華夏英才基金學術文庫

王志坚 费玉奎 娄渊清 编著

软件构件技术及其应用

科学出版社
www.sciencep.com

华夏英才基金学术文库

软件构件技术及其应用

王志坚 费玉奎 娄渊清 编著

科学出版社
北京

内 容 简 介

本书主要介绍软件构件技术的有关内容以及相关技术的发展状况，内容包括绪论、构件表示、构件模型、构件库的设计及检索、构件适配技术、构件组装技术、软件复用、模式与框架、网格计算与 Web service、移动 Agent 技术。

本书可供在软件开发领域工作的科研人员及计算机相关专业的研究生参考。

图书在版编目(CIP)数据

软件构件技术及其应用/王志坚，费玉奎，娄渊清编著．—北京：科学出版社，2005

(华夏英才基金学术文库)

ISBN 7-03-015041-4

Ⅰ.软… Ⅱ.①王…②费…③娄… Ⅲ.软件工程 Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2005)第 013654 号

责任编辑：段博原 贾瑞娜 / 责任校对：张 琪

责任印制：钱玉芬 / 封面设计：陈 敬

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2005 年 4 月第 一 版 开本：B5(720×1000)

2005 年 4 月第一次印刷 印张：16 1/4

印数：1—2 500 字数：311 000

定价：35.00 元

(如有印装质量问题，我社负责调换〈环伟〉)

前　言

软件复用是提高软件生产效率和质量的有效途径。软件复用是指重复使用“为了复用目的而设计的软件”的过程。依据复用的对象，可以将软件复用分为产品复用和过程复用。产品复用指复用已有的软件构件，通过构件组装得到新系统；过程复用指复用已有的软件开发过程，使用可复用的应用生成器来自动或半自动地生成所需系统的过程。过程复用依赖软件自动化技术的发展，目前只适用于一些特殊的应用领域，产品复用是目前现实的、主流的途径。

软件构件技术是支持软件复用的核心技术，其发展迅速并受到高度重视。构件技术的不断发展与成熟为软件开发提供了新的方法学，学术界和产业界都围绕构件技术展开了大量的研究，并使这一技术在一些领域得到了有效的应用。软件构件技术研究的不断深入，必将导致软件生产方式的变革，从而将极大地提高软件生产的效率和质量。

目前，软件构件还没有一个被广泛接受的定义。一般认为，构件是一个可以独立交付的软件单位，具有相对独立的功能和复用价值。随着对软件复用理解的深入，构件的概念已不再局限于源代码构件，而扩充到软件开发过程中各个阶段的产品，如需求、需求规约、构架、文档、测试计划、测试用例和数据等，所有这些都被称为可复用的软件构件。

软件构件技术的主要研究内容包括构件生产与获取、构件模型、构件描述语言、构件分类与检索、构件组装和标准化等，本书对此作了详尽的论述。全书共分十章，全面系统地论述了软件构件技术的主要研究内容，对软件构件技术的相关研究内容进行了分析介绍。第一章是本书的概述，主要对软件构件的定义和特征、产生和演变、现状、目的和用途进行阐述，详细介绍了基于构件的软件工程方面的有关内容；第二章至第六章论述软件构件技术主要的研究内容，包括软件构件的描述方法、构件模型、软件构件库的构造机制、软件构件的检索与适配、软件构件的组装和机制；第七章到第十章主要论述与软件构件密切相关的技术，包括软件复用、设计模式与框架、网格计算与 Web service 以及移动 Agent 技术。

本书一方面是对作者长期研究成果的总结，同时，也是为推动软件构件技术的进一步发展所做的尝试，目的是力求为软件构件研究者、软件生产企业和高校研究生提供一本能反映软件构件技术全貌的书籍。由于作者水平所限，书中难免存在错误与不足之处，恳请读者批评指正。

编　者

2004 年 3 月于河海大学

目　录

第一章 绪 论

软件复用是在软件开发中避免重复劳动的解决方案。通过软件复用，可以提高软件开发的效率和质量。近年来，面向对象技术出现并逐步成为主流技术，为软件复用提供了基本的技术支持。软件复用研究重新成为热点，被视为解决软件危机、提高软件生产效率和质量的现实可行的途径。软件复用通常可分为产品复用和过程复用两条途径。基于构件的复用是产品复用的主要形式，也是当前复用研究的焦点。基于构件的软件开发方法(CBD)逐渐被人们接受，并用于软件开发实践中，特别是基于分布式对象的开发领域。CBD 集软件重用、分布式对象计算、CASE 和企业级应用程序开发等技术为一体，以软件构架为组装蓝图，以可重用软件构件为组装块，支持组装式软件重用，是提高软件生产效率和产品质量、缩短产品交付时间的现实有效的途径之一。本章主要对软件构件的定义和特征、产生和演变、现状、目的和用途进行阐述，最后叙述分布式对象技术及软件构件技术与分布式对象技术的关系，使读者对软件构件技术有一个整体的认识。

第一节 软件构件技术及其演变

一、软件构件的认知过程

计算机软件(简称软件，software)是计算机系统中的程序及其相关文档。程序是计算任务的处理对象与处理规则的描述；文档是为了便于理解程序所需的资料说明。

软件充满了其创造者的个性化特征。随着软件系统的大型化与复杂化，要求软件生产必须标准化、规模化和节约化，必须从个人和团体的手工作坊式的生产模式向社会化的专业分工协作生产模式转化，即实现软件生产的工业化。但是，目前软件生产的工业化程度还比较低，还不能对好的程序进行持续不断的复用。为此，业界对软件复用的技术进行了几十年的研究，并逐渐形成了利用软件构件来实现软件复用的共识。

软件构件(简称构件，component)的概念共生于软件复用。早在 1968 年，在北大西洋公约组织(NATO)软件工程会议上就提出了软件复用的概念，后来还为此制定了一整套软件复用的指导性标准，其中包含了利用标准构件实现软件复用的基本思路。也就是在这次会议上，McIlroy 在题为“*Mass-produced Software*

Components"的报告中提出了软件构件、构件工厂等概念。

生产标准软件零部件，从而组装成软件的设想一产生就受到了人们广泛关注。但在不同历史时期人们对它的认识却不尽相同。

在20世纪70～80年代，软件构件主要指可复用的程序代码片段，一般被称为代码件。这一时期软件生产考虑的主要问题是如何充分利用已有的源程序代码、子程序库和类库来提高软件开发的效率。此时的代码件主要有子程序、程序包、类、模板等形态。

到20世纪90年代，软件构件应当包括分析件、设计件、代码件、测试件等多种类型的观念被广泛认同。由于软件复用的多样性，又将其分为产品复用和过程复用。随之产生了许多新的概念，如设计模式、框架以及软件体系结构等。但是，对软件构件基本属性的深入探讨却主要围绕代码件展开，若不特别注明，本文所论及的软件构件通常是指代码件。

1995年，Will Tracz提出构件应具有以下属性：有用性(usefulness)，构件必须提供有用的功能；可用性(usability)，构件必须易于理解和使用；质量(quality)，构件及其变形必须能正确工作；适应性(adaptability)，构件应该易于通过参数化等方式在不同语境中进行配置；可移植性(portability)，构件应能在不同硬件运行平台和软件环境中工作。归纳起来就是构件应具有柔性。

近年来，随着分布式对象、Internet、Java、Client/Server等技术以及基于构件的软件开发技术的发展，对构件的认识又产生了新的变化，出现了若干新的软件构件的定义，其中比较有代表性的有：

"软件构件是一个具有规范接口和确定的上下文依赖的组装单元。软件构件能够被独立部署和被第三方组装" —— 欧洲面向对象编程(ECOOP)会议(1996)。

"软件构件是可单独生产、获取、部署的二进制单元，它们之间可以互相作用构成一个功能系统(functioning system)" —— Szyperski(1998)。

"软件构件是一个不透明(opaque)的功能实现；能被第三方组装；符合一个构件模型" —— Bachman等，卡内基·梅隆大学软件工程研究所(CMU/SEI)(2000)。

"构件是一个带有契约化接口和显式上下文依赖的组装单元，它能被独立发布并且可以被第三方组装" —— Guijun Wang等，美国波音公司(1999)。

"软件构件是一个可以独立交付的软件单元，封装了设计和实现的内容，并向外提供接口，通过接口与其他构件组装成更大的整体" —— Desmond等(1998)。

"构件代表一个自包含的实体，能够向其环境输出功能并可通过定义明确的开放接口从环境输入功能" —— Michael(1998)。

"构件由一个动态变化的对象集合组成，这些对象既可以在构件的内部也可能是其接口的一部分。构件之间可以直接交互，也可以通过独立的对象进行胶

合”——Franz等(1997)。

“构件是一个通过接口向外界提供服务的软件包”—— Microsoft Corporation(1997)。

上述定义的共同要素是:软件构件是单独开发并具有特定功能的软件单位,用于与其他构件及支撑环境组装成应用系统。这一共同要素反映了构件的3个基本特征:①单元特征——构件不是完整的应用程序,需要组装。②复用特征——构件的价值在于实现软件复用,需要规范。③商品特征——构件是预制的知识服务,需要封装。

上述定义的共同要素说明业界对构件技术的认知正逐渐趋于一致。

软件构件的概念从提出到在业界形成一定的共识,经历了几十年的演化。与计算机硬件技术的发展相比,进展较慢。计算机硬件虽然十分复杂,但建立在技术复用思想基础之上的开放性、标准化技术体系,实现了技术的不断有效积累和开放竞争的市场,进而形成了以高效率和低成本为基本特征的现代化成熟生产工业。而计算机软件的生产却面临着高失败率、很少能按时交付、经常超过预算成本等困难,形成了软件生产高投入、高风险却不一定能高产出的行业特征。软件资源的多元化,应用环境的复杂性和多变性是软件危机的本质。如何借鉴硬件技术发展的成功经验,是软件构件技术正在研究的问题。

二、构件技术的发展

基于构件的开发方法与软件复用有密切的联系。最早成功复用的实例是软件库,如数学库。它由许多函数(如三角函数等)的原代码及连接方式组合而成,这种类型的复用实体其优点在于:

(1) 有一个良好的有关函数类型的定义理论。

(2) 应用程序与函数之间的通信比较简单,应用程序调用函数时只需传入输入参数,通过执行相应的功能,库响应请求,返回输出参数。

(3) 输入和输出都是精确定义的。

(4) 有良好的容错机制,当输入参数有错时,输出就会返回一个特定的数值表示出错。

(5) 这种构件类型的缺陷是它不够灵活。对于一个新版本的库,对应的应用程序需要重新创建。这个问题可通过引入动态链接库来部分解决。另外不灵活的地方是对输入输出参数的限制。当参数类型改变时,必须使用新的库。

另一种复用实体是客户机与服务器分离。应用程序(客户机)发出请求到服务器,服务器响应请求,执行相应的功能并把结果发送回客户端。典型的应用实例是数据库或图形接口(GUI)。要使得服务器能够复用,需要定义一个API(application

programmable interface)形式的标准通信协议。对于关系型数据库,不同的数据库提供者使用相同的标准——SQL 语言。这样就可以实现同一个应用程序使用不同的数据库而不需要重写代码。

随着 Internet 的出现以及不同操作系统的建立,分布式技术变得尤为重要,对应用程序、操作系统兼容性的要求也变得更加严格。另外,人们对系统构件的可替换性的要求也越来越高。这些需求导致了一个软件开发的新范型——基于构件的软件开发方法的出现,基于构件的软件工程(component based software engineering,CBSE)走入了人们的视线。

第二节　基于构件的软件工程

随着计算机技术的飞速发展,各行各业对软件开发的速度和质量要求都有了很大提高。然而,传统的"手工作坊"式的软件开发状况并没有得到根本改变,软件技术的进步仍远远落后于硬件技术的进步。值得庆幸的是,20 世纪 90 年代发展起来的"基于构件的软件工程"有望从根本上解决这一问题,从而成为软件工程中的又一里程碑。

CBSE 是指用装配可重用软件构件的方法来构造应用程序。它包含了系统分析、构造、维护和扩展的各个方面,在这些方面中都是以构件方法为核心的。CBSE 的主要目标是通过集成构件来开发系统、开发可复用的构件及通过定制和替换构件为维护和更新系统等提供支持。它的基本宗旨是软件系统可以通过选择构件,在一个定义良好的软件构架内进行构件组装以生成系统。它不同于传统的方法,如系统开发一切从零开始。商业构件(commercial off-the-shelf (COTS) component)可以由不同的开发者使用不同的语言在不同的平台下进行开发。商业构件可以从构件库中选取、检验,经过组装、部署以形成目标系统。

用构件来构建系统,为不同的系统需求开发构件所需要建立的方法及过程不仅涉及开发维护方面,而且与整个构件和系统的开发周期相关,特别是构件特有的方面,如构件规约、组装等。许多软件工程的原理都需要为基于构件的应用制定特别的方法学。大多数方法学还不可用,甚至还没有建立。近期软件开发方面的进步将依赖于有关 CBSE 方法学的建立,这已成为工业及学术界的共识。

一、CBSE 的特点

与传统的软件重用方法比较,CBSE 有以下特点。

(1) 即插即用。构件可以方便地集成于框架中,不用修改代码,也不用重新编译。

(2) 以接口为核心。构件的接口和实现是分离的。构件通过接口实现与其他构件的交互与组装,构件的具体实现被封装在内部,组装者只关心接口,不必知道其实现细节。

(3) 标准化。构件的接口必须严格地标准化,这是构件技术成熟的标志之一。目前,主要的构件标准有 Microsoft 的 COM/DCOM,SUN 的 EJB,OMG 组织的 CORBA。可以说,计算机界很久以前就有用构件来装配成应用软件的想法,但始终未能成为现实,其中的主要原因是构件标准的缺乏。正是由于出现了以上较为成熟的构件标准,才使得 CBSE 由梦想走向现实。

(4) 构件通过市场销售和分发。大量成熟的构件可以通过市场购得,市场的竞争机制也可以保证构件生产质量的提高、种类的增加和价格的降低。

二、CBSE 的意义

1. CBSE 从根本上改变了软件生产方式

正是福特创造了汽车的流水线制造法,才开创了工业化规范大生产的新纪元。福特制造的精髓就是将汽车生产的重点从制造每一个零件转到装配,汽车制造者不必自己设计制造每一个零件,大部分零件由外购而来。过去的软件生产方式与旧的汽车生产方式十分相似,开发者往往要编写程序中的绝大多数代码。因此,如果能实现像组装汽车或机器一样地进行软件开发,将是软件工程的巨大进步。

2. CBSE 提高了软件重用率,保护了已有的投资

生产好的构件可以分发销售给多个其他用户,一方面大大降低了单个构件的成本;另一方面也大大降低了软件开发中的重复劳动。目前,在各家企事业单位中存在着许多旧的计算机软件系统,可以将这些系统分成模块后通过构件技术封装起来,成为新系统的组成部分。这种通过标准的接口将旧的程序代码隐藏起来的做法,巧妙地保护了已有的软件投资。

3. CBSE 使开发者将更多的注意力放到业务流程和业务规则上去

由于开发者的主要工作是构造框架和装配构件,使他们可以摆脱编程的细节问题,将更多的精力投入到与用户交流。另外,业务管理者也可以在更高的层次上,用偏近于业务而不是偏近于计算机的语言进行讨论。

4. CBSE 开发的系统维护十分方便

由于 CBSE 是模块化开发,如果某个模块需要修改,只需用修改好的模块替换

掉以前的模块，不用重新编译整个系统。若想扩展系统的功能，也只需将符合框架约束条件接口要求的扩展模块直接加入到该系统即可。由此可见，CBSE 开发的系统维护和升级都十分方便。

5. CBSE 降低了对系统开发者的要求

尽管 CBSE 没有消除系统开发者和使用者之间的分界线，但却移动了这条分界线。这是因为 CBSE 的开发者主要任务是装配已有模块，不需要有很高的编程技巧，从而使更多的人可以构造适用于自己的系统。在开发环境中，仅仅在构造构件时才需要对编程语言的熟悉和高超的应用技巧。

三、CBSE 与 OO 技术的联系和区别

CBSE 与 OO 技术有着密切的关系，因此往往有人将两者混淆起来。实际上 OO 技术对于 CBSE 既不是必要条件，也不是充分条件。

首先，构件不一定要用 OO 语言编写，任何一种可以实现构件标准接口和所需功能的语言都可以用来编写构件。由于 OO 语言的种种特点，一般认为它是编写构件的最自然的语言。但国外有一些专家反而认为目前使用的 OO 技术，如Java、CORBA 和 ActiveX 不稳定且并非足够成熟，不是最佳选择。

其次，CBSE 扬弃了 OO 技术的某些特点。多态性和继承性是 OO 技术的重要特点。但是，对象之间的继承性可能造成系统间的级联影响，即父类的改动会引起相应的子类的性质的自动变化。这些性质对于 CBSE 显然是不利的。CBSE 更重视的是封装性，它希望程序修改导致的影响严格限制在构件的内部。

另外，CBSE 的涵盖面要广于 OO 技术。CBSE 包括了系统分析、系统的设计与建模、项目组织、构件的开发管理等各个方面，而 OO 技术通常只包含 OOA、OOD、OOP 这 3 个方面。

四、CBSE 的生命周期

基于构件的软件系统是通过组装构件而形成应用系统，不是从头开始编程。因此，它的生命周期不同于传统的软件系统。其生命周期可总结如下：①构件需求分析。②构件开发。③构件验证。④构件定制。⑤系统集成。⑥系统测试。⑦系统维护。

1. 构件需求分析

构件需求分析是发现和理解文档、确认和管理构件需求的过程。构件需求分析

的目标是得到完全的、一致的和与构件应实现的功能相关的需求。

需求分析包括4个步骤:需求收集和定义、需求分析、构件模型和需求确认。该阶段的输出是用户需求文档。

2. 构件开发

构件开发是通过生成具有良好功能、高质量的构件实现需求的过程。其目标是最终的构件产品、接口和文档。构件产品能够正确地实现需求,应有良好的行为和灵活的接口。构件开发过程包括4部分:实现、功能测试、可靠性测试和文档。

3. 构件验证

构件验证包括:①构件外购。②构件选取——根据需求,兼顾功能和可靠性选择合适的构件。③构件测试——验证构件能否满足需求。

4. 构件定制

构件定制包括:①为特定需求更改构件。②为了能够在特定平台下运行对构件进行必要的改变。③更新构件以得到更好的性能和更高的质量。构件定制的目标是对开发的构件做必要的修改使其能使用于特定的环境或者与其他构件更好地合作。

5. 系统集成

系统集成是在系统体系结构下,集成已选取的构件形成整个系统。其目标是一个由构件组装而成的最终系统。

6. 系统测试

系统测试是评估系统能否满足规定要求,确定和纠正系统实现过程的缺陷的过程。其目标是一个满足系统需求的由构件组装集成的最终系统。

7. 系统维护

系统维护是系统交付以后所提供的对系统维护的过程。其目标是通过纠正错误、改善性能及适应环境的改变为最终用户提供一个高效的产品和服务。

五、CBSE的主要设计原理

(1) 面向领域。软件复用是增加生产、减少维护费用及开发费用的最有效的方法。为了达到复用的成功,相关系统的共同性应发现并以一定的形式表现出来以便

在开发类似系统时能够被利用起来。面向领域是一种方法，它力图发现系统在应用或技术领域的共同性，进而开发模块或构件并用于该领域的开发活动中。虽然面向领域的方法被认为是取得 CBSE 成功的关键因素，但是，大多数面向领域的工程技术还处于幼年期，还有很多问题需要解决。

(2) 分离原则(separation of concern)。构件应设计得具有单一性，任何一个功能构件都应有独立的接口机制与其他构件通信。在选择特定的功能构件或接口方法/机制时不能对选择的其他构件强加限制。

(3) 抽象虚拟机接口(abstract virtual machine interface)。构件的接口必须作为虚拟机来设计。一个构件必须提供完整的无冗余的接口。接口应隐藏构件的内部结构以便产生构件的不同实现。

(4) 上下文绑定延迟(postponement of context binding)。在设计构件时，有必要争取做到开发上下文无关的构件，精力集中在构件的核心功能上。在一定程度上，绑定特定的上下文有关的参数(如数据类型)是有可能的，存储长度、实现算法、通信方法、操作环境等应当延缓到构件集成时期，性能优化已经做完之后。

(5) 设计复用(design reuse)。对于基于构件的软件集成而言，设计(构件开发上下文)应该在特定用户中共享和复用，即在构件复用之前进行设计复用。结构设计给出了构件的功能分配，对于构件集成，复用基于构件开发的结构是必要的。

(6) 层次结构(hierarchically layered architecture)。结构和代码构件应设计得在组装构件时有尽量大的柔性。图 1-1 包括一个层次结构模型。它将面向应用的构件(用于控制和活动协同)和功能构件(主要是计算)分开。在领域中通常使用的实现技术也与功能构件相分离。

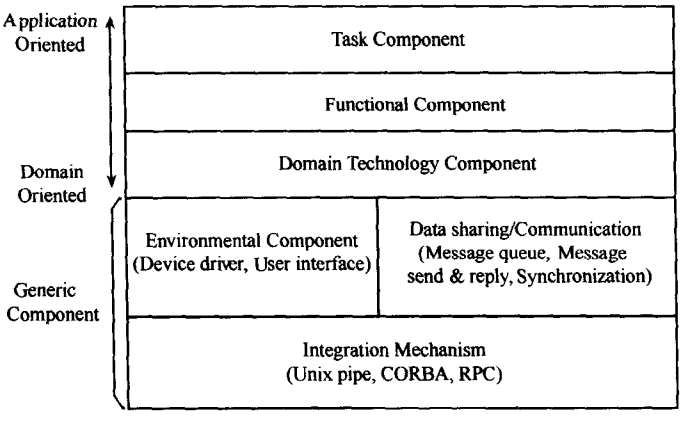

图 1-1　构件的层次结构模型