北京大学学报(自然科学版),第 41 卷,第 1 期,2005 年 1 月
Acta Scientiarum Naturalium
Universitatis Pekinensis, Vol. 41, No. 1 (Jan. 2005)

述 评
Review

软件测试研究进展[1)]

单锦辉[2),3)] 姜 瑛[2),4)] 孙 萍[3)]

([2)] 北京大学信息科学技术学院软件研究所,北京,100871,E-mail:{shanjh, jiangy}@sei.pku.edu.cn;
[3)] 酒泉卫星发射中心; [4)] 昆明理工大学信息工程与自动化学院,昆明,650093)

摘 要 从软件测试的技术与过程、持续的软件测试、软件测试的充分性准则等方面简要介绍软件测试的基本思想;讨论软件测试中的若干问题,包括面向路径的测试数据自动生成、测试预言、期望结果的自动生成、回归测试等;并且探讨软件测试的发展趋势,包括构件测试、软件的易测试性与基于合约的构件易测试性设计和 Web Services 测试等。

关键词 软件测试; 软件质量; 软件的易测试性; 构件测试
中图分类号 TP 311

0 引 言

随着社会的不断进步和计算机科学技术的飞速发展,计算机及软件在国民经济和社会生活等方面的应用越来越广泛和深入。作为计算机的灵魂,软件在其中起着举足轻重的作用。软件的失效有可能造成巨大的经济损失,甚至危及人的生命安全。例如,1996 年 Ariane 5 运载火箭的发射失败等都是由软件故障引起的[1,2]。

软件开发的各个阶段都需要人的参与。因为人的工作和通信都不可能完美无缺,出现错误是难免的。与此同时,随着计算机所控制的对象的复杂程度不断提高和软件功能的不断增强,软件的规模也在不断增大。例如,Windows NT 操作系统的代码大约有 3 200 万行[3]。这使得错误更可能发生。人们在软件的设计阶段所犯的错误是导致软件失效的主要原因。软件复杂性是产生软件缺陷的极其重要的根源[4]。

软件测试是保证软件质量和可靠性的重要手段。目前许多项目的软件工程实践以结构化分析和设计为核心,在开发阶段的前期,包括需求分析和设计都是以技术评审和工程管理作为质量保证的手段,而技术评审和工程管理主观因素很大,很可能又引入错误并扩展到后续开发阶段[2]。

另一方面,软件测试确实能够发现软件中隐藏的许多缺陷。例如,在英国约克大学为英国海军开发的 SHOLIS 项目中,尽管采用形式化方法描述和证明软件规约,并且采用程序正确性证明方法排除了软件开发前期的许多缺陷,单元测试仍然发现了整个软件开发过程 15.75%

1) 国家"八六三"高技术研究发展计划(2001AA113070)、国家自然科学基金(60373003)、国家重点基础研究发展规划(973 计划)(2002CB312200003)、中国博士后科学基金(2003034077)资助项目
收稿日期:2003-10-13;修回日期:2004-05-24

的缺陷[5]。

随着人们对软件测试重要性的认识越来越深刻, 软件测试阶段在整个软件开发周期中所占的比重日益增大。现在有些软件开发机构将研制力量的 40% 以上投入到软件测试之中; 对于某些性命攸关的软件, 其测试费用甚至高达所有其他软件工程阶段费用总和的 3 到 5 倍[1]。

尽管人们在软件开发过程中也采用形式化方法描述和证明软件规约[5], 并采用程序正确性证明[6]、模型检验[7] 等方法保证软件质量, 但是这些方法都存在一定的局限性[8], 尚未达到广泛实用阶段。因此, 程序代码最终体现了软件的质量, 无论是从软件开发方法学还是软件测试自身的效益看, 软件测试在今后较长时间内仍将是保证软件质量的重要手段。

1　软件测试的基本思想

下面从软件测试的技术与过程、持续的软件测试、软件测试的充分性准则等方面简要介绍软件测试的基本思想。

1.1　软件测试的技术与过程

如图 1 所示, 现有的软件测试技术通常分为静态测试和动态测试。静态测试是不执行程序代码而寻找程序代码中可能存在的缺陷或评估程序代码的过程。静态测试包括主要由人工进行的代码审查、代码走查、桌面检查以及主要由软件工具自动进行的静态分析。如果广义地理解, 静态测试还包括软件需求分析和设计阶段的技术评审[2]。

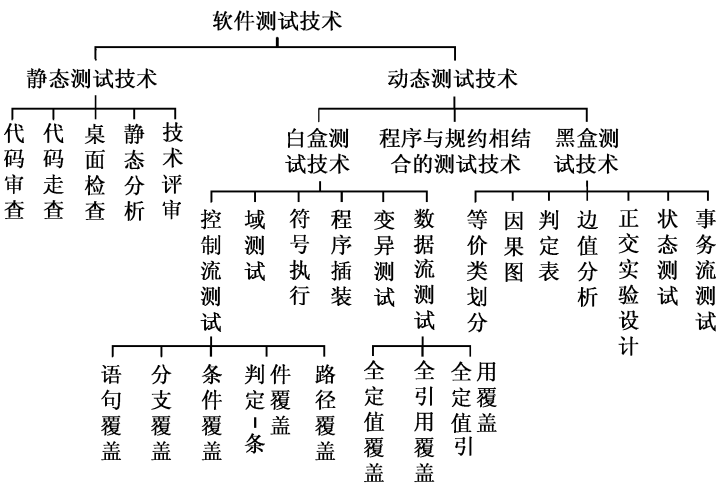


图 1　软件测试技术的分类

Fig. 1　Classification of the software testing technologies

动态测试通过在抽样测试数据上运行程序来检验程序的动态行为和运行结果以发现缺陷。动态测试包括生成测试用例、运行程序和验证程序的运行结果 3 部分核心内容, 以及文档编制、数据管理、操作规程及工具应用等辅助性工作。动态测试最重要的问题是生成测试用例的策略。它是动态测试有效、高效的关键。测试用例包括输入数据和期望结果。一般说到测试用例生成时, 由于期望结果构造的困难性, 都侧重或仅生成输入数据, 并称之为测试数据, 下面的讨论即按此约定。

按照生成测试数据所根据的信息来源,动态测试分为基于规约的测试(又称黑盒测试或功能测试)、基于程序的测试(又称白盒测试或结构测试)以及程序与规约相结合的测试。基于规约的测试是指测试人员无须了解程序的内部结构,直接根据程序输入和输出之间的关系或程序的需求规约来确定测试数据,推断测试结果的正确性。基于规约的测试包括:等价类划分、因果图、判定表、边值分析、正交实验设计、状态测试、事务流测试等。基于程序的测试是指测试人员根据程序的内部结构特性和与程序路径相关的数据特性设计测试数据。它包括控制流测试和数据流测试两类主要技术以及域测试、符号执行、程序插装和变异测试等其他技术。程序与规约相结合的测试则综合考虑软件的规范和程序的内部结构来生成测试数据[1,2,9,10]。

软件测试的过程分为单元测试、组装测试、确认测试、系统测试等几个阶段[1,2,9]。单元测试可以运用白盒测试(控制流、数据流测试)、黑盒测试(等价类划分、因果图、边值分析)等多种测试技术。组装测试主要采用黑盒测试中的等价类划分、边值分析,白盒测试中的数据流测试,域测试,调用对覆盖等测试技术。组装测试的策略是指进行单元组装的方法和步骤。组装测试的策略有渐增式组装和非渐增式组装两类,而前者又分为自底向上和自顶向下两种方式[1,2]。确认测试主要采用黑盒测试中的状态测试、事务流测试等测试技术[2]。

目前,面向对象的软件开发方法已被人们广泛接受,并且被基于UML(Unified Modeling Language)的建模工具以及C++、Java等程序设计语言所支持。与传统的软件开发方法相比,面向对象引入了类、对象、继承等新特征。面向对象中的继承、多态、动态绑定等机制对面向对象软件的测试产生了影响。以传统的软件开发方法为背景发展起来的测试技术,并不能完全适用于面向对象软件的测试[1,11—14]。从面向对象软件结构的角度出发,面向对象软件测试可分为类测试、类簇测试和系统测试[12,14—16]。其中类测试可分为3个部分:方法测试、基于状态的测试和基于状态响应的测试[12,17]。方法测试和系统测试分别与传统的单元测试和确认测试相对应。

1.2 持续的软件测试

软件测试是保障软件质量的重要手段,但它不是万能的,不能取代其他软件质量保障手段。完整的软件质量保障活动应该贯穿整个软件生存周期,包括评审、检查、审查、设计方法学和开发环境、文档编制、标准、规范、约定及度量、培训、管理等。软件质量需要综合运用包括软件测试在内的诸多手段才能得到最有力的保障。

完整的软件测试工作也应该贯穿整个软件生存周期,它有两方面的含义:(1)软件开发不同阶段都有软件测试工作;(2)软件测试工作的各个步骤分布在整个软件生存周期中[2]。

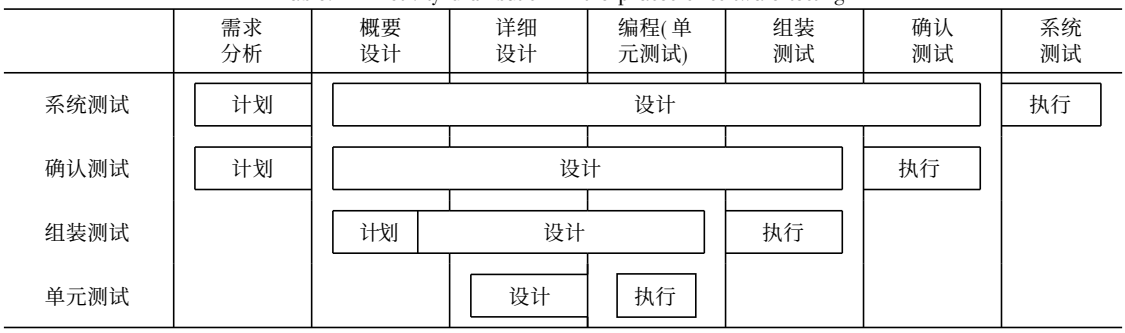
表1描述了软件测试各阶段工作在软件生存周期中的分布情况(表中从左往右各列存在时间由前往后的顺序性)。按照软件测试流程,将软件测试工作划分为计划(指进行测试计划)、设计(指进行测试设计)和执行(含评价,指执行测试并判别结果、评价测试效果和被测试软件)几个阶段。表1表明软件测试工作连续不断地在软件开发过程中进行。这体现了软件测试的一个原则:尽早开始软件测试工作,不断进行软件测试工作。

1.3 软件测试的充分性准则

Goodenough和Gerhart于1975年在研究软件测试能否保证软件的正确性时提出软件测试充分性的概念[18]。软件测试的充分性是根据被测软件在有限多个测试数据上的行为判断在所有测试数据上的行为的逻辑基础[10]。

表 1　软件测试各阶段工作的分布

Table. 1　Activity distribution in the phases of software testing



测试充分性准则是判定测试数据集对于被测程序是否充分的准则。如果测试数据集不充分, 就必须增加更多的测试数据, 否则可以结束当前测试工作。在文献中, 有许多软件测试的充分性准则, 以及对充分性准则的研究[10]。良好的软件测试充分性准则应该具有如下基本性质: 空集不充分性、有限性、单调性、非复合性、非分解性、非外延性、一般多重修改性、复杂性、回报递减律。

20 世纪 80 年代中期, 提出了充分性准则满足的 11 条公理。目前, 通常用测试覆盖准则度量测试充分性。到目前为止, 已经提出许多针对程序内部结构的测试覆盖准则, 主要包括控制流测试覆盖准则和数据流测试覆盖准则。控制流测试覆盖准则包括语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、路径覆盖等。数据流测试覆盖准则包括定值覆盖、引用覆盖、定值-引用覆盖等准则。这些准则不仅可以定量地规定软件测试需求, 指导测试数据的选择, 而且可以度量测试数据集揭示软件特定特征的能力, 对测试结果和软件可靠性评估具有重要影响[9,10,19]。

2　软件测试中的若干问题

本节讨论面向路径的测试数据的自动生成、测试预言(test oracle)、期望结果的自动生成和回归测试等问题。

2.1　面向路径的测试数据自动生成

软件测试在整个软件开发周期所占的比重很大。据统计, 在所有的软件测试的开销中, 约 40% 花费在设计测试用例上, 约 50% 花费在编写和编译测试脚本上, 另外约 10% 花费在测试脚本的执行和配置管理上[20]。

在软件测试中, 面向路径的测试数据生成问题(在本文中简称为 Q 问题) 描述为: 给定一个程序 P 和 P 中一条路径 W, 设 P 的输入空间为 D, 求 $\bar{x} \in D$, 使得 P 以 $\bar{x}$ 为输入运行, 所经过的路径为 W。

软件的单元测试中控制流测试中诸如语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、路径覆盖等问题和数据流测试中的全定值覆盖、全引用覆盖等问题[9,10], 以及组装测试中的调用对覆盖和数据流测试等问题可以归结为 Q 问题。面向断言的测试中的一些测试数据生成问题也可以归结为 Q 问题[20,21]。文献[22] 将回归测试中的一些测试数据生成问题也归结为 Q 问题。

自动求解 Q 问题将有效地减轻测试人员的劳动强度, 提高测试的效率和质量, 节省软件

开发的成本。根据估算,对于一个典型的大型软件项目,若能自动生成测试数据,则能节省整个软件开发费用的4%,相当于数百万美元[20]。

求解Q问题的实质在于约束系统的建立和求解。建立约束系统的困难是分析、化简路径W上的各种语句成分和各种数据类型,建立尽可能简洁的约束系统;求解约束系统的主要困难是处理可能存在的非线性约束。文献[23]的研究表明,不存在通用的有效的算法,对于任意的P和W,能生成使W被经过的输入数据。但是实际应用的需要迫使人们进行研究,并提出各种方法求解Q问题。

文献[8]将各种求解Q问题的方法分为随机法、静态法、动态法和试探法。静态法对程序进行静态分析和转换,不涉及程序的实际运行,它包括符号执行法[24]和文献[25]提出的方法等。动态方法则基于程序的实际运行,其生成测试数据的过程是确定性的,它包括直线式程序法[26]和文献[27,28,29]分别提出的方法等。试探法包括遗传算法[30]和模拟退火算法[31],这些方法虽然也需要实际运行程序,但是其生成测试数据的过程不完全是确定性的,而采用了概率论的思想。

文献[8]改进文献[29]提出的方法中构造和求解线性约束系统的方法。与原方法相比,改进后的方法不仅建立线性约束系统的效率更高,而且生成测试数据的能力更强,能够用于黑盒测试数据的自动生成。文献[8]以改进后的方法为核心算法,开发了一个面向路径的测试数据生成原型工具,并用实际的程序路径对该工具进行实验。初步的实验结果表明改进后的方法是比较有效的,不仅能够用于白盒、黑盒测试数据的自动生成,而且能够用于面向断言的测试数据自动生成和回归测试数据的自动生成。

2.2 测试预言、期望结果的自动生成

测试预言是一种检验待测系统在特定执行下是否正确运行的方法。期望结果用来确定测试用例执行的成功与否,它是程序根据输入应该得到的输出。因此,期望结果是一种比较理想的测试预言。自动生成期望结果不仅能有效地减轻测试人员的负担,而且能为不间断的持续测试提供有力的支持。但是现有的对自动生成期望结果的研究工作很少。文献[32]介绍了一种为自动化的黑盒测试生成期望结果的技术。该技术通过分析程序的输入-输出关系确定影响输出变量的输入变量集合,执行一个小规模的测试用例集并检验它们的输出结果的正确性。如果这个测试用例集执行结果正确,就可以自动生成更大的输入数据集合的期望结果。作者对这种技术进行了实验,根据检验384个测试用例的输出结果来自动生成大约600,000个测试用例的期望结果。结果表明该技术是比较有效的。

但是在有些情况下,某些程序的期望结果很难获得,即此时不能得到相应的测试预言。文献[33]研究了不需借助测试预言来对程序进行测试的技术。利用该技术可以对无法获得测试预言的程序进行测试。其原理是根据被测程序的性质,采用变形关系,即被测程序必要但不充分的条件,按照多组测试数据执行被测程序,检查变形关系是否能被满足。若变形关系不能被满足,则说明被测程序中肯定存在缺陷。例如对于计算余弦函数值的程序,可以利用变形关系$\cos(-x)=\cos(x)$来进行测试。取2个数值a,$-a$分别作为程序的输入计算$\cos(a)$和$\cos(-a)$,如果两次计算所得到的结果不相等则说明程序有缺陷。

2.3 回归测试

回归测试的目的是确认修改后的软件,以保证在以前测试过的代码中没有引入新的缺陷。

据统计, 回归测试占整个软件系统开销的 1/3[34]。已有的测试用例集是回归测试的基础。回归测试还要根据需要设计新的测试用例。针对已有的测试用例集, 回归测试主要有选择性重测和全部重测两种策略。为减少回归测试的开销, 在保证回归测试的质量的前提下, 应尽量减少回归测试时需求运行的测试用例数目。文献[35] 研究了如何简化测试用例集以满足相同的测试目标。文献[36] 则从测试过程、开销分析、层次性、支持工具等几个方面对工业环境中的回归测试进行了研究。虽然该文所讨论的是工业领域里的情况, 但是它同样适用于其他领域。

对于选择性重测的测试策略, 在选择哪些测试用例需要重新运行时要进行大量的分析, 代价很大。如果分析之后, 结果发现所有或者几乎所有的测试用例都被选中, 那么就根本就没必要去进行分析——简单地重新运行整个测试用例集同样有效甚至更有效。为此, 文献[3] 提出了一种能高效运算的预告方法, 对于一个给定的程序, 它能预告使用某些类型的选择性回归测试技术的代价- 效力关系。其思想是假如能很快就判定很可能重新运行回归测试集里很大一部分, 那么, 简单地重新运行整个测试用例集就很可能是代价有效的, 从而节省了分析的开销。

3　软件测试的发展趋势

下面探讨软件测试一些新的研究动向和发展趋势。

3.1　软件的易测试性设计

Voas 将软件的易测试性定义为一定测试策略迫使软件中存在的故障被暴露的概率[37]。软件的易测试性包括可观察、可控制、可理解等几个方面[38]。软件易测试性设计即是在软件的设计和编码中考虑测试问题, 它借鉴了硬件易测试性设计的思想。为了减少测试集成电路的测试开销, 提高产品的质量, 工程师采用易测试性设计技术, 即在集成电路上增加额外的引脚, 通过这些引脚能够在测试时探测集成电路的内部, 提高了可控制性和可观察性。内建式测试(Bulit- In Testing) 方法是一种易测试性设计技术, 它在集成电路上引入测试电路或引脚, 使得集成电路能够工作在测试模式下, 并且传输测试输入, 捕获输出。这种方法的进一步扩展就是再增加测试输入生成电路, 从而避免外界的输入。这就是内建式自测试概念[38]。

软件易测试性设计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[39]。软件易测试性设计符合软件测试的一个原则: 尽早开始软件测试工作, 不断进行软件测试工作。软件易测试性设计体现软件测试的如下观点: 软件产品的质量是生产(包括分析、设计、编码、测试) 出来的, 而不是仅仅依靠软件测试来保障。软件易测试性设计也体现了软件测试的一个发展趋势: 向软件开发的前期发展, 与软件开发的设计和编码阶段相融合。易于测试的软件本身所包含的缺陷也会减少[2]。软件易测试性设计将有效地提高软件测试的效率和质量, 提高软件设计和编程的质量, 进而提高软件产品的质量。软件的易测试性设计方法包括合约式设计(Design by Contract) 、内, 建式测试和内建式自测试等几种方法。

合约是管理对象之间的交互的一组规则[40]。合约的来源是软件的规约, 它指明了软件"做什么"而不涉及"怎样做"。常见的合约类型包括: 前置条件、后置条件、不变式、循环变式/不变式和轨迹等。合约表明了过程调用方(客户方) 与实现方相互的职责和利益: 客户方只有在满足前置条件的条件下才能调用对应的过程; 实现方承诺当过程结束时, 后置条件指明的工作将被完成, 并且不变式仍然满足。合约可用于区分软件失效时的责任: 如果前置条件被违

反,则应该在客户方寻找错误;如果后置条件或不变式被违反,责任在实现方。合约有助于减少冗余的检查代码,提高软件设计的效率和运行的性能;利用自动检查合约的工具,能够减轻用户的负担,减少用户犯错误的机会;并且合约被违反时引发异常,便于就近定位故障。常见的合约式设计方法是在程序代码的注释中提供软件的合约。现在有一些对合约进行自动检查的工具,如针对 Java 语言的 iContract, Jass, JMSAssert 等工具。

软件的内建式测试方法是在程序中添加额外的测试机制,使软件能够工作在测试模式下。软件的内建式自测试方法就是在此基础上再增加测试用例生成机制[38]。

3.2 构件测试

当前社会的信息化过程对软件的开发方法与生产能力提出了新的要求,软件复用是提高软件产品质量与生产效率的关键技术。软件构件概念的提出为软件复用提供了技术基础[41]。

构件的高可靠性是构件能被成功复用的前提。构件测试是保障和提高构件的可靠性的重要手段。构件的开发者和复用者必须对构件进行充分的测试,以确保它在新的环境中工作正常。

与传统的软件测试相比,构件测试有着自身的固有特点[42]:(1) 不能对构件的执行环境和用户的使用模式进行完全准确的预测,故构件开发者不能完全、彻底地对构件进行测试,并且很难确定何时结束测试;(2) 构件复用者和第三方测试人员通常无法得到构件的源代码及详细的设计知识,通常只能对构件进行黑盒测试,即调用构件的方法后只能通过观察执行的结果判断构件的行为是否正确,无法检查执行过程中的构件的内部状态,使得构件执行过程中的一些故障被隐藏。这些困难对构件测试提出了严峻的挑战。

国际上于20世纪90年代后期对构件测试开展了研究[43,44]。近年来,出现了大量的文献报道[38,42,45—54]。构件测试成为当前软件工程学术界和工业界的热点问题之一。大体上,对构件的测试可以从以下几个方面来进行分类。从构件测试的内容可分为:构件内部实现细节的测试[38],构件接口的测试[43—48,53,54],构件组装(构架)的测试[46,51]。从测试者与构件的关系可分为:构件开发者的测试(拥有构件的源代码)[38,45,53]、构件复用者和第三方的测试(没有源代码)[42,43,52]。从测试过程中所采用的技术手段可分为:基于变异测试的方法[38,53,54],基于构件状态机的方法[50],对构件的回归测试[52],以及构件的易测试性设计[38,44,48,49]等。

构件测试一个重要的发展方向是基于合约的构件易测试性设计。合约可以在运行时被检查,便于捕获构件执行过程中的一些故障,提高构件的易测试性。因此,基于合约的构件易测试性设计不仅为构件开发者开发高质量的构件提供帮助,而且在构件开发者与复用者之间架起一座桥梁,为构件复用者的测试提供支持,也为构件开发者捕获错误提供便利,便于区分构件开发者与复用者的责任。如果众多的构件开发者都采用合约式设计方法生产构件,那么失效时很容易定位到构件和其中的方法。为使基于合约的构件易测试性设计方法能够实用,需要研究解决以下问题:构件合约的描述、表达,构件中合约的获取,对构件合约的自动检查,以及针对构件合约的软件测试。

3.3 Web Services 测试

随着软件产业模式从以产品为中心的制造业转变为以客户为中心的服务业,WWW 从2层体系转变为3层体系,B2B 从复杂专用的连接转变为简单通用的连接,分布计算中间件从 Intranet 扩展到 Internet,CORBA、COM 及 EJB 等中间件技术已不能适应这些发展需求,因而导致

了新型中间件技术 Web Services 的产生[55]。

当前, Web Services 越来越受到人们的关注, 它使用了包括 SOAP、WSDL 和 UDDI 在内的标准协议。这些标准协议体现了互操作性, 并用于应用的开发及运行时 Web Services 的选择和调用。Web Services 的测试和评估对服务提供者和服务使用者来说都是相当重要的。Web Services 的测试相当于黑盒测试, 可以获得规约, 却不能得到设计或源代码。Web Services 规约是用 WSDL 写的, 而代码是用 Java、C# 或其他编程语言写的。

当前的 WSDL 信息包括输入/输出的数量、每个输入和输出的变量类型、输入的顺序和输出返回的顺序和一个 Web Service 应当如何调用的信息。但是, 为了执行 Web Services 的黑盒测试和回归测试, 以上信息是远远不够的。文献[56] 从输入和输出的依赖关系、调用顺序、层次功能描述和并发顺序规范四个方面对 WSDL 进行了扩展, 提高了 Web Services 的易测试性。文献[57] 则提出了一个支持测试执行和测试脚本管理的测试框架。

对用户而言, 通常需要合并多个 Web Services 来满足他们的需求, 所谓的服务聚合就是根据用户的要求合并现存的 Web Services。这时需要一种标准语言来描述不同的 Web Services 是如何集成在一起的, 即描述 Web Services 流的语言。当前有两种常见的 Web Services 流描述语言 WSFL 和 XLANG。如果让用户自己来描述 Web Services 流, 很可能会导致错误。在发现错误之前, 流中的许多 Web Services 已经被调用了, 其后果会导致事务回滚困难、引起网络拥塞, 从而造成资源浪费。文献[58] 和[59] 对 Web Services 流的测试进行了研究。

4　软件测试综述论文的比较

经过多年的发展, 一些基本的软件测试技术已经渐趋成熟。在此过程中, 人们不断进行总结。文献[60] 从软件测试的充分性角度比较全面地综述了已有文献中的各种软件测试技术, 包括结构测试技术、基于故障的测试技术以及基于错误的测试技术, 并讨论了比较软件测试充分性准则的方法, 以及软件测试充分性准则的公理化研究和评估方法。但是该文献主要集中在模块单元级, 并且由于受当时软件开发方法等因素的限制, 该文献没有涉及面向对象软件的测试和基于构件的软件测试技术。

文献[14] 分析了面向对象程序设计语言的特点及其对软件测试的影响, 从基于规约的测试方法和基于程序的测试方法以及测试方法与软件开发过程的集成途径等方面, 综述面向对象软件测试领域有代表性的研究工作。

文献[61] 分析和介绍软件构件和构件化软件, 以及伴随而来的对软件测试的挑战, 并讨论构件的易测试性问题; 然后介绍软件构件的验证方法和构件化软件的测试方法, 包括组装测试、回归测试和性能度量; 最后探讨软件构件和构件化软件的质量保证方法。该文献也将构件的易测试性作为解决构件测试问题的一个重要方面。但是由于构件测试研究的时间较短, 目前尚不成熟, 故该文献也只是将现有研究工作进行综述, 并进行相应探讨, 而无法给出彻底解决软件构件和构件化软件的测试问题的方案。

本文从软件测试的技术与过程、持续的软件测试、软件测试的充分性准则等方面简要介绍软件测试的基本思想; 讨论软件测试中面向路径的测试数据自动生成、测试预言、期望结果的自动生成、回归测试等关键问题; 并且探讨当前软件测试一些新的研究动向和发展趋势, 包括构件测试、软件的易测试性与基于合约的构件易测试性设计和 Web Services 测试等。

5 结束语

软件测试是一项费时、费力并且单调乏味的活动,测试人员需要设计、执行、分析大量的测试用例。软件测试的自动化将有效地减轻测试人员的劳动强度,提高测试的效率和质量,从而节省软件开发的成本,提高软件的质量。

虽然软件测试在软件质量保证中正发挥着越来越重要的作用,但是,目前在有些单位或部门软件测试仍然没有得到足够的重视,主要有两个原因:(1) 任务或市场的压力大。软件产品一般都有一个最后的发布期限。如果在软件开发的前期由于各方面的原因造成了工期的延误(这种现象在软件开发中很普遍),并且软件产品的发布期限不容更改(即所谓的"后墙不倒"),那么只能缩短软件测试的时间,其结果是牺牲了软件产品的质量。这其实是由于错误地预计了软件开发的进度所造成的,需要更加准确地预计软件开发的进度;(2) 软件测试的实施效果不很理想,表现为投入大,回报低。这主要是由于当前软件测试的自动化程度偏低,现有的软件测试技术不能满足当前软件开发的要求,需要研究更加有效的软件测试技术,并且加速把先进、有效的技术从实验室转化为工业界实用的方法的进程,开发更多、更好的支持软件测试自动化的产品,为测试人员提供帮助。

随着软件技术的不断向前发展,构件、Web Services 等新技术的应用为软件测试带来新的问题和挑战,也为软件测试的发展带来新的机遇。软件测试技术自身的不断发展对软件开发方法学将产生影响。随着软件易测试性概念的提出和研究的不断深入,软件的易测试性将成为衡量软件质量的一项指标。软件易测试性分析技术将为度量软件的易测试性,进而为改进和提高软件测试的过程乃至软件开发的过程提供帮助。软件测试目前呈现向软件开发的前期发展、与软件开发的设计阶段和编码阶段相融合的趋势。软件易测试性设计技术将帮助软件开发者在软件中嵌入测试信息,开发具有自测试能力,并且能够向外界提供相应测试信息的软件实体(如构件),为解决基于构件、Web Services 等新技术的软件开发方法所带来的新问题提供有前途的解决办法。

参考文献

1 齐治昌, 谭庆平, 宁洪. 软件工程. 北京: 高等教育出版社, 2001

2 周涛. 航天型号软件测试. 北京: 宇航出版社, 1999

3 Weyuker E J. Evaluation Techniques for Improving the Quality of Very Large Software Systems in a Cost-Effective Way. The Journal of Systems and Software, 1999, 47: 97-103

4 蔡开元. 软件可靠性工程基础. 北京: 清华大学出版社, 1995

5 King S, Hammond J, Chapman R, et al. Is Proof More Cost-Effective Than Testing? IEEE Transactions on Software Engineering, 2000, 26(8): 675-686

6 陈火旺, 罗朝晖, 马庆鸣. 程序设计方法学基础. 长沙: 湖南科学技术出版社, 1987

7 Clarke E M, Grumberg O, Peled D. Model Checking. Cambridge, Massachusetts, London, England: The MIT Press, 1999

8 单锦辉. 面向路径的测试数据自动生成方法研究: [学位论文]. 长沙: 国防科学技术大学研究生院, 2002

9 郑人杰. 计算机软件测试技术. 北京: 清华大学出版社, 1992

10 朱鸿, 金凌紫. 软件质量保障与测试. 北京: 科学出版社, 1997

11 Kung D, Gao J, Hsia P, et al. Developing an Object-Oriented Software Testing and Maintenance Environment. CACM, 1995, 38(10): 75-87

12 赵元聪, 朱三元. 面向对象软件测试的认识. 计算机应用与软件, 1996, 13(3): 1—4, 46

13 邵维忠, 王立福, 梅宏, 等. 面向对象的软件测试——方法研究及系统设计. 见: 杨芙清, 何新贵主编. 软件工程进展. 北京: 清华大学出版社, 1996. 115—122

14 金凌紫. 面向对象软件测试技术进展. 计算机研究与发展, 1998, 35(1): 6—13

15 Murphy G C, Townsend G, Wong S W. Experience With Cluster and Class Testing. CACM, 1994, 37(9): 39-47

16 Chen H Y, Tse T H, Chen T Y. TACCLE: A Methodology for Object-Oriented Software Testing at the Class and Cluster Levels. ACM Transactions on Software Engineering and Methodology, 2001, 10(4): 56-109

17 Kung D, Suchak N, Gao J, et al. On Object State Testing. In: Proceedings of COMPSAC'94, 1994. 222-227

18 Goodenough J B, Gerhart S L. Toward a Theory of Test Data Selection. IEEE Transactions on Software Engineering, 1975, SE-3 (6): 156-173

19 李留英. UML 测试技术的研究与实现: [学位论文]. 长沙: 国防科学技术大学研究生院, 2000

20 Tracey N J. A Search-Based Automated Test-Data Generation Framework for Safety-Critical Software: [PhD thesis]. Department of Computer Science, University of York, 2000

21 Korel B, Al-Yami A M. Assertion-Oriented Automated Test Data Generation. In: Proceedings of the 18th International Conference on Software Engineering, Berlin: 1996. 71-80

22 Korel B, Al-Yami A M. Automated Regression Test Generation. In: Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Beach: 1998. 143-152

23 Weyuker E J. The Applicability of Program Schema Results to Programs. International Journal of Computer Information Sciences, 1979, 8(5): 387-403

24 Ramamoorthy C V, Ho S-B F, Chen W T. On the Automated Generation of Program Test Data. IEEE Transactions on Software Engineering, 1976, SE-2 (4): 293-300

25 王言志, 刘椿年. 区间算术在软件测试中的应用. 软件学报, 1998, 9(6): 438—443

26 Miller W, Spooner D L. Automatic Generation of Floating-Point Test Data. IEEE Transactions on Software Engineering, 1976, SE-2 (3): 223-226

27 Korel B. Automated Software Test Data Generation. IEEE Transactions on Software Engineering, 1990, 16(8): 870-879

28 Gallagher M, Narasimhan V L. ADTEST: A Test Data Generation Suite for Ada Software Systems. IEEE Transactions on Software Engineering, 1997, 23(8): 473-484

29 Gupta N, Mathur A P, Soffa M L. Automated Test Data Generation using an Iterative Relaxation Method. In: Proceedings of the ACM SIGSOFT 6th International Symposium on the Foundations of Software Engineering, Orlando: 1998. 231-244

30 Michael C C, McGraw G, Schatz M A. Generating Software Test Data by Evolution. IEEE Transactions on Software Engineering, 2001, 27(12): 1 085-1 110

31 Tracey N, Clark J, Mander K. Automated Program Flaw Finding using Simulated Annealing. In: Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Beach: 1998. 73-81

32 Schroeder P J, Faherty P, Korel B. Generating Expected Results for Automated Black-box Testing. In: Proceedings of the 17th International Conference on Automated Software Engineering, Alamitons, 2002. 139-148

33 Chen T Y, Tse T H, Zhou Z Q. Fault-Based Testing without the Need of Oracles. Information and Software Technology, 2003, 45(1): 1-9

34 Harrold M J. Testing: A Roadmap. In: Finkelstein A eds. The Future of Software Engineering. New York: ACM Press, 2000

35 Chen T Y, Lau M F. A New Heuristic for Test Suite Reduction. Information and Software Technology, 1998, 40: 347-354

36 Onoma A K, Tsai W-T, Poonawala M H, et al. Regression Testing in an Industrial Environment. CACM, 1998, 41(5): 81-86

37 Voas J M. Software Testability Measurement for Assertion Placement and Fault Localization. In: Proceedings of the Automated and Algorithmic Debugging, Saint-Malo: 1995. 133-144

38 Martins E, Toyota C M, Yanagawa R L. Constructing Self-Testable Software Components. In: Proceedings of the International Conference on Dependable Systems and Networks, Goteborg: 2001. 151-160

39 宫云战, 刘海燕, 万琳, 等. 软件测试性的分析与设计技术研究. 2000年全国测试学术会议(CTC'2000). 北京: 2000, 271—274

40 Meyer B. Object-Oriented Software Construction. New Jersey: Prentice Hall, 1997

41 杨芙清, 王千祥, 梅宏, 等. 基于复用的软件生产技术. 中国科学(E辑), 2001, 31(4): 363—371

42 Ma Y-S, Oh S-U, Bae D-H, et al. Framework for Third Party Testing of Component Software. In: Proceedings of the 8th Asia-Pacific Software Engineering Conference, Macau SAR: 2001. 431-434

43 Voas J M. Certifying Off-the-shelf Software Components. IEEE Computer, 1998, 31(6): 53-59

44 Edwards S H, Shakir G, Sitaraman M, et al. A Framework for Detecting Interface Violations in Component-Based Software. In: Proceedings of the 5th International Conference on Software Reuse, British Columbia: 1998. 46-55

45 Yoon H, Choi B. Inter-Class Test Technique between Black-Box-Class and White-Box-Class for Component Customization Failures. In: Proceedings of the 6th Asia-Pacific Software Engineering Conference, Takamatsu: 1999. 162-165

46 Vitharana P, Jain H. Research Issues in Testing Business Components. Information & Management, 2000, 37(6): 297-309

47 Wu Y, Pan D, Chen M-H. Techniques for Testing Component-Based Software. In: Proceedings of the 7th International Conference on Engineering of Complex Computer Systems, Los Alamitos: 2001. 222-232

48 Edwards S H. A Framework for Practical, Automated Black-Box Testing of Component-Based Software. Software Testing, Verification and Reliability, 2001, 11(2): 97-111

49 Deveaux D, Frison P, Jezequel J-M. Increase Software Trustability with Self-testable Classes in Java. In: Proceedings of the 2001 Australian Software Engineering Conference. Canberra: 3-11

50 Beydeda S, Gruhn V. An Integrated Testing Technique for Component-Based Software. In: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Beirut: 2001. 328-334

51 Jin Z, Offutt J. Deriving Tests from Software Architectures. In: Proceedings of the 12th International Symposium on Software Reliability Engineering, Hong Kong: 2001. 308-313

52 Orso A, Harrold MJ, Rosenblum D, et al. Using Component Metacontent to Support the Regression Testing of Component-Based Software. In: Proceedings of the International Conference on Software Maintenance, Florence: 2001. 716-725

53 Yoon H, Choi B. An Effective Testing Technique for Component Composition in EJBs. In: Proceedings of the 8th Asia-Pacific Software Engineering Conference, Macau SAR: 2001. 229-236

54 Lee S C, Offutt J. Generating Test Cases for XML-Based Web Component Interactions using Mutation Analysis. In: Proceedings of the 12th International Symposium on Software Reliability Engineering, Hong Kong: 2001. 200-209

55 杨芙清, 梅宏, 吕建, 等. 浅论软件技术发展. 电子学报, 2002, 30(12A): 1 901—1 906

56 Tsai W T, Paul R, Wang Y, et al. Extending WSDL to Facilitate Web Services Testing. In: Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering, Tokyo: 2002. 171-172

57 Tsai W T, Paul R, Song W, et al. Coyote: An XML-based Framework for Web Services Testing. In: Proceedings of the 7th International Symposium on High Assurance Systems Engineering, Tokyo: 2002. 173-174

58 Nakajima S. On Verifying Web Service Flows. In: Proceedings of the Symposium on Applications and the Internet Workshops, Nara: 2002. 223-224

59 Nakajima S. Verification of Web Service Flows with Model-Checking Techniques. In: Proceedings of the 1st International Symposium on Cyber Worlds, Tokyo: 2002. 378-385

60 Zhu H, Hall P A V, May J H R. Software Unit Test Coverage and Adequacy. ACM Computing Surveys, 1997, 29(4): 366-427

61 Gao J Z, Tsao H S J, Wu Y. Testing and Quality Assurance for Component-Based Software. Boston: Artech House, 2003

Research Progress in Software Testing

SHAN Jinhui[1),2)] *JIANG Ying*[1),3)] *SUN Ping*[2)]

([1)] Software Institute, Peking University, Beijing, 100871, E-mail: {shanjh, jiangy}@sei.pku.edu.cn;

[2)] Jiuquan Satellite Launch Center;

[3)] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650093)

Abstract Software testing is one of the most important techniques used to assure the quality of software products. Software testing keeps a high proportion during the whole software life cycle. Gradually, software testing develops towards the former phase of software development and inosculates with the design and coding phases of software development. This paper introduces the basic ideas of software testing from the techniques and the process of software testing, persistent software testing, adequate criteria of software testing and so on. Then, this paper discusses some problems in software testing, including automated path-wise test data generation, test oracle, automated expected results generation, regression testing and so on. At last, this paper explores the development tendency of software testing, such as component testing, software testability, design by contract for testability in components and Web Services testing.

Key words software testing; software quality; testability of software; component testing