计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
43(8):1375~1382, 2006

构件软件测试技术研究进展

毛澄映　　卢炎生
(华中科技大学计算机科学与技术学院　武汉　430074)
(maochy@yeah.net)

Research Progress in Testing Techniques of Component-Based Software

Mao Chengying and Lu Yansheng
(*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract　Software component technology provides a more effective design pattern than object-oriented methodology. Component-based software has been widely used in many fields and become a fairly popular software form in recent years. However, the features of component, such as information encapsulating and high evolvability, and properties of component composition (e.g., heterogeneous, loosely coupled and interoperability, etc.) bring great challenges to the testing of component-based software systems. In this paper, the representative testing methods and techniques for component and component-based software in recent years are analyzed and surveyed. Some effective testing frameworks and tools are also summarized and compared. Furthermore, the directions of future research are explored.

Key words　component-based software (CBS); testing; testability; internetware; reliability

摘　要　软构件技术提供了一种较面向对象方法更为有效的软件设计模式,构件软件被广泛应用并成为一种主流软件形态.然而,构件的内部信息屏蔽、演变速度快以及构件间的异质、松耦合等特点给构件软件系统的测试带来极大的挑战,寻求高效的构件软件测试技术和开发实用的测试工具是当今软件业界一个亟待解决的课题.分析和归纳近年来一些典型的构件、构件软件测试方法和技术并对当前较为有效的测试框架和工具进行总结;最后,对其今后若干研究方向进行了展望.

关键词　构件软件;测试;可测试性;网构软件;可靠性
中图法分类号　TP311.5

软构件技术把构架从系统逻辑中清晰地隔离出来,可以实现不同开发语言和平台的协作,组织大规模的开发,而且使得系统的开发周期更短、造价更低.基于构件的软件工程(CBSE)已成为软件工程领域的研究热点.

构件形态也先后经历了函数库、类库、可重用中间件(如 CORBA)到可重用框架(如 EJB,COM+等)的演变.虽然各种新型构件开发技术能极大地提升其性能,但并不足以完全解决构件软件可靠性的困扰,因为作为人为智力活动的程序开发不可避免地会使构件中潜留缺陷.对构件和构件系统进行测试是保障和提高系统可靠性的重要途径[1].构件的提供者在开发构件过程中要不断地进行测试以保证其质量,构件的使用者也要在实际运行环境中对单个构件和整个构件软件实施测试.

1　构件软件特点及其测试挑战

构件与传统软件模块的区别主要体现在:①以第3方复用为目的进行开发,多数情况下源代码

收稿日期:2005-06-26;修回日期:2006-01-13
基金项目:湖北省自然科学基金项目(2005ABA266)

对使用者不可见;②是一个可发布、可执行、有良好接口的高质量模块;③可在不同项目的不同上下文环境中被复用.一般而言,基于构件的软件系统(component-based software)简称构件软件,有如下特点:①以独特的软件组装过程开发,是多种异质构件的松耦合结构;②由于构件易被替换,因而构件软件的演变速度快;③系统的可靠性对构件质量高度依赖;④系统有更好的互操作性、可扩展性和重用性.此外,构件软件常应用于网络分布式环境之中.

构件及构件软件所固有的新特性却给测试带来一系列问题[2-3]:构件提供者需要使用相当充分的覆盖准则进行测试以提高构件的可复用性;对用户使用构件的上下文环境并不十分了解;并且不能很好地获得构件的错误报告.对构件使用者而言,源代码不可见性给测试设计和用例生成带来极大的障碍;系统中构件的异质(混杂)性不利于测试标准的统一及自动化的实现;构件版本变更快及不确定性迫使要对构件系统进行较为频繁的回归测试等.

构件软件测试通常分为构件测试、子系统测试、系统测试3个阶段[4],分别与传统软件测试过程的单元测试、集成测试和系统测试相对应.一般来讲,构件测试主要由构件的开发者完成,而后两个阶段的测试则由使用者实施.

2 构件单元测试技术

2.1 基于传统方法扩展的技术

面向对象技术为构件的实现提供了基本技术支持.在绝大部情况下,构件采用面向对象程序设计语言(如Java,C++/C#等)实现.对单个构件进行测试时,传统的面向对象测试技术可以直接或经改造用于软构件的测试.

(1) 构件结构性测试

构件结构性测试一般只能由构件的开发者进行(通过Java字节码分析的方法[5]除外),并且其测试方法和类结构性测试并无太大差别[6],只是测试时要求选用更严格的测试准则和达到更高的覆盖率,以便适用不同使用者的上下文环境和提高构件的可复用性.

(2) 构件功能性测试

从构件使用者的角度看,构件是一个具备良好接口、不可修改、内部不可见的封装体.该特征非常符合传统意义上黑盒测试的要求.Ramachandran[7]首先分析构件接口的类型、取值范围,再运用边界值分析技术产生测试用例,最后使用一个集成工具StP-Test实施测试.

基于状态转换的测试是最为典型的类测试方法.Kung等人提出一种基于感知状态机的测试方法[8].构件使用者根据IDL文件提供的有限信息人为地理解构件的状态变量和预期行为,结合所提供的接口函数(即转换)构造状态机,再由此产生用例并运用重放(replay)机制实施测试.该方法简单、直观,能实现测试用例的自动生成,适合于分布式、并发系统中的构件测试.但仅依靠接口文件和自身的理解往往导致构件状态行为建模不准确且难以保证正确性.类似地,文献[9]基于构件规约(specification)构造通信有限自动机CFSM来监测构件软件在实际运行环境中的错误.Beydeda等人则将构件状态机CSM和具体实现代码相结合,形成一种黑盒-白盒组合测试策略[10].

2.2 构件性能与兼容性测试

当构件应用于实时系统或给外界提供服务的系统(如Web services)中时,不仅要保证构件功能的正确性,还需要保证其性能和健壮性.通常,性能测试是通过在构件中内建测试代码的方法来实现(内建式测试,built-in test,参见第4.1节),对运行时构件行为的观测可以衡量其服务质量QoS[11]和对实时约束的满足能力.例如,在Vincent等人提出的增强实时可测试性的内建式测试框架中[12],在被测构件中内建定时(timing)模块和相应的接口,测试时当该模块负责的事件完成后就会通知外部的测试器去判断是否符合时间约束要求.

构件开发者在开发构件时,要考虑到构件对各种运行环境的适应能力,还要考虑与其他构件在接口上的适应性;同样,使用者在选择构件组装成系统时也要考察构件间的兼容性.测试构件兼容性的信息来源主要是构件接口,即重点是进行构件接口兼容性的测试.构件规约的匹配检查[13]和以IDL文件为基础的构件接口兼容性匹配算法[14]都能较好地解决问题.构件预期应用环境繁多,要想对环境中每个部件的可能取值进行穷尽组合几乎不可能.因此,运用成对组合测试(pairwise testing)[15]能实现以最少的用例达到最大可能的覆盖.

3 构件软件集成测试技术

3.1 构件交互测试

构件软件一般通过构件接口组接而成,良好的

协同工作能力主要通过正确的构件交互来保证. 因此,构件交互测试(component interaction testing, CIT)是构件软件测试的一个极其重要方面.

文献[16]对 UML 进行形式化的扩展,将构件之间的交互用标记的转换系统(labeled transition systems, LTS)表示. 在事件流覆盖准则的指导下,运用模型检验(model checking)技术生成用于测试的用例. 所采用的形式化建模语言有良好的语法定义,能简化复杂事件序列的调试,还能处理同步问题;其不足是存在状态组合爆炸问题.

通过分析构件间直接或间接的交互方式,Wu 等人认为测试构件软件时应主要考虑接口、事件、上下文(控制)依赖关系和内容(数据)依赖关系这 4 类元素[17-18]. 用构件交互图(component interaction graph, CIG)描述构件间的交互场景[17],还给出了从所有接口到所有内容依赖的 5 个层次的覆盖准则集. 这种方法能测试基于多种异质构件组建的系统,但处理大型复杂系统时 CIG 图的规模过大.

3.2 变异测试

变异测试(mutation testing/analysis)也称做变异分析,是一种相当成熟的排错性测试技术,为衡量测试用例集的足够性提供了客观准则. 由于构件的源码不可见以及异质性,导致测试构件软件时难以判定测试覆盖的充分性,而变异测试提供了一种解决途径. 对一个假定为'无缺陷'的构件进行变异操作(例如接口变异[19]、规约变异和合约变异[20]等)产生多个版本,再对这些含有错误的版本分别执行测试,根据运行失效的版本数可计算出用例集的覆盖度(或充分性).

3.3 其他集成测试策略

构件组装成系统时还有其他许多特性值得考虑(例如一致性、完整性等). 不少集成测试策略相继被研究者们提出,最为典型的有:①Zhu 等人利用行为观测理论(behavior observation theory)实施构件的集成测试[21],并提出了用于并发系统的观测模式公理. 文献[22]也是利用行为捕获与测试的方法实现构件集成的自动检查与分析. ②用 UML 建模表示的信息(如接口、构件间交互的规约等)可以给测试提供便利. Wu 等人[18]从协作/序列图和状态图中提取构件间控制依赖和数据依赖信息,进而实施构件软件的集成测试. ③当前,更多的集成测试是和自测试(self-test)构件[23]、内建式测试构件[24]的开发结合在一起,是一种基于可测试性改进的测试策略.

4 基于可测试性增强的测试技术

软件可测试性增强的目的是在不增加或者少增加软件复杂性的基础上,将易于测试的原则融合到设计和编码之中,提高测试时对软件的可控制性和可观察性[25-26]. 它体现了软件测试的发展趋势:测试向软件开发阶段的前期迁移,与软件开发的设计和编码相融合. 就构件软件的测试而言,主要分以下 3 类:内建式测试、基于内省/回顾(intro-/retrospection)的测试和基于包装器(wrapper)设计的测试.

4.1 内建式测试

构件内建式测试的基本思想[27]是提供者在构件中预置测试脚本并设置相应的测试接口,使用者通过调用该接口实施构件的自行测试,脚本通常是测试用例或能够产生用例的语句.

文献[28]通过在构件中增加能包含或产生测试用例的附加成员方法来增强可测试性(如图 1 的示例代码). 认为构件有两种运行模式,即常规模式和维护模式. 构件的使用者在维护模式下调用能驱动测试的代表性方法执行测试. 前提是必须假定构件是一个类,但是所内建用于测试的方法可以被子类继承.

```
Class BIT_INT_Stack {
    int stack[N];
    int stack_index;
    BIT_INT_Stack();//The constructor
    ~BIT_INT_Stack();//The destructor
    //Normal member functions
    int BITs_Stack_Empty();
    int Pop();
    int Push(int element);
    //Built-in test implementation
    void BIT_INT_Stack_Test1(…) {…};
    void BIT_INT_Stack_Test2(…) {…};
};//The INT_Stack component
```

Fig. 1 A built-in test component example.

图 1 一个内建式测试构件的示例

Gao 等人针对 EJB 构件,内建能和外界工具交互的接口和用于追踪的函数来确保使用者在执行测试时能方便地观测构件的行为[29],并实现了两个测试辅助工具[30]:Test Agent 和 Tracking Agent. 其特点是可观测性强,利于使用者检测出构件的异常执行行为,但只限定于特定的构件. 同样沿着检测构件行为的思路,文献[11]提出了一个针对 Java 语言的内建合约测试框架 BIT/J,用于验证构件的行为语义并实现对构件 QoS 的测试.

为克服内建式测试用例存储开销过大的不足,

"Component + "的测试方法[31]将要开发的构件分为3类:①有内建测试能力的构件(BIT构件);②能访问BIT构件接口和包含测试用例的构件;③用于运行时失效恢复等用途的构件.这种分类开发构件的方法一定程度上实现了开发和维护的分离,不会增加BIT构件在运行环境中的资源消耗,但会加大使用者部署测试的难度.

4.2 基于内省/回顾的测试

构件开发者所提供的文档往往不完整,或是背离了构件本身应有的语法或语义,从而造成信息缺乏难以实施测试.部分构件(如EJB构件)提供了内省(反射)机制,所提供的部分语法信息能为测试该构件提供便利.元数据(meta-data)方法[32]扩展了用于构件分析和测试的信息,使之包含语义信息.元数据按标准格式组织,不必被包装进构件并能实现按需(on-demand)生成和远程存储,但仅支持从开发者向使用者的单向信息流表示.

回顾方法[33]能实现双向信息交互,除了提供给使用者已测试的历史记录和进一步测试的推荐信息外,使用方的测试和运行信息可以反馈给构件开发方以便质量的进一步改进.由于元数据和回顾方法都未成为一种业内标准,构件开发商不一定提供这方面的支持.

4.3 基于包装器设计的测试

图2是一种典型的基于包装器的构件可测试性增强框架[34-35].包装器一般不会改变构件原有的接口,增加一些能够为测试提供信息的方法或是限制使用某些功能以便于实施测试,它既可以由构件的提供者实现,也可以是使用者以测试为目的定制.实现的方式有多种:Edwards用元数据表示由包装器中的附加方法所提供便于测试的信息[36];Haddox等人[35]实现的包装器则允许给外部接口提供数据流信息,通过错误注入、数据收集和断言检查来实现由第3方构件组成系统的测试;Bertolino等人的方法[37]则允许动态改编构件接口,用XML文档交换信息,通过对测试用例执行信息的追踪实现用例重用时的优化.包装器还具备相当的灵活性:当测试完成后可以卸去包装器从而减少构件运行时占用的资源.

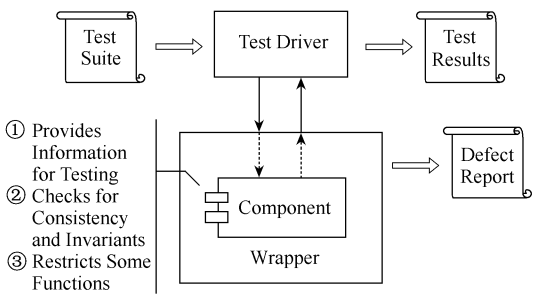

Fig. 2 A testability improvement infrastructure based on wrapper design.

图2 基于包装器的构件可测试性增强框架

5 构件软件回归测试技术

构件软件的后期变更主要源于两个因素:一是由于构件供应商提供了新版构件或者不再提供某些构件的服务等;二是构件软件的开发者改变了组装模式或是重新定制了某些构件.构件软件回归测试同样有确定受影响的模块、选择能检测新引入缺陷的用例等问题.

5.1 回归测试范围的界定

一般来讲,变更的只是极少一部分构件,因此没有必要对所有构件和交互都重新展开测试.通过对构件之间交互行为的图形建模表示,可以计算出变更构件所影响到的构件和接口交互.例如,文献[38]中使用模块推理(modular reasoning)确定需要重新测试的构件集合;而Wu等人则在构件交互图CIG的基础上为每个测试用例构建接口事件执行图(interface and event execution graph, IEEG)[39],利用IEEG图很容易得到受影响的接口、事件和依赖关系等.此外,运用切片技术也能计算出变更所波及的范围[40].

5.2 典型回归测试技术

黑盒测试技术不仅对单个构件和构件间交互的测试有效,也能用到回归测试中.文献[41]从系统开发者对构件进行定制的角度出发,将构件定制分为泛化、特定化和重构3种类型,再将多种黑盒测试技术(如等价类划分、边界值分析)调整用于每类定制的重新测试中.所提出的构件软件黑盒回归测试框架如图3所示.

如果从系统规约的角度考虑变更,则依据

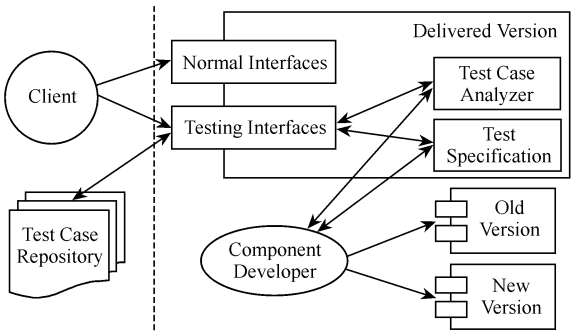

Fig. 3 Testing framework for black-box testing of CBS.

图3 构件软件黑盒回归测试框架

UML 的协作图或状态图可以分析出变更对控制序列的影响,从协作图还可以获得对数据依赖的影响[42]. Sajeev 等人[43]则在方法级考虑变更,对构件的版本数据进行建模并用 XML 文件存储,通过分析对变更方法的直接或间接调用,从用例库中选择出用于回归测试的用例. 版本变更信息、用例与方法调用序列的关联关系是该方法的两个基本前提.

基于构件元内容(meta-content)的回归测试技术[44]运用描述构件信息的元数据和用以计算或检索这些信息的元方法指导用例选择. 除了基于代码元内容的技术外,基于规约元内容的方法使得在构件源代码不可见的情况下也能开展测试. 实验表明,基于元内容的回归测试用例选择技术能平均节省 26% 左右的用例.

此外,基于构件行为捕获与测试的方法也能用于回归测试[22,45],通过检查 I/O 不变式和交互不变式发现由于变更所引发的交互错误[46].

6 网构软件测试技术

基于 Internet 平台的网构软件(internetware)正成为一种新型的软件形态[47]. 由于构件应用于一个开放、动态和多变环境下,并且有着更高的质量要求(如 QoS 的要求),使得对 Web 应用中的构件进行测试相当困难.

文献[48]提出了一种基于移动代理(mobile agent)的 Web 服务构件测试技术:移动代理对象以操作剖面(operational profile)为参照创建并携带测试用例到远程的 Web 服务站点进行测试,能减少调用远程服务的代价并能极大地增加测试的并行性. 姜瑛等人[20]提出了一种基于合约式设计的 Web 服务测试技术框架,可以添加服务提供者和使用者的合约,根据测试数据生成 XML 格式的测试脚本,在测试执行时进行合约检查,返回测试结果并提交测试报告. 其缺点是合约格式尚有一定的局限性,且目前实现的原型工具 WSTT 难以处理复杂的数据类型. 以上两种方法在衡量测试用例的充分性时均采用变异分析技术.

7 构件软件测试工具

构件的开发者可沿用已经产品化的面向对象软件测试工具对构件进行单元测试,这些工具有 C++ Test,Panorama++ 和 ParaSoft JTest 等. 就纯粹用于测试构件的工具而言,目前尚未见成型的商业产品,主要以原型工具为主,表 1 给出了它们的概况. 此外,一些诸如 CDT[4] 的测试框架相继被提出,能在一定程度上指导测试工具的实现.

Table 1 Some Representative Tools for Testing CBS

表 1 典型的构件软件测试工具

Tools	Supported Comp.	Testing Tech.	Strong Points	Shortcomings
CTM[49]	EJB components	Functional testing	① Can evaluate a large number of components within short period of time in a cost-effective way.	① Needs the metadata provided by component developer.
OTS[50]	Components in distributed system	Interactive testing based on observation	① System specification and test cases are written in standard TCNN-3 format; ② Can directly check system's behaviors at runtime.	① Needs the instrumentation for subject system.
ConAn[51]	Concurrent Java component	Unit testing, black-& white-testing	① Adequate controllability over the interactions between Java threads; ② Be proved to be feasible by a number of industry applications.	① The detected faults are not comprehensive; ② Can't conductthread priority.
BIT/J[11]	Java component	Component behavior testing based on monitoring	① Supports the remote testing over the Internet; ② No need to access the component's source code.	① Depends on the Java reflectance package.
JaBUTi[5]	Java component	Structural testing	① Works at the bytecode level rather than source code; ② Provides detailed coverage reports and support customization; ③ User-friendly interfaces for visualizing control-flow and data-flow.	① Can't support test cases automatic generation and integration testing at present.
SCTE[52]	C++/Java component	State-based testing	① Can extract information from two commercial CASE tools; ② Supports test generation and automated testing.	① Can't support integration testing at present but can be extended.

Table Continued
续表

Tools	Supported Comp.	Testing Tech.	Strong Points	Shortcomings
COACH[53]	CORBA component	Unit and interactive testing based on trace (monitoring)	① Supports different levels testing; ② Can visualize causality relations between invocations at runtime; ③ Can run standardized abstract TTCN-3 test cases; ④ Can realize fault localization.	① Merely supports the CCM components.

8 结 语

纵观构件软件测试方法和技术的现状,尚有如下方向值得深入研究:

(1) 内建式测试中如何进行交换信息的表达、理解和提取等均需进一步明确化以便实用.

(2) Agent 体具有自治性、智能性和良好的协作和通信能力,将智能 Agent 技术应用于构件软件(特别是分布式构件系统)的测试将提高灵活性和效率,有利于实现测试的自动化等[54].

(3) 结合程序切片技术将便于回归测试用例的选择与优化.

(4) 当前的构件软件测试辅助工具总体上存在应用面狭窄、自动化程度不高和不能较好地指导缺陷定位等问题,针对普遍使用的构件模型研制和开发自动化或半自动化的测试工具也是极具现实意义的课题.

参 考 文 献

[1] E J Weyuker. Testing component-based software: A cautionary tale [J]. IEEE Software, 1998, 15(5): 54-59

[2] M J Harrold. Testing: A roadmap [C]. In: Proc of the Future of Software Engineering (Special Volume of the Proceedings of the Int'l Conf on Software Engineering). New York: ACM Press, 2000. 63-72

[3] S Beydeda, V Gruhn. State of the art in testing components [C]. In: Proc of the 3rd Int'l Conf on Quality Software (QSIC '03). Los Alamitos, CA: IEEE Computer Society Press, 2003. 146-153

[4] A Bertolino, A Polini. A framework for component deployment testing [C]. In: Proc of the 25th Int'l Conf on Software Engineering (ICSE '03). Los Alamitos, CA: IEEE Computer Society Press, 2003. 221-231

[5] A M R Vincenzi, J C Maldonado, W E Wong, *et al*. Coverage testing of Java programs and components [J]. Science of Computer Programming, 2005, 56(1/2): 211-230

[6] Y Pan, M Chen. Dataflow analysis and testing of software components [C]. In: Proc of the 8th IASTED Int'l Conf on Software Engineering and Applications (SEA '05). Calgary, Canada: ACTA Press, 2004. 136-142

[7] M Ramachandran. Testing software components using boundary value analysis [C]. In: Proc of the 29th EUROMICRO Conf New Waves in System Architecture (EUROMICRO '03). Los Alamitos, CA: IEEE Computer Society Press, 2003. 94-98

[8] H W Sohn, D C Kung, P Hsia. CORBA components testing with perception-based state behavior [C]. In: Proc of the COMPSAC '99. Los Alamitos, CA: IEEE Computer Society Press, 1999. 116-121

[9] M Zulkernine, R Seviora. Towards automatic monitoring of component-based software systems [J]. The Journal of Systems and Software, 2005, 74(1): 15-24

[10] S Beydeda, V Gruhn. An integrated testing technique for component-based software [C]. In: Proc of the ACS/IEEE Int'l Conf on Computer Systems and Applications. Los Alamitos, CA: IEEE Computer Society Press, 2001. 328-334

[11] F Barbier, N Belloir. Component behavior prediction and monitoring through built-in test [C]. In: Proc of the 10th IEEE Int'l Conf and Workshop on the Engineering of Computer-Based Systems (ECBS '03). Los Alamitos, CA: IEEE Computer Society Press, 2003. 17-22

[12] J Vincent, G King, P Lay, *et al*. Principles of built-in-test for run-time-testability in component-based software systems [J]. Software Quality Journal, 2002, 10(2): 115-133

[13] A M Zaremski, J M Wing. Specification matching of software components [J]. ACM Trans on Software Engineering and Methodology, 1997, 6(4): 333-369

[14] J M Zaha, M Geisenberger, M Groth. Compatibility test and adapter generation for interfaces of software components [G]. In: Proc of the ICDCIT 2004, LNCS 3347. Berlin: Springer-Verlag, 2004. 318-328

[15] A W Williams, R L Robert. A measure for component interaction test coverage [C]. In: Proc of the ACS/IEEE Int'l Conf on Computer Systems and Applications. Los Alamitos, CA: IEEE Computer Society Press, 2001. 304-311

[16] W Liu, P Dasiewicz. Component interaction testing using model-checking [C]. In: Proc of the Canadian Conf on Electrical and Computer Engineering (Volume 1). Los Alamitos, CA: IEEE Computer Society Press, 2001. 41-46

[17] Y Wu, D Pan, M H Chen. Techniques for testing component-based software [C]. In: Proc of the 7th IEEE Int'l Conf on Engineering of Complex Computer Systems (ICECCS '01). Los Alamitos, CA: IEEE Computer Society Press, 2001. 222-232

[18] Y Wu, M H Chen, J Offutt. UML-based integration testing for component-based software [G]. In: Proc of the ICCBSS '03, LNCS 2580. Berlin: Springer-Verlag, 2003. 251-260

[19] S Ghosh, A P Mathur. Interface mutation to assess the adequacy of tests for components and systems [C]. In: Proc of the 34th Int'l Conf on Technology of Object-Oriented Languages and Systems (TOOLS). Los Alamitos, CA: IEEE Computer Society Press, 2000. 37-46

[20] Jiang Ying, Xin Guomao, Shan Jinhui, *et al*. A method of automated test data generation for Web service [J]. Chinese Journal of Computers, 2005, 28(4): 568-577 (in Chinese)
(姜瑛, 辛国茂, 单锦辉, 等. 一种Web服务的测试数据自动生成方法[J]. 计算机学报, 2005, 28(4): 568-577)

[21] H Zhu, X He. An observational theory of integration testing for component-based software development [C]. In: Proc of the COMPSAC'01. Los Alamitos, CA: IEEE Computer Society Press, 2001. 363-368

[22] L Mariani, M Pezze. Behavior capture and test: Automated analysis of component integration [C]. In: Proc of the ICECCS'05. Los Alamitos, CA: IEEE Computer Society Press, 2005. 292-301

[23] L Mariani, M Pezze, D Willmor. Generation of integration tests for self-testing components [G]. In: Proc of FORTE 2004 Workshops, LNCS 3236. Berlin: Springer-Verlag, 2004. 337-350

[24] H G Gross, N Mayer. Built-in contract testing in component integration testing [J]. Electronic Notes in Theoretical Computer Science, 2003, 82(6): 1-11

[25] E Martins, C M Toyota, R L Yanagawa. Constructing self-testable software components [C]. In: Proc of the Int'l Conf on Dependable Systems and Networks. Los Alamitos, CA: IEEE Computer Society Press, 2001. 151-160

[26] Shan Jinhui, Jiang Ying, Sun Ping. Research progress in software testing [J]. Acta Scientiarum Naturalium Universitatis Pekinensis, 2005, 41(1): 134-145 (in Chinese)
(单锦辉, 姜瑛, 孙萍. 软件测试研究进展[J]. 北京大学学报(自然科学版), 2005, 41(1): 134-145)

[27] S Beydeda. Research in testing COTS components-built in testing approaches [C]. In: Proc of the 3rd ACS/IEEE Int'l Conf on Computer Systems and Applications. Los Alamitos, CA: IEEE Computer Society Press, 2005. 101-104

[28] Y Wang, G King, H Wickburg. A method for built-in tests in component-based software maintenance [C]. In: Proc of the European Conf on Software Maintenance and Reengineering (CSMR'99). Los Alamitos, CA: IEEE Computer Society Press, 1999. 186-189

[29] J Gao, K Gupta, S Gupta, *et al*. On building testable software components [G]. In: Proc of the COTS-Based Software Systems (ICCBCC), LNCS 2255. Berlin: Springer-Verlag, 2002. 108-121

[30] J Gao, E Y Zhu, S Shim. Monitoring software component and component-based software [C]. In: Proc of the COMPSAC2000. Los Alamitos, CA: IEEE Computer Society Press, 2000. 403-412

[31] J Hornstein, H Edler. Test reuse in CBSE using built-in tests [C]. In: Proc of the Workshop on Component-Based Software Engineering, Composing Systems from Components. Los Alamitos, CA: IEEE Computer Society Press, 2002. 11-14

[32] A Orso, M J Harrold, D Rosenblum. Component metadata for software engineering tasks [G]. In: Proc of the Int'l Workshop on Engineering Distributed Objects (EDO), LNCS 1999. Berlin: Springer-Verlag, 2000. 129-144

[33] C Liu, D Richardson. Software components with retrospectors [C]. The Int'l Workshop on the Role of Software Architecture in Testing and Analysis (ROSATEA), Marsala, Sicily, Italy, 1998

[34] S H Edwards. A framework for practical, automated blackbox testing of component-based software [J]. Software Testing, Verification and Reliability, 2001, 11(2): 97-111

[35] J M Haddox, G M Kapfhammer, C C Michael. An approach for understanding and testing third party software components [C]. In: Proc of the Annual Reliability and Maintainability Symposium. Los Alamitos, CA: IEEE Computer Society Press, 2002. 293-299

[36] S H Edwards. Toward reflective metadata wrappers for formally specified software components [C]. In: Proc of the Workshop on Specification and Verification of Component Based Systems, held in conjunction with OOPSLA 2001. New York: ACM Press, 2001. 1-8

[37] A Bertolino, A Polini. WCT: A wrapper for component testing [G]. In: Proc of the FIDJI 2002, LNCS 2604. Berlin: Springer-Verlag, 2003. 165-174

[38] B W Weide. Modular regression testing: Connections to component-based software [C]. In: Proc of the 4th ICSE Workshop on Component-Based Software Engineering. Los Alamitos, CA: IEEE Computer Society Press, 2001. 47-51

[39] Y Wu, D Pan, M H Chen. Techniques of maintaining evolving component-based software [C]. In: Proc of the 16th Int'l Conf on Software Maintenance (ICSM2000). Los Alamitos, CA: IEEE Computer Society Press, 2000. 236-246

[40] Y Pan, D Pan, M H Chen. Slicing component-based systems [C]. In: Proc of the ICECCS'05. Los Alamitos, CA: IEEE Computer Society Press, 2005. 155-164

[41] Y Wu. Black-box testing for evolving COTS-based software [C]. In: Proc of the IWICSS 2004. Berlin: Springer, 2004. 39-45

[42] Y Wu, J Offutt. Maintaining evolving component-based software with UML [C]. In: Proc of the CSMR'03. Los Alamitos, CA: IEEE Computer Society Press, 2003. 133-142

[43] A S M Sajeev, B Wibowo. Regression test selection based on version changes of components [C]. In: Proc of the 10th Asia-Pacific Software Engineering Conference (APSEC'03). Los Alamitos, CA: IEEE Computer Society Press, 2003. 78-85

[44] A Orso, M J Harrold, D Rosenblum, *et al*. Using component metacontent to support the regression testing of component-based software [C]. In: Proc of the ICSM'01. Los Alamitos, CA: IEEE Computer Society Press, 2001. 716-725

[45] L Mariani. Behavior capture and test for verifying evolving component-based systems [C]. In: Proc of the ICSE'04. Los Alamitos, CA: IEEE Computer Society Press, 2004. 78-80

[46] L Mariani, M Pezze. A technique for verifying component-based software [J]. Electronic Notes in Theoretical Computer Science, 2005, 116(1): 17-30

[47] Yang Fuqing. Thinking on the development of software engineering technology [J]. Journal of Software, 2005, 16(1): 1-7 (in Chinese)
(杨芙清. 软件工程技术发展思索[J]. 软件学报, 2005, 16(1): 1-7)

[48] J Zhang. An approach to facilitate reliability testing of Web services components [C]. In: Proc of the 15th Int'l Symposium on Software Reliability Engineering (ISSRE'04). Los Alamitos, CA: IEEE Computer Society Press, 2004. 210-218

[49] Y S Ma, S U Oh, D H Bae, *et al*. Framework for third party testing of component software [C]. In: Proc of the APSEC'01. Los Alamitos, CA: IEEE Computer Society Press, 2001. 431-434

[50] P H Deussen, G Din, I Schieferdecker. An on-line test platform for component-based systems [C]. In: Proc of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW'02). Los Alamitos, CA: IEEE Computer Society Press, 2002. 96-103

[51] B Long, D Hoffman, P Strooper. Tool support for testing concurrent Java components [J]. IEEE Trans on Software Engineering, 2003, 29(6): 555-566

[52] H K Kim, O H Kwon. SCTE: Software component testing environments [G]. In: Proc of the ICCSA 2005, LNCS 3481. Berlin: Springer-Verlag, 2005. 137-146

[53] H Batteram, W Hellenthal, W Romijn, *et al*. Implementation of an open source toolset for CCM components and systems testing [G]. In: Proc of the TestCom 2004, LNCS 2978. Berlin: Springer-Verlag, 2004. 1-16

[54] J Grundy, G Ding, J Hosking. Deployed software component testing using dynamic validation agents [J]. The Journal of Systems and Software, 2005, 74(1): 5-14

Mao Chengying, born in 1978. Ph D. His main research directions include software analysis, comprehension and testing.
毛澄映,1978 年生,博士,主要研究方向为软件分析、理解与测试.

Lu Yansheng, born in 1949. Professor and Ph D supervisor of Huazhong University of Science and Technology. His main research interests include advanced database system, software testing, component technology and data mining.
卢炎生,1949 年生,教授,博士生导师,主要研究方向为现代数据库系统、软件测试与构件技术、数据挖掘等.

Research Background

Some specialties of component, such as high evolvability and implementation transparent, and properties of component composition (e.g., heterogeneous, loosely coupled, etc.) bring great challenges to the testing of component-based software (CBS). In this paper, we firstly address the characteristics of CBS and its testing difficulties. Then we deeply analyze and survey some recent research developments related to the representative testing methods and techniques for CBSs. We also briefly summarize and compare several effective testing frameworks and tools in this field. Furthermore, we explore some ongoing research directions in future work. This work is supported by the Natural Science Foundation of Hubei Province of China under Grant No.2005ABA266.