实现构件的动态集成。

3）问题说明　关于构架的设计及构架与构件的连接是一个复杂的问题，将另有文章专门讨论。

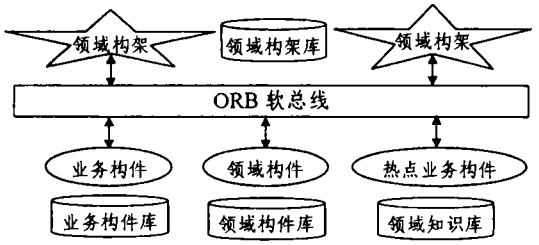

图3　基于软总线的构件组装模型

结束语　RRIS是业界的研究热点，针对可重构实现技术方面存在的不足，本文提出基于业务构件的方法来实现信息系统的快速可重构；将系统的可重构范围界定在功能重构及业务过程重构上，以简化问题的复杂性，并提出有关重构策略；重点研究了BCRRIS的关键技术：构件模型、体系结构、业务构件及框架，限于篇幅有关方面的阐述不能十分详尽，旨在提供某种思路供同行探讨。

参考文献

1　可重构制造系统研究室. http://sy.863cims.net/lab2
2　李群明，张士廉，王成恩，宋国宁. 可重构的企业管理信息系统. http://sy.863cims.net/lab2
3　朱建江，戴勇，王宁生. 基于组件装配的分布式可重构企业信息系统. 南京航空航天大学学报，2001.8
4　沈延森. 快速可重构信息系统及其关键技术研究：[南京航空航天大学博士学位论文]. 2001.5
5　万麟瑞，李绪蓉. 系统集成方法学研究. 计算机学报，1999，22(10)
6　于卫，杨卫东，蔡西尧. 软件体系结构的可视化描述与实现. 计算机科学，1999，26(10)：81～83，31
7　郑明春，王岩冰，等. 一种基于事务构件的ERP系统集成方法. 小型微型计算机系统，2001
8　Aoyama M. New Age of Software Development：How Component-Based Software Engineering Changes the Way of Software Development. 1998 international Workshop on Component-Based Software
9　常煜芬. CORBA构件模型的研究与实现：[南京航空航天大学硕士论文]. 2001.2
10　Herzum P，Sims O. The Business Component Approach. OOPSLA'98 Business Object Component Workshop IV
11　Kozaczynski W. Architecture framework for business components. [C] In：5th Intl. Conf. on Software Reuse，computer socity press，1998. 300～307
12　Bronsard F，Bryan D，Kozaczynski W. Toward Software Plug-and-play. In：Proc. of the Symposium on Software Reusability，Boston，May 1997
13　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)
14　Baster G，Konana P，Scott J E. Business Components：A Case Study of Bankers Trust. Communications of the ACM，2001，44(5)
15　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

(上接第152页)

专业人员使用计算机解决其领域问题成为可能，极大地推动了计算机的普及与发展；面向对象技术的引进，使得类、对象、继承、重用等术语加入到软件语言中，其内涵更丰富，抽象层次更高；软件模式中的设计模式和体系结构模式是高级语言和面向对象语言的使用知识的抽象，是目前内涵最丰富、抽象层次最高的交流语言，如前面所述的23种面向对象的设计模式及多种体系结构模式比高级语言语句和面向对象语言语句包含有更丰富的内容；软件模式中的分析模式是需求分析经验与知识的抽象与总结，为需求分析提供了高层次的交流语言，我们说人类语言极大地推动了人类的文明进步与发展，分析模式为需求分析提供专业术语与语言，将促进需求分析的科学化与规范化；软件模式中的过程模式是软件过程中基本过程步的抽象，为软件过程的分析与改进提供交流语言，是提高软件过程可视性与可控性的基本保障；总之，软件模式是软件生命周期中各项软件活动经验与知识的抽象、概括和总结，使软件活动在更高的层次上进行，将推动软件领域的进步。

参考文献

1　Gamma E，Helm R，Johnson R，Vllissides J. Design Patterns：Elements of Reusable Object-Oriented Software. Addison-Wesley，1995
2　Owler M. Analysis Patterns：Resuable Object Models. Addison-Wesley，1997
3　Haw M. Patterns for Software Architectures. In：Coplien J，Schmidt D. Pattern Languages of Program Design. Addison-Wesley，1995. 453～462
4　Coplien J. A Generative Development-Process Pattern Language. In：Coplien J，Schmidt D. Pattern Languages of Program Design. Addison-Wesley，1995
5　Ambler S. Process Patterns：Building Large-Scale Systems Using Object Technology. Cambridge University Press，1998
6　Coplien J，Vlissides J，Kerth N. Pattern Languages of Program Design-Vol. 2. Addison-Wesley，1995
7　Kotula J. Using Patterns to Create Component Documentation. IEEE Software，1998，15(3)：84～92
8　Keepence B，Mannion M. Using Pattern to Model Variability in Product Families. IEEE Software，1999，16(7)：102～108
9　Aridor Y，Lange D B. Agent Dedign Patterns：Elements of Agent Application Design. In：Proc. of Autonomous Agents '98，Minneapolis，MN，1998. 108～115
10　Florijn G，Meijers M，Van Winsen P. Tool Support for Object-Oriented Patterns. In：Proc. of ECOOP'97，Finland，1997
11　Beck K，et al. Industrial Experience with Design Patterns. In：Proc. of 18th Intl. Conf. on Software Engineering，Berlin，Germany，February 1996. 103～114
12　Schmidt D. Using Design Patterns to Develop Reusable Object-Oriented Communication Software. Communications of the ACM 38，Oct. 1995. 65～74