现代软件工程专业系列教材

软件构件技术

RUANJIAN GOUJIAN JISHU

夏榆滨　主　编

王　玲　庞培宇　孟凡宾　副主编

清华大学出版社
http://www.tup.tsinghua.edu.cn
北京交通大学出版社
http://press.bjtu.edu.cn

现代软件工程专业系列教材

软件构件技术

夏榆滨　主　编
王　玲　庞培宇　孟凡宾　副主编

清华大学出版社
北京交通大学出版社
·北京·

内 容 简 介

本书较全面地介绍了软件构件技术的产生、发展、构件化思想、基于构件的软件过程及其他相关技术。书中还结合作者的相关研究成果，给出了软件构件视图的相关概念，较详细地介绍了基于软件构件视图技术的构件组装管理技术及相关软件平台实例的设计与实现方法，并对关键实现代码进行了说明。随书光盘附有较完整的源代码、编译后的相关安装程序及使用说明。本书每一章的前面都有关键问题一节，旨在引导读者对关键概念进行预先的思考，增加对随后内容的兴趣，便于深入理解有关概念，掌握本章的重点。

本书可作为软件工程专业本科生和研究生的教材，也非常适合于软件技术领域工作的工程技术人员作参考。

图书在版编目（CIP）数据

软件构件技术 / 夏榆滨主编. — 北京：清华大学出版社；北京交通大学出版社，2010.11
ISBN 978－7－5121－0384－9

Ⅰ. ① 软…　Ⅱ. ① 夏…　Ⅲ. ① 软件工程　Ⅳ. ① TP311.5

中国版本图书馆 CIP 数据核字（2010）第 208127 号

责任编辑：刘　洵
出版发行：清 华 大 学 出 版 社　邮编：100084　电话：010－62776969　http://www.tup.com.cn
　　北京交通大学出版社　邮编：100044　电话：010－51686414　http://press.bjtu.edu.cn
印 刷 者：北京东光印刷厂
经　　销：全国新华书店
开　　本：185×260　印张：18.25　字数：456 千字
版　　次：2011 年 1 月第 1 版　2011 年 1 月第 1 次印刷
书　　号：ISBN 978－7－5121－0384－9/TP·625
印　　数：1 ～ 3 000 册　定价：33.00 元（含光盘）

前　言

软件构件技术是当前解决软件问题的最有效和最前沿的技术，目前，图书市场上关于软件构件技术的书籍还不多，国内学者原创的相关书籍也很少见，适合于教学工作的更是寥寥无几。近几年，软件构件技术有了显著的进展，很多已出版书籍的内容已略显陈旧，跟不上科研、实践和教学的需要。本书作者在软件构件技术和相关领域从事了多年的教学和科研工作，为本书的编写做了充分的前期准备。本书力图将软件构件技术理论与应用结合起来，着眼于教学，但又兼顾为所有参与或对软件构件技术感兴趣的在职人员提供帮助。

与同类书籍相比，本书有以下几个特点。

① 在各章节前明显地提出本章节的关键问题，使读者带着问题，学习与思考相关的基本概念和基本原理。

② 既注重基本概念、基本原理，又注重创新思想的培养。教材在保证软件构件技术知识体系完整性的前提下，在各章节结尾处以适当的篇幅专门讨论软件构件技术的本质或核心理念，引导读者进行开放性和创新性的思考。

③ 注重实践能力的培养。结合具体实例，有的放矢地介绍软件构件知识体系的应用思路和具体技术。在习题和应用实例部分，本书以实际企业项目或具有明确的企业背景的科研课题内容为例，进行讲解和指导读者进行练习。

本书共分为七部分。

① 绪论。绪论从软件开发面临的挑战和软件开发的演进入手，引出软件构件技术和基于构件的开发方法。通过对绪论的学习，读者将对软件构件技术的兴起有所了解，并认识到软件构件技术的重要地位和其流行的必然性。

② 构件的基本概念。本章对软件构件技术的基础知识进行介绍，使读者对软件构件技术的基本概念有清晰的认识，为以后进一步应用软件构件技术打下坚实的基础。

③ 构件管理和组装技术。本章通过对构件库的设计与管理、构件组装技术的介绍，使读者了解怎样利用现有构件组装成符合用户要求的应用程序系统。

④ 基于构件的软件过程。本章通过对基于构件软件过程的讲解，使读者了解怎样将基于软件构件的开发思想应用到软件开发过程中。

⑤ 构件组装平台的设计与实现。本章介绍构件组装平台设计与实现中的关键问题和系统要求分析。

⑥ 构件组装平台开发实例。本章通过具体实例，使读者了解软件构件技术在信息系统建设中所扮演的角色及怎样应用构件技术构建信息系统。

⑦ 其他相关技术。本篇通过对软件构件相关技术的介绍，使读者获得一个较为系统的软

件构件知识框架，全面、深刻地掌握软件构件知识体系，从而更好地运用所学知识进行灵活创新。

本书的讲授建议安排 58 个课时，其中课堂授课为 44 个课时，余下 14 个课时作为实践课。

本书中如果不做特殊说明，所有的构件均指软件构件。

本书在编写过程中参考了大量文献，在此向相关作者表示感谢！

编　者

2011 年 1 月

目　录

第1章 绪 论

1.1 软件开发面临的挑战

大规模集成电路集成度的提高，大幅度提高了计算机的计算能力，软件运行的性能瓶颈被不断突破，从而使得软件系统可以实现的功能更加复杂且程序量更加庞大。如何提高大型复杂软件系统的开发效率和开发质量，一直是本领域的一个重要研究方向。从一定意义上讲，软件系统复杂程度的提高，是软件开发方法不断演进的主要动力。

同时，随着 Internet、中间件、电子商务和 XML 等新技术的引入，不仅影响了软件开发的方式，而且也影响了最终用户对软件系统的期望。相关技术的高速发展增加了具有较长生命周期项目的风险。而且，计算机软件所面临的环境开始从静态封闭走向开发、动态和多变，大规模分布式系统的开发越发复杂，软件系统为了适应这样一种发展趋势，将面临更多的需求和挑战。

例如，计算机软件的复杂性持续增长，既体现在系统的规模上，也体现在了快速演变的功能需求和非功能需求上。这使得软件开发与维护技术变得复杂，开发的软件质量难以保证，可靠性低，软件成本难以控制，极少有在预定的成本预算内完成的，软件开发进度难以控制，软件提供者难以较好地满足应用的需求，维护人员和费用不断增加。

另外，IT 行业的雇佣状况在动态变化着。随着当前企业之间人才竞争的白热化，软件开发企业面临着软件工程师频繁流动这个问题。这意味着一个员工在一个企业就职的短短时间内，要学习和消化不断出现的新技术，而将所学到的理论投入实践并真正掌握需要很长一段时间。我们在不断增强能力和技术可能性，却没有给我们的软件工程师足够时间充分消化他们需要知道的东西。可见，雇佣情况的变化给企业软件开发工作带来了更高的要求。

显然，当软件系统达到一定的规模和复杂程度以后，软件所面临的挑战非常严峻。要设计一个系统，使它可以在操作环境改变、用户需求改变和错误暴露出来的时候很容易地演化，将是一个非常艰巨的任务。

1.2 软件开发方法的螺旋式演进——从结构化到构件化

随着应用程序复杂程度的不断提高，软件开发方法也在不断进步。其中一些具有里程碑意义的进展。

1.2.1 结构化开发方法

20 世纪 60 年代末至 70 年代中期，在一系列高级语言应用的基础上，出现了结构化分析和开发手段。这种方法着眼的是系统的功能，要点是系统的功能分解，即将系统的功能作为划分模块的标准，程序设计从高层模块开始就与当前系统的需求紧密结合，并且使数据结构的设计服从于功能实现的要求，因此传统开发方法开发出来的软件能够很好地满足当前对系统的功能要求，但也为以后的扩充和重用设置了障碍。

1.2.2 面向对象开发方法

随着信息系统的加速发展，应用程序日趋复杂化，结构化开发方法难以满足发展的新需求。20 世纪 80 年代中期至 90 年代，出现了面向对象开发方法。面向对象方法提供了强有力的抽象机制，它使用类和对象作为分析和实现的主要结构。面向对象方法使功能设计人员和开发人员能够把注意力集中到相当稳定的单元上，即系统的类和对象，这些单元与业务概念匹配，可以直接在信息系统中表示。

面向对象方法实现了“开发时”重用，即与以前的方法相比，面向对象方法使开发人员构建软件时，能够重用其他开发人员提供的设计和代码片段，这虽然为人们提供了便利，但它始终关注的是单一系统的开发，单凭这种方法不能解决跨平台的可移植性问题，而且，部分地采用面向对象技术的可能性微乎其微。

出现这种问题的关键原因在于：面向对象方法改变了构建应用程序的方式，但没有改变应用程序本身的性质。例如，无论在面向对象方法出现之前还是之后，最终用户都要接受整块应用程序，只不过开发语言发生了变化，但这对于最终用户来讲并没有什么真正的区别，即面向对象开发方法是服务于开发人员的，而不是最终用户。面向对象方法只解决了整个软件开发过程中很重要但是有限的问题，即设计和开发问题，但它根本没有解决部署问题，更没有解决即插即用问题。

1.2.3 分布式对象方法

随着信息技术继续向前发展，人们很快发现面向对象开发方法难以解决可移植性、互操作性、分布式应用等问题。20 世纪 90 年代，软件行业开始引入分布式对象。分布式对象方法扩展了面向对象方法，使其能够跨越地址空间边界调用对象，一般是利用对象请求代理，如 CORBA 对象请求代理。分布式对象方法的价值在于解决孤立分布式系统的互操作问题，屏蔽了操作系统和网络协议的差异，为应用层软件提供了多种通信机制和统一的应用平台，简化了软件开发的复杂度，提高了开发的效率，但是这往往本质上是以面向对象方法的开发思路为基础的，如何降低开发成本的问题依然存在。

1.2.4 基于构件的开发方法

软件复用是在软件开发过程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质量和效率。面向对象技术的产生和发展，为软件复用提供了基本的技术支持。软件构件（Component）的概念共生于软件复用。1968 年在北大西洋公约组织软件工程会议上提出了软件复用的概念，同时制定了一整套软件复用的标准，其中包含了利用标准软件构件实

现软件复用的基本思路。

近年来，随着软件构件技术的发展，基于构件的软件开发成了软件业关注的焦点。基于构件开发的基本思想是，创建和利用可复用的软件构件来解决应用软件的开发问题，只要遵循构件技术的规范，各个软件开发商就可以用自己熟悉的语言去实现构件，应用程序的开发人员就可以挑选构件组合新的应用软件，这样的应用软件系统不再是一种固化的整体系统，而是通过构件间相互提出请求和返回服务结果的协同工作机制来达到系统目标。这是一种从头到尾使用构件概念的系统开发方法，即通过构件思维统一整个开发生命周期。这种新型软件开发方法使得软件开发由“从零开始”的传统模式转变为基于已有构件的组装过程，消除了包括分析、设计、编码、测试在内的许多重复劳动，从而降低开发费用、提高生产率及以最小的代价实现软件的随需而变和最大限度的成果共享，同时，通过复用高质量的已有开发成果，避免了重新开发可能引入的错误，从而提高了软件的质量。

1.3 为什么要应用软件构件技术

在计算机应用越来越普及的今天，软件的生产远远满足不了用户急剧增长的需求。考虑到软件复杂程度的不断提高，可以说软件危机还在进一步加剧，软件制造还没有从根本上解决手工编程和调试的低效方法。基于传统开发方法实现的软件系统在可用性、可靠性、性能、安全及其他服务质量方面存在一系列的问题。

传统制造业一般是基于大规模批量生产模式的，通过零部件的标准化和生产的流水化，可以大幅度提高生产效率，降低生产成本，提高产品的可维护性。

对比传统制造业，目前的软件生产效率仍然很低，还未真正形成规模化的软件产业。软件构件技术的产生，是基于构件的开发方法的基础，为软件的规模化生产创造了条件。那么，基于软件构件的开发到底有什么好处呢？下面作一个概要的归纳，详细说明见本书后面相应章节。

- 开发工作构建在已有成果的基础上。基于构件的开发并不是摒弃已有的好的原则和技术，而是构建在过去所有最成功的技术、原则和最佳实践的基础上，不仅包含面向对象和分布式对象方法的优点，而且还突破了这些方法的局限性。
- 可以控制开发复杂性。采用面向软件构件的开发思路来组织整个开发过程，是降低开发阶段复杂性和成本的有利机制。
- 可以控制软件系统部署复杂性。对于采用传统方法构建的系统，通常一个小的修改会影响多个模块，在很多情况下，整个应用程序都需要重新部署。而基于软件构件的系统，修改通常只影响所需的构件，只需重新部署那个受影响的构件。
- 简化整个软件需求和开发周期内的工作。基于构件的开发，不仅涉及开发阶段，还涵盖需求、定制、部署和维护的方法。
- 便于系统升级。构件是按规范开发出来的，能被较容易地替换为更便宜、有更多附加值、使用更好技术并且功能更强的构件，这样提高了升级系统的灵活性，系统开发者也不必过分依赖于某个构件供应商。
- 较好地利用本组织的最佳方法。构件的开发者可以使用最适合的开发工具而不必学习新的语言和工具，这样就允许开发组织运用已有的最佳技术。
- 降低开发费用。由于构件的供应商开发及维护构件是针对多个用户的，那么使用该构

件的软件的开发及维护费用被多个用户分担。

● 缩短产品投放市场所需的时间。基于构件的开发很容易通过现有构件组装新的系统，大大缩短了软件系统的发布时间。

软件构件技术和基于构件的开发方法是软件工厂得以实现的重要理论基础和技术保证。

软件工厂是指一种大规模生产软件的组织和方法。它使得软件生产条理化、系统化；软件“元器件”可以进行批量生产；项目实施人员可以对软件“元器件”进行自由组合；软件工厂可以最大限度地利用已有资源，对已有“元器件”进行装配，使得软件“元器件”的复用性得到提高，软件开发工作量明显减少，软件成本大幅度降低；当某个软件“元器件”出现问题时，可以很容易地用新版本替换存在问题的软件“元器件”，并且有可能是来自另一个软件提供商的新版本。利用软件工厂进行软件开发，项目实施人员可以控制项目周期、成本、质量，开发速度得到提高，项目效益和成功性得到保障。

由此，软件工厂有两个要素：一是软件“元器件”技术，二是软件“元器件”的组装、链接、合成技术。每个软件“元器件”都应该有具体的特性，满足具体的用户需求，具有标准的接口和详细的规格说明，然后按照预先确定的解决方案，将这些软件“元器件”装配成用户需要的应用程序。构件和基于构件的开发恰恰满足了这种需求。构件就是这些软件“元器件”，而基于构件的开发方法，以软件构架为组装蓝图，以可复用软件构件作为组装预制“元器件”，把来自不同厂商的构件组装成实现特定功能的模块，从而提高软件生产效率和生产质量，减轻人员流动副作用，缩短产品的交付时间。

曾有人提出一种基于构件的软件工厂模式，如图 1-1 所示。

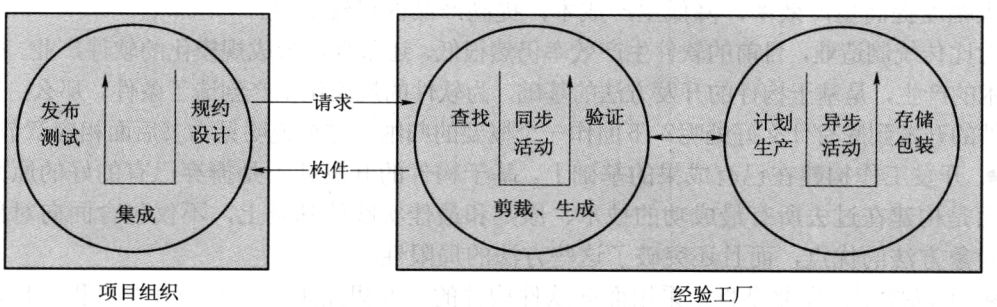

图 1-1　基于构件的软件工厂模式

在这种模型中，构件生产组和系统开发组间严格按照生产者——消费者关系进行任务分工：经验工厂负责生产、提供构件，项目组无需编程，而是从经验工厂中请求所需的构件，再将其组装成最终所需的系统。经验工厂的活动分为同步活动和异步活动。同步活动指配合项目组的活动，为项目组服务。异步活动指有目的的构件生产或对同步活动中的构件进行再加工以提高构件的可复用性。

讨论与思考

1. 软件开发面临的挑战有哪些？为应对这些挑战，有哪些相关技术？
2. 简述软件构件技术的作用。

第2章

构件的基本概念

2.1 关键问题的提出

构件与面向对象技术中的对象概念之间有何关系？对这个问题的正确理解，有助于初学者深入掌握软件构件概念的本质，更加灵活地理解构件化的思想。

2.2 构件的概念

从构件技术提出至今，出现了若干软件构件的定义，但到目前为止，软件构件还没有一个统一的定义，其中有代表性的定义包括以下几种。

① 1996 年 ECOOP 会议上提出的定义：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。

② Szyperski 在 1998 年给出的定义：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。

③ CMU/SEI 的 Felix Bachman 等人在 2000 年 5 月的一份关于基于构件的软件工程的报告中给出的定义：构件是一个不透明的功能实现，能够被第三方组装，符合一个构件模型。

④ OMG 的定义：构件是一个物理的、可替换的系统组成部分，它包装了实现体且提供了对一组接口的实现方法。

这些定义说明了软件构件是单独开发并且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，也反映了构件的三个基本特征：单元特征，即构件不是完整的应用程序，需要组装；复用特征，即构件的特征在于实现软件复用，需要规范；商品特征，即构件是预制的知识服务，需要封装。

⑤《大规模基于构件的软件开发》给出的定义：构件是一个包含约定的被调用接口，可以被独立部署，且通常依赖于组装运行环境的结构单元。(这是面向软件范畴的狭义构件定义，本书将不讨论软件工程范畴的广义构件定义。)

下面解释一下此定义所包含的意思。

所谓构件，是指可以被独立部署的结构单元，表示构件是基于某种结构与运行模型的，独立于具体应用系统的，可以被另外发布或交付的功能单元，是系统的一个物理的、可单独

替换或升级的部分，是对一系列软件操作或实现的包装，这种包装可以用来构造应用程序或更大的构件。

构件就像是软件系统的一个可重用片段，它可以与其他片段组装在一起形成各种更大的片段，甚至是整个的应用解决方案。扩展开来，从系统开发的角度讲，开发人员可以有意识地以构件为基本实现目标，进行所谓的基于构件的需求分析、设计和系统实现。

构件依赖于组装运行环境，是指一个构件按照其模型要求，常依赖于其他构件和软件系统提供的组装或运行支持服务。对于异构构件来说，还需要标准互操作通信平台的支持。一般来说，单个构件只有通过与其他构件协同或安装在一起，才能实现构件的有用性。

构件包含约定的被调用接口，表示构件是服务的提供者，它具有运行时可以被访问的接口，外部程序或其他构件要严格按照此构件的接口约定来调用它（使用它提供的服务），而不需要去探察这些构件的实现细节。构件接口的约定，是一种对接口规格的描述，这种规格也可以被叫做“契约”，它仅仅表示构件实现的功能或交互需求，但不包含具体的实现方法或过程。从一定意义上讲，接口实际上代表了一种需求定义。通过接口设计，可以将需求与实现有机地结合起来，并能够带来构件实现上的柔性和部署上的开放性。

2.3 构件的要素

通常，构件建立于面向对象和分布式基础之上，它有以下 5 个要素。

1. 规格说明

每个构件都提供一定的服务，而接口就是客户方可以向该构件请求的服务的描述。接口概况地描述了客户端应如何与构件进行交互，同时又隐藏了底层的实现细节。构件通过接口输出其功能，外界只能通过接口访问构件。构件的接口使用规格说明进行描述，规格说明除了描述了构件的一系列可用的操作外，还描述了该构件的使用环境（硬件平台、操作系统等）、性能等，并且指导客户方与构件进行交互。虽然构件的接口可以用规格说明进行描述，但构件的接口信息可以独立于任何对其实现的构件而存在。例如，有一组行为被定义为管理一些人的姓名和电话号码，可以将这样一组行为称为接口 IPhoneManagement。假设有两个软件公司开发提供电话号码管理功能的构件，则它们都实现了这个 IPhoneManagement 接口。

2. 一个或多个实现

构件的实现就是以构件的接口说明信息为基础，借助具体的开发平台，用具体的开发工具实现构件接口所描述的功能。构件的接口信息必须被一个或多个实现所支持，这些实现必须符合接口的规格说明。只要构件实现者满足在规格说明中定义的行为，实现者可以选择任何一种被认为是合适的实现方法，即构件规格说明允许构件在内部操作上有很大的自由度，这包括实现时对编程语言的选择。也只有实现了的构件对用户才是真正有意义的。

3. 受约束的构件标准

软件构件模型是关于开发可重用构件和构件之间相互通信的一组标准的描述。软件构件存在于构件模型的环境中才能相互协同工作。构件模型提供了一套服务，并且定义了构件利用这些服务必须遵守的一套规则，例如提供创建和实现构件的指导原则。构件模型规定了构件接口的结构及构件与构件之间、构件与软件架构之间的交互机制，例如，它解决了诸如一个构件怎样使其他构件获得它的服务、怎样在运行时发现新构件和它们的服务这样的问题。

已有的构件模型包括 Microsoft 的 COM+、SUN 的 EJB、OMG 的 CCM 标准。

4. 包装方法

构件可以按不同的方式分组来提供一套可以被替换的服务。这个分组就称为包。一般地，当从第三方购买构件的时候获取的就是这些包。为了使这个包可以使用，在构件模型中要有某种对包的注册机制。

5. 部署方法

一旦构件安装在一个运行环境中，它们将被部署，这时将创建构件的可执行实例，并且允许其他构件与其发生交互。每个实例都是独立地运行于属于自己的进程中。例如，在同一台计算机上可能会有一个构件的两个相互独立的实例在运行，它们处理不同种类的用户请求。

2.4　构件的来源

构件的定义并没有对构件怎样实现提出任何要求，因此，它的来源可以有很多种不同的途径。

- 从现有构件中获取符合要求的构件，直接使用或作适应性修改，得到可复用的构件。
- 提取现有遗产系统的有用功能，这些功能可以包装成构件以在未来使用。
- 从第三方构件市场上购买现成的商业构件。目前市场上已经有大量面向 GUI、数据库和网络的 ActiveX 构件、Java Beans 构件，以及众多的类库、DLL 接口和 API，如开发环境中自带的 ActiveX 构件和 Delphi 构件。这些源代码和目标代码大大提高了程序员的开发效率，但具有更高复用价值的分析设计构件及面向特定应用领域的业务构件还是非常少见。
- 为满足现在的业务需要而从头专门开发的构件。

在进行以上决策时，必须考虑不同方式获取构件的一次性成本和以后的维护成本。虽然构件的来源可能是多种多样的，但由于以下几点原因，使得这些不同来源的构件组装成符合用户要求的目标系统成为可能。

- 使用构件模型作为所有构件必须遵循的标准，不管它们的来源如何。
- 一种构件管理方法，被适当的工具所支持，用于按要求存储、索引、查找及检索构件。
- 一种设计方法，允许当用基于构件的开发方法来设计一个解决方案的构架时，只考虑构件的抽象功能，忽略它们以后的实现特性，这也是基于接口的设计方法的推动力。

2.5　构件的分类

从不同角度出发，可以将构件进行不同的分类。

1. 从构件性质来看，构件可以分为抽象构件和具体构件

① 抽象构件：该类构件是适应领域要求，对同领域一族具有共性和变化性的构件进行的抽象。在抽象构件的接口中有描述领域变化性的成分。抽象构件一般是不够完整的，在使用之前必须具体化。抽象构件的例子包括超类型、超类和带有参数的模板。

② 具体构件：该类构件是相对于抽象构件而言的，具体构件描述应用系统固定的构成成分，其接口不具有描述变化性的成分，它可以不加修改地直接复用。使用具体构件，需要做的只是输入该构件和它所依赖的其他构件。

2. 根据构件重用的方式，通常可以分为白匣子、灰匣子和黑匣子三类

① 白匣子：提供构件的同时也提供实现构件的全部源代码。在应用这个构件的时候，开发人员需要对源代码进行某些修改，然后才能将它集成到应用系统中实现一定的应用目的。

② 灰匣子：灰匣子只提供有关界面部分的源代码，开发人员在应用构件时对构件的内部实现是不清楚的，只能在接口界面上做一些用户化的工作。

③ 黑匣子：黑匣子则完全不提供源代码，其他构件只能通过构件接口访问这个构件。构件应该是封闭、透明、独立、可替换的，而白匣子的可复用和可维护性都较差，因此在基于构件的开发过程中，原则上应尽量不使用白匣子。

3. 根据构件的使用范围，可分为专用构件和通用构件

① 专用构件：针对某个领域中的某一特定系统设计开发的构件。

② 通用构件：可以被某领域所有组织共享的构件。

4. 根据构件的粒度大小，可以分为分布式构件、业务构件和系统级构件

① 分布式构件。这是粒度最小的构件，同时分布式构件也是基于构件的开发方法中一个关键的概念，它作为一种设计工件，在开发生命周期的早期阶段就可以看到，是构建和部署的一个基本单元。

② 业务构件。这是中粒度构件，实现单个业务概念的构件。业务构件通常由一个或多个分布式构件构成，组合在一起解决业务构件所要求的各种分布问题。业务构件可以物理地部署在两台或更多台计算机上。整个基于构件的开发方法是以业务构件概念为核心的。

③ 系统级构件：这是基于构件的开发方法中粒度最大的构件。它将业务构件构成的系统进行封装，将该系统作为一个整体当作黑盒处理，并且它具有清晰的设计接口。

5. 根据构件的功能用途，可以分为系统构件、支撑构件和领域构件

① 系统构件，即在整个构件集成环境和运行环境都使用的构件。

② 支撑构件，在构件集成环境及构件管理系统中使用的构件。

③ 领域构件，即为专门领域开发的构件。

6. 根据构件重用时状态，可以分为动态构件、链接构件和静态构件

① 动态构件，即在软件运行时可以动态嵌入的构件。

② 链接构件，如对象链接和嵌入库（OLE）、动态链接库（DLL）。

③ 静态构件，如源代码、系统分析构件、系统设计构件。

2.6 构件的粒度

2.6.1 构件粒度的概念

软构件粒度的大小是影响软件成本和软件重用质量的重要因素，它是一个难以清晰定义的概念，它表征了构件的规模与复杂程度。一般认为，构件的粒度是用来描述构件所提供特征的粗细程度的量化，或者说是构件所提供特征的计算量的大小，但通常难以精确度量。

2.6.2 构件粒度的划分

软构件可以划分为大粒度构件、中粒度构件和小粒度构件。软构件所处的粒度层次越高，