基于构件开发与传统面向对象开发之比较

雷宁宁[1]，薛锦云[1,2]，刘 超[1]

（1.江西师范大学 计算机信息工程学院，江西 南昌 330027；
2.中国科学院 软件研究所 计算机科学重点实验室，北京 100080）

摘 要：基于构件的软件开发(CBSD)是一种新的软件开发方法，构件技术是它的核心。构件技术以面向对象技术为基础，并很好地发展了面向对象技术。使用基于构件软件开发可以设计出质量好、可靠性高、可重用性好、可维护性好的软件。现在有很多应用软件都是采用基于构件的开发方法。文中概述了构件及CBSD方法的系列核心概念，比较了基于构件软件开发和面向对象软件开发(OOD)的异同。

关键词：基于构件的软件开发；构件；对象；面向对象的软件开发

中图分类号：TP311 **文献标识码**：A **文章编号**：1673－629X(2007)08－0088－04

A Comparison Between CSBD and Traditional OOD

LEI Ning-ning[1], XUE Jin-yun[1,2], LIU Chao[1]

(1. College of Computer Information Engineering, Jiangxi Normal University, Nanchang 330027, China;
2. Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

Abstract: Component－based software development (CBSD) is a new method of software development, whose core is component technique. Component technique is based on OO technique, and also developed OO technique very well. Software with high quality, good reliability, good reusability, good maintainability, can be obtained, when designed using the method of CBSD. Nowadays, lots of applications are designed using the method of CBSD. In this paper, the main concepts of component and the technology of CBSD are introduced, and the similarities and differences of CBSD and traditional objected oriented development (OOD) are attained.

Key words: component－based software development; component; object; OOD

0 引 言

随着计算机软件的广泛应用与发展，软件功能越来越强，并更加易于使用。但是，其规模和复杂度也成倍地增长，因而开发、维护它们的难度和成本也越来越高。另外，要对软件进行修改(如增加新功能或删除不再需要的功能)非常困难。然而，在软件设计过程中，甚至设计完成后，还可能要经常对其进行修改，这也是为什么要提出面向对象软件开发的原因，但是，面向对象未能完全解决这些问题。面向对象修改的代价还是很高，而且也有一定的风险，从而造成软件的可扩充性较差。而构件是“可插拔”的独立的单元[1]，因而基于构件的软件开发方法更好地解决了修改的问题。

基于构件的软件开发(CBSD)是近几年软件界发展的一种新的软件设计方法，它被开发人员普遍看好而且发展很快。这种方法以面向对象技术为基础，通过对构件的选择、例化、组装和集成来构造新的应用系统。构件技术是基于构件的开发的核心技术。构件技术以面向对象技术为基础，并很好地发展了面向对象技术，它的目的是将对象(包括用户界面、对外接口等属性以及对象的功能)实现封装成一个规范的、标准的、可以方便地被构件容器所操纵和使用的整体，使其成为一个通用、高效的软件部件。使用CBSD可以设计出质量好、可靠性高、可扩展性好、可重用性好、可维护性好的软件[1]。现在有很多应用软件都是采用基于构件的开发方法。

文中描述了构件的相关概念，并将基于构件的软件开发与传统的面向对象的软件开发做了比较，阐述了基于构件的软件开发与传统的面向对象开发的区别，进而阐述了CBSD的优势和它所带来的风险。

收稿日期：2006－11－17

基金项目：国家自然科学基金资助项目(60273092)

作者简介：雷宁宁(1982－)，女，江西进贤人，硕士研究生，研究方向为软件工程；薛锦云，教授，博士生导师，研究方向为软件形式化与自动化、软件工程；刘 超，副教授，研究方向为计算机系统结构。

1　构件的相关概念

1.1　构件的定义

软件构件概念虽然已经出现三十多年了,但是,到目前为止,依然没有形成一个能够被广泛接受的定义。一些比较典型的定义有:

(1)Desmond D' Souza & Alan Wills 将软件构件定义为[2]:

一个可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,通过接口与其它构件组装成更大的整体。

(2)Clemens Szyperski 认为[3]:

一个软件构件是一个仅通过规范的接口和确定的上下文依赖进行组装的单元,能够被独立地部署和由第三方组装。

(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如下定义[4]:

构件:①是一个不透明的功能实现;②能够被第三方组装;③符合一个构件模型。

综合上述几种典型的观点,可以概括出构件的一般性定义,即构件是一个可配置的实体,可独立开发和交付的软件单位,其存在的目的是向外界(应用框架、其它构件或最终用户)提供服务(services)。

1.2　构件技术

软件构件技术是以面向对象技术为基础[1],以嵌入后马上可以使用的即插即用型构件为中心,通过构件的组合来建立应用的技术体系,是通过构件组合支持应用的开发环境和系统的总称。构件技术是支持软件复用的核心技术,是近几年来迅速发展并受到高度重视的一个学科分支。

其主要研究内容包括[5]:

(1)构件获取:有目的的构件生产和从已有系统中挖掘提取构件;

(2)构件模型:研究构件的本质特征及构件间的关系;

(3)构件描述语言:以构件模型为基础,解决构件的精确描述、理解及组装问题;

(4)构件分类与检索:研究构件分类策略、组织模式及检索策略,建立构件库系统,支持构件的有效管理;

(5)构件复合组装:在构件模型的基础上研究构件组装机制,包括源代码的组装和基于构件对象相互操作性的运行级组装;

(6)标准化:构件模型的标准化和构件库系统的标准化。

2　构件与对象类区别

构件的概念和面向对象中的对象类概念有很多类似之处,但是也有一些区别。总的来说,构件和对象类的区别主要体现在以下几个方面[1,6]:

①平台无关性。应用系统可以在不同的平台上使用相同的构件来完成特定功能。

②语言无关性。构件可以使用任何一种构件标准支持的语言进行编写,如 VC, Java 等。

③所有交互都通过接口实现。构件通过接口提供服务,使用构件,要求通过这些接口提出请求,而不是通过访问构件的内部实现细节。

④构件是独立的、可组装的实体。构件是不需要重新编译,可独立执行的,可独立地被生产、获得。构件可以与其他构件组装形成更大的整体。

⑤构件是标准化的。与对象类不同,构件必须按照一定的构件模型来实现。

3　基于构件的软件开发

基于构件的软件开发(Component Based Software Development, CBSD)是指利用已开发完成的商业构件(或可复用的构件) 按应用需求组装形成软件应用系统的软件开发方法。基于构件的软件开发有时也称为基于构件的软件工程(Component Based Software Engineering, CBSE)[1]。

基于构件的软件设计与其他传统方法的设计有一些不同之处,CBSD 的关键是自上而下地将需求分解为构件集合, 及自下而上地将构件组合成目标应用系统。基于构件的软件开发方法有: 需求分析和选择构件、构件修改与测试、专用构件的开发与测试、构件的组装和测试、系统的演化[1,6,7]。

3.1　需求分析和选择构件

需求分析阶段除分析、确定系统构架外,还要进行构件的评价。一般分两步:查找所要构件和评价构件。首先按需求分析结果,从构件库或构件市场查找所需要的标准的构件,了解构件的功能、可靠性、可预测性等特性,包括市场份额、以前业绩;然后选出一些候选构件,再进行评价和决策。

3.2　构件的修改与测试

确定的构件不可能马上都能用,有时要对已有构件进行属性扩展或修改。这常需由使用者提出扩展或修改的意见,由构件开发者去完成。需要注意的是:只能改内部属性,不能改接口。修改之后的构件必须进行测试,保证构件功能及接口规范的实现。

3.3　专用构件的开发和测试

构件选择和获取后,可能要重新开发一些新构件,

这些新开发的构件,必须在构造应用工具上进行测试,保证构件功能及接口规范的实现。

3.4 构件的组装和测试

按照系统构架组装构件,得到应用系统,对应用系统进行总体测试。

3.5 系统的演化

应用系统的维护就是系统的演化过程,系统功能的不断改善和扩展,是通过对系统中构件的升级替换来实现的。随着构件版本的变换,应用系统也形成了不同的版本。这就是系统的演化。

3.6 基于构件的软件开发流程图

基于构件的软件开发流程图如图 1 所示。

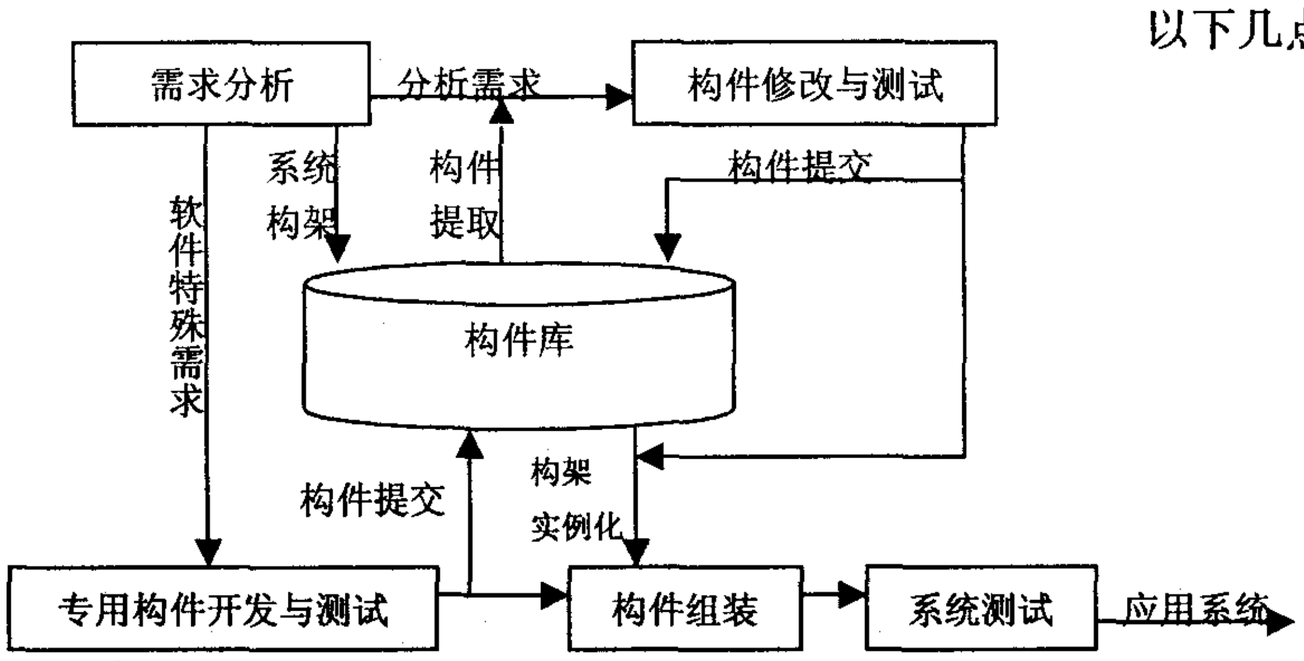

图 1 基于构件的软件开发流程

4 基于构件的软件开发与传统 OOD

4.1 基于构件的软件开发及其与传统 OOD 的比较

由于构件的应用,开发人员所沿用的软件工程方法也正在发生变化,传统的软件开发通常是“白手起家”,开发组织承担系统所有部分的开发,并拥有对整个系统的控制权。一般过程是[1]:

1)获取和定义用户需求;

2)确定满足需求规约的体系结构;

3)详细设计体系结构内部的每个子系统;

4)编码、测试和调试子系统,以符合给定的需求;

5)集成子系统,形成完整的系统;

6)集成测试和系统发布。

在基于构件的系统开发中,系统开发的观念被组装和集成现有构件的观念所取代。CBSD 把系统开发的重点从编程转移到系统的组装方面。在传统开发方法中,系统集成通常是实现过程的最后阶段;CBSD 则强调以构件集成为中心,进行系统构造[6,7]。传统的软件工程过程与 CBSD 的开发的比较如图 2 所示。

可见,传统开发过程从系统需求分析开始然后生成符合相应需求的软件系统,其中工程人员是生产者。CBSD 的开发以构件为中心,而且在可行性研究和需求分析阶段就开始进行构件的收集工作,增加了开发的并行程度,这从另一个方面提高了开发效率。

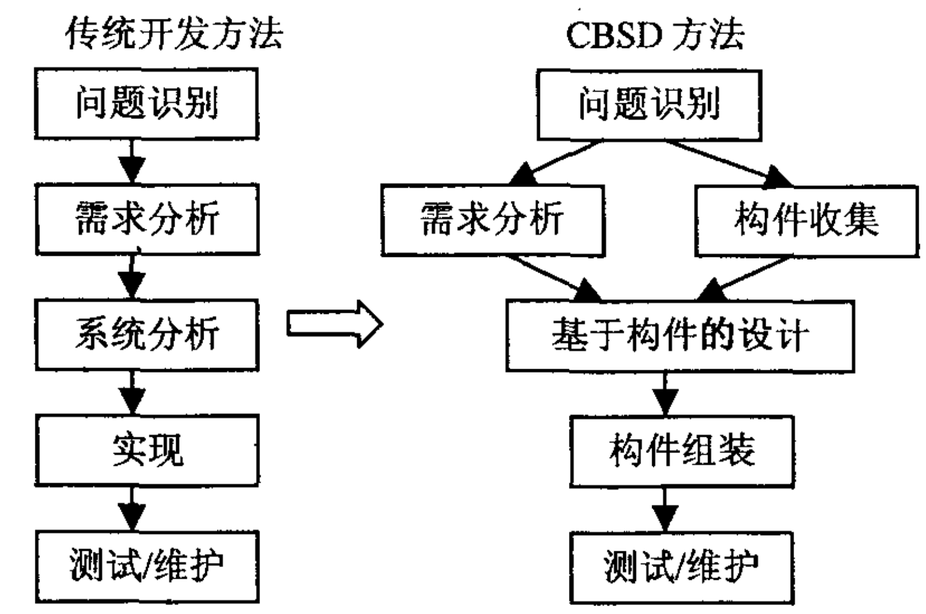

图 2 传统开发方法与 CBSD 方法比较

总的来说,CBSD 和传统 OOD 的区别主要体现在以下几点[1]:

(1)以接口为中心,。构件的接口与实现是分离的,构件通过接口与外界通信。构件接口在粘合和接插独立部署的构件中起主要作用。

(2)独立于程序设计语言。与通过应用系统的开发环境(包括编辑器、编译器、连接器和打包器)集成到应用系统中的源代码不同,构件可以用做二进制代码,或更一般地,用做可直接部署的可执行程序。这意味着可以与构件进行“沟通”,不必考虑实现构件所使用的程序设计语言。

(3)基于构件构架的应用程序组装集成。开发过程是一种组装集成机制,是通过构件组装集成应用系统,而不是从头开发和实现应用系统。

(4) 基于构件开发的应用系统的功能是“可插拔”的,对系统的功能可以根据需要“插接”或是“拔除”,这样就提高了系统的可维护性。

4.2 基于构件的开发的优势

一般来说,基于构件的开发具有以下一些优势[1]:

(1)提高开发速度。由于构件开发商已经编制好了大量的构件模型,因此减少了用户开发的工作量,使开发周期大大缩短。

(2)降低开发成本。由于用户进行软件开发的工作量大大降低,因此开发成本也相应减少。

(3)增加应用软件的灵活性、可扩展性和可重用性。由于应用软件是在基于构件编制的,因此对于使用者的不同需要,往往通过更换、修改应用中的一个或几个部件就可以实现。

(4)降低软件维护费用。由于基于构件的应用软件修改起来比较简单,一般都是通过修改构件来实现,而不需要对整个软件进行全方位的大规模修改。因此,软件的维护费用也大大降低。

4.3 基于构件的开发的缺点

虽然 CBSD 有许多优点,但也存在不少缺点,这些缺点给软件开发带来一定的风险。基于构件的开发具有以下缺点[1,5]:

①开发者无法得到构件的源码,也就无法通过修改源码来改变构件的功能,这就意味着分析、调试以及测试构件必须完全以黑盒子的形式进行。

②构件的使用者无法控制构件的升级。

③在构件集成时,所选用的构件之间可能不匹配或者是作为单独应用设计的构件与软件的其他部分不易交互。在有些时候这些问题在开发阶段的后期才会暴露出来。

5 结　论

基于构件的开发方法由于对软件功能性、开发效率、质量、可靠性、可移植性等方面的良好支持而受到越来越多软件开发组织的重视[1,5]。作为一种新的软件开发方法,它在参照准则、质量控制、开发过程等方面使软件工程的研究与实践产生了极大的更新和改进,相对传统开发方法体现出许多新优势,同时随着实施的进展也暴露出许多问题,有待研究和开发人员在更多的工作中进行分析总结、充实和完善 CBSD 的理论体系。

参考文献:

[1] Sommerville I. 高级软件工程[M]. 第 7 版. 北京:机械工业出版社,2004:440 - 459.

[2] Souza D D, Wills A. UML 对象、组件和框架——CATALYSIS 方法[M]. 王　慧,等译. 北京:清华大学出版社,2004:3 - 32.

[3] Szyperski C, Pfister C. WCOP'96 Summary in ECOOP'96 Workshop Reader[M]. Heidelberg: Verlag, 1997:127 - 130.

[4] Bachman F, Bass L, Buhman C, et al. Volume Ⅱ: Technical Concepts of Component - based software Engineering [R/OL]. 2nd Edition. Pittsburgh: Carnegie Mellon Software Engineering Institute, 2000. Chapter 4. http://www.sei.cmu.edu/publications/documents/00. reports/00tr008/00tr008chap04.html.

[5] 杨芙清,梅　宏,李克勤. 软件复用与软件构件技术[J]. 电子学报,1999(2):68 - 75.

[6] Brow A W. 大规模基于构件的软件开发[M]. 赵文耘,张志,等译. 北京:机械工业出版社,2003.

[7] 梅　宏,陈　锋,冯耀东,等. ABC:基于体系结构面向构件的软件开发方法[J]. 软件学报,2003,14:721 - 732.

(上接第 83 页)

[D]. New Mexico: New Mexico Institute of Technology, 1997.

[8] Oikawa S, Rajkumar R. Linux/RK: A Portable Resource Kernel in Linux[C]// In 19th IEEE Real - Time Systems Symposium. Madrid, Spain:[s. n.],1998.

[9] Liu S, Rajkumar R, Lehoczky J. Priority Inheritance Protocols: An Approach to Real - Time Synchronization[J]. IEEE Transaction on Computers,1990,39(9):1175 - 1185.

[10] Lin Kwei - Jay, Wang Yu - Chung. The Design and Implementation of Real - Time Schedulers in RED - Linux[J]. Proceedings of the IEEE, 2003, 91(7):1114 - 1129.

[11] Oikawa S, Rajkumar R. Portable RK: A portable resource kernel for guaranteed and enforced timing behavior[C]// In Proceedings of the IEEE Real - Time Technology and Applications Symposium. Vancouver:[s. n.],1999:111 - 120.

[12] 陈向群,杨芙清. 面向 Aspect 的操作系统研究[J]. 软件学报,2006,17(3):620 - 627.

(上接第 87 页)

Proc. of the 11th Int. Conf. on Data Engineering. Taipei:[s. n.],1995:3 - 14.

[2] Srikant R, Agrawal R. Mining Sequential Patterns: Generalizations and Performance Improvements[C]//In: Proc. of the Fifth Int. Conf. on Extending Database Technology (EDBT). Avignon, France:[s. n.],1996.

[3] Pei J, Han J, Mortazavi - Asl B, et al. PrefixSpan: Mining sequential patterns efficiently by prefix - projected pattern growth[C]//In: Proc. 2001 Int. Conf. Data Engineering (ICDE'01). Heidelberg, Germany:[s. n.],2001:215 - 224.

[4] Han J, Pei J, Mortazavi - Asl B, et al. FreeSpan: Knowledge Discovery and Data Mining (KDD'00). Boston, MA:[s. n.], 2000:355 - 359.

[5] 孙　莹,胡学钢. 基于 Bitmap 的序列模式研究[D]. 合肥:合肥工业大学,2004.

[6] Lin T Y, Hu X. Louie E. A Fast Association Rule Algorithm Based on Bitmap and Granular Computing[C]//The Proceedings of the IEEE International Conference Fuzzy Systems. St Louis, Missouri:[s. n.],2003:678 - 683.

[7] Morzy T, Zakrzewicz M. Group Bitmap Index: A Structure for Association Rules Retrieval[C]//Proc. of the 4th Int'l Conf. on Knowledge Discovery and Data Mining (KDD - 98). US: American Association for Artificial Intelligence, 1998.

[8] Amer - Yahia S, Johnson T. Optimizing Queries on Compressed Bitmaps[C]//Proc. of the 20th VLDB Conf. Cairo, Egypt:[s. n.],2000.