ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.9, September 2010, pp.2135−2147
doi: 10.3724/SP.J.1001.2010.03659

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于多维度覆盖率的软件测试动态评价方法*

安金霞[1+], 王国庆[2], 李树芳[1], 朱纪洪[1]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](中国航空无线电电子研究所,上海 200233)

Dynamic Evaluation Method Based Multi-Dimensional Test Coverage for Software Testing

AN Jin-Xia[1+], WANG Guo-Qing[2], LI Shu-Fang[1], ZHU Ji-Hong[1]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](China National Aeronautical Radio Electronics Research Institute, Shanghai 200233, China)

+ Corresponding author: E-mail: ajx06@mails.tsinghua.edu.cn

An JX, Wang GQ, Li SF, Zhu JH. Dynamic evaluation method based multi-dimensional test coverage for software testing. *Journal of Software*, 2010,21(9):2135−2147. http://www.jos.org.cn/1000-9825/3659.htm

Abstract: As the size and complexity of mission-critical application software continues to grow, the cost of Software Testing (ST) is also increasing. The numerous methods and processes used to evaluate ST dynamically and quantitatively to improve testing efficiency serve as practice problems in the ST field. Based on multi-dimensional test coverage models, this paper proposes a dynamic evaluation method for ST and discusses it from the viewpoint of testing monitoring information, dynamic analysis and evaluation models, and testing optimized strategy. Furthermore, a concept, Synthetic Test Coverage (STC), is defined in this paper, and its empirical formulas are also presented. Examples show that the methods are useful in helping ST evaluation groups to track and control the effects of ST and for improving the user's ability to observe and control the ST process.

Key words: software testing; multi-dimensional; test coverage; dynamic; testing optimized strategy

摘 要: 随着关键领域应用软件规模和复杂性的不断提高,软件测试成本也不断增加.如何动态、定量地评价软件测试情况,提高测试效率是软件测试领域面临的现实问题.提出了一种基于多维度测试覆盖率的软件测试动态评价方法,并从测试监测信息、动态分析和评价模型、测试优化策略几个方面展开讨论.给出了综合测试覆盖率的定义和经验公式.实例显示,该方法有助于软件项目评测人员动态跟踪和定量监控软件测试效果,提高软件测试过程的可观察性和可控制性.

关键词: 软件测试;多维度;测试覆盖率;动态;测试优化策略

中图法分类号: TP311 **文献标识码**: A

航空航天等领域软件的规模和复杂性在不断提高,作为有效保证和验证软件质量的重要环节和依据,软件测试已逐渐成为软件研发成本最高的阶段.如何动态跟踪和定量分析软件测试能力、测试效率,分析测试薄弱点,持续优化软件测试是目前软件工程项目中迫切需要研究的现实问题.

目前,关于软件测试方面的专著[1−4]侧重于论述如何开展测试,包括测试过程组织、管理和具体的测试方法

Received 2008-11-12; Revised 2009-03-05; Accepted 2009-05-21

等,测试评价主要从软件缺陷跟踪、缺陷报告及测试进度评价等方面论述,缺少对软件测试本身是否科学、有效的评价方法.对测试用例集、测试方法本身的测试能力、效率及薄弱点等缺少动态跟踪和定量的在线评价方法.因此,实际软件测试工程中,常面临如下问题:虽然测试过程按照测试计划开展,但由于测试用例集庞大,对测试执行的有效性缺少在线的定量评价,因此很难在测试执行过程中及时发现测试薄弱点,提出测试优化策略,并有效控制和提高测试效率.

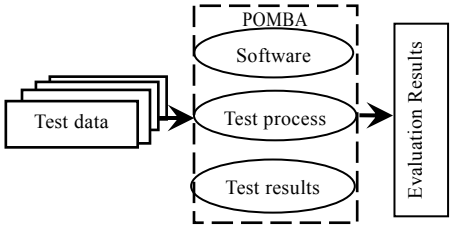

Fig.1 Framework of POMBA method
图 1 POMBA 方法框架

2007 年第 7 届软件质量国际会议上,Mayer 等人[5]提到了第 1 个国际软件测试评价组织(First International Workshop on Software Test Evaluation Committees,STEV 2007)成立的相关消息,并指出软件测试评估一直受到忽视.文献中提到了几个软件测试评估关注的问题,主要侧重于对测试数据库方面评估方法的研究.如针对测试用例集的合理性分析.Chernak[6]强调了对测试用例有效性的在线评价,定义了一种测试用例有效性的度量方法,认为测试用例发现的软件缺陷越多,测试越有效.但文献中未考虑测试的其他定量监测信息,如测试覆盖率.He[7]等人基于软件测试成熟度模型 TMM(the test maturity model)思想,提出了如图 1 所示的一种基于度量的面向过程软件测试评估方法 POMBA(process-oriented metrics-based assessment),讨论的度量指标主要有被测软件的规模、复杂度、测试密度和用例数等.

另一方面,测试覆盖率是定量度量和控制软件测试过程的重要手段之一,通过分析目前测试覆盖率的应用,其还存在以下不足:

a) 基于测试覆盖率的度量方法主要是对各种覆盖率分别进行统计[8,9],度量软件测试充分程度,以满足相应的测试充分性准则为目标.如何综合利用多种测试覆盖率的动态变化特征,对各种测试方法和测试用例本身的测试效果及测试薄弱点进行动态跟踪和定量评价的相关研究则很少.

b) 目前提出了语句覆盖、分支覆盖、条件覆盖和 c-use 覆盖等 10 余种测试覆盖度量指标,测试覆盖率不同,反映软件测试程度的角度也不同,各有优缺点[4,8,9].但在工程应用中,由于测试时间和成本的约束,常常依据少数几个测试充分性准则,分别统计相应的几种测试覆盖率,并独立地评价软件测试的充分程度.因此,如何充分利用所有能够统计的各种覆盖率数据及其特点,给出软件测试充分性的综合度量方法是一个有意义的研究.

针对上述分析,本文提出一种基于多维度测试覆盖率的软件测试动态评价方法,并从测试监测信息、动态分析和评价模型、测试优化策略几方面阐述如何实现,同时通过多个实例给出了方法的应用情况.

1 软件测试动态评价方法框架

如图 2 所示,软件测试指依据测试用例集中的测试用例,执行测试,并输出测试执行结果.本文将软件测试动态评价定义为:实时跟踪软件测试,收集测试监视信息,从软件测试能力、测试效率和测试薄弱点等方面对软件测试进行动态分析和评价,并依据评价结果及时提出测试优化策略,指导软件测试的持续优化.与软件测试成熟度模型(TMM)[10]、软件测试控制论[11]等中的“软件测试过程”不同的是,这里不考虑软件测试过程中的人员、组织和管理相关因素,主要对软件测试执行阶段中测试的执行结果进行在线、定量的监测和评价.下面几节内容分别从测试监测信息、动态分析和评价模型、测试优化策略几方面阐述基于多维度测试覆盖率的软件测试动态评价方法.

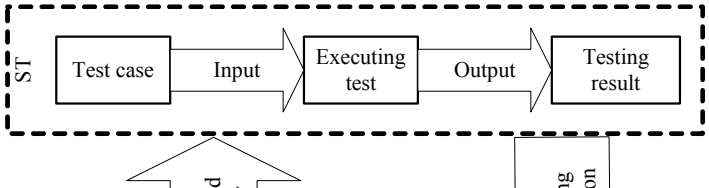

Fig.2 Framework of dynamic evaluation method for ST

图 2 软件测试动态评价方法框架

2 软件测试监测信息

2.1 累积测试用例数n_{case}和测试执行时间t

n_{case}是指达到当前测试覆盖率执行的测试用例总数,t表示执行这n_{case}个测试用例累积占用的CPU时间(以下同).

2.2 多维度测试覆盖率

1) 多维度测试覆盖率的含义

文献[4]指出,测试覆盖率是用来度量软件测试充分程度的重要手段,可通过比率公式(1)来表示:

$$覆盖率=\left(\frac{至少被执行一次的item}{item的总数}\right)\times 100\% \tag{1}$$

公式(1)中,假设要对*item*的覆盖情况进行计算.为了从不同角度度量软件测试的充分程度,提出了侧重不同角度的 10 余种测试覆盖率指标,主要包括:基于代码的覆盖率(如语句覆盖、分支覆盖、路径覆盖、分支-条件覆盖、组合覆盖、指令覆盖、输入输出域覆盖、块覆盖)、基于需求的覆盖率(如功能覆盖、需求覆盖、接口覆盖)和面向对象的覆盖率(如继承上下文覆盖、基于状态的上下文覆盖、基于线程的上下文覆盖)等.本文将每种覆盖率称作软件测试的一个维度覆盖率,多维度测试覆盖率表示用来度量测试充分程度的多种测试覆盖率.

2) 多维度测试覆盖率模型

要利用多维度测试覆盖率实时度量软件测试情况,首先需要提取其特征属性和度量参数,建立其度量模型.本文从以下几个方面加以讨论.

假设测试覆盖率的维度为m,即有m种测试覆盖率$C_1,C_2,\ldots,C_m$,且$0\leqslant C_j\leqslant 1,j=1,\ldots,m$.$n_{case}$和$t$均可作为测试覆盖率的统计自变量,因此用一个变量$x$表示,即以下$x$可取$n_{case}$或$t$.

a) 各维度测试覆盖率的期望值$\hat{C}_j$

由于测试时间和成本限制,一般测试不可能使所有的覆盖率都达到 100%,工程中依据软件安全关键等级(如航空航天安全等级为 A、B 级的软件测试覆盖率要求相对 C、D 级软件要高)、实现测试覆盖率的难易程度(如路径覆盖率很难达 100%)、被测软件代码和功能重要性(如对核心模块的测试覆盖率要求更高)、测试阶段(如单元测试对代码覆盖率要求较高,在集成测试和系统测试阶段对需求覆盖率要求较高)、开发方法(如面向对象的软件开发方法强调对面向对象的测试覆盖)等因素,提出不同的测试覆盖率期望值$\hat{C}_j$,并且$0\leqslant\hat{C}_j\leqslant 1$,$j=1,\ldots,m$.如语句覆盖率的期望值常设为 90%~100%;由于路径覆盖存在路径数呈指数增长限制,覆盖率很难达到 100%,因此常设为 70%~80%左右,甚至更低.

b) 测试覆盖率的类型 *Type*

基于代码和基于需求两类测试覆盖所处的主要测试阶段和采用的测试方法差别较大,不同测试阶段对各种测试覆盖率的要求和度量的侧重点也不一样,因此将多维度测试覆盖率分为基于代码和基于需求两类讨论.

c) 测试覆盖率满意度 *Sat*

定义 1. 软件测试中,当执行完第 n_{case} 个测试用例后,为了表示实际实现的测试覆盖率达到其期望值的程度,定义测试覆盖率 C_j 当前的满意度 $Sat_j(x)$为

$$Sat_j(x)=(C_j(x)/\hat{C}_j)\times 100\% \tag{2}$$

其中,j=1,…,m;$C_j(x)$为测试覆盖率 C_j 的当前值.

由于实测覆盖率值可能超过期望值,例如,某被测软件分支覆盖率期望值设为 75%,但实际达到了 76%,此时有 $Sat_j(x)>1$,因此规定:

$$\text{if } Sat_j(x)\geqslant 1, \text{ then } Sat_j(x)=1, \text{即 } 0\leqslant Sat_j(x)\leqslant 1.$$

举例见表 1.

Table 1 Illustration for satisfaction degree of test coverage

表 1 测试覆盖率满意度示例

	Current value $C_j(i)$	Expected value $\hat{C}_j$	Satisfaction degree
Statement coverage (%)	55	100	55
Branch coverage (%)	45	75	60

表 1 中,虽然当前分支覆盖率低于语句覆盖率,但由于期望值不同,其满意度反而高.利用满意度可动态了解各种测试覆盖率达到的满意程度,为优化测试策略提供依据.

d) 各维度测试覆盖率的优先级因子ω_j

由于多维度测试覆盖中,各种覆盖反映的测试角度不同、实现的难易程度不同、期望目标和对软件可靠性的影响程度不同,故测试中对不同测试覆盖给予的重视程度也就不同.由此引入"优先级因子"作为各维度测试覆盖率的权重值,即 $\omega_1,...,\omega_j,...,\omega_m$,且取值须满足约束条件:$0\leqslant\omega_j\leqslant 1, \sum_{j=1}^{m}\omega_j=1$.并且规定优先级越高,$\omega_j$ 的值越大.测试中希望优先实现优先级高的测试覆盖.

e) 软件模块关键性测试影响因子θ_k

规模大、复杂性较高的软件常由很多模块组成(这里,模块可指软件配置项、软件构件、类、功能组件等可以独立度量的软件组成体),且关键性级别不同.软件测试中,往往对关键级别较高的核心模块提出的覆盖率要求更高.为了科学评价软件测试的充分性,动态分析和评价各软件模块及整个软件测试情况,并在测试时间和成本约束条件下,优先使关键级别高的软件模块得到更加充分的测试,引入了软件模块关键性相关测试影响因子θ_k.

假设根据软件体系结构和组成模块的关键程度,将软件划分为关键级别不尽相同的 n 个模块,其对应的测试影响因子为 $\theta_1,...,\theta_k,...,\theta_n$,则$\theta_k$ 的取值须满足以下约束条件:

$$0\leqslant\theta_k\leqslant 1,\ \sum_{k=1}^{n}\theta_k=1,\ k=1,2,...,n \tag{3}$$

并且被测模块的关键级别越高,θ_k 值越大.

为了进一步讨论θ_k 的取值方法,下面举例说明如何给出 $\theta_1,...,\theta_k,...,\theta_n$ 的值.

假设将被测软件划分成 z 个模块,对应的关键级别为核心模块(z_1 个模块)、一般模块(z_2 个模块)和辅助模块(z_3 个模块)这 3 个级别,其中 $z_1+z_2+z_3=z$.

设 z 个模块对应的测试影响因子为 $\theta_1,...,\theta_{z_1},\theta_{z_1+1},...,\theta_{z_1+z_2},\theta_{z_1+z_2+1},...,\theta_{z_1+z_2+z_3-1},\theta_z$,易知:

$$\theta_1=\theta_2=...=\theta_{z_1};\quad \theta_{z_1+1}=\theta_{z_1+2}=...=\theta_{z_1+z_2};\quad \theta_{z_1+z_2+1}=\theta_{z_1+z_2+2}=...=\theta_z,$$

则由式(3)得:

$$\sum_{k=1}^{z}\theta_k=\sum_{k_1=1}^{z_1}\theta_{k_1}+\sum_{k_2=z_1+1}^{z_1+z_2}\theta_{k_2}+\sum_{k_3=z_1+z_2+1}^{z_1+z_2+z_3}\theta_{k_3}=z_1\theta_1+z_2\theta_{z_1+1}+z_3\theta_{z_1+z_2+1}=1.$$

假设按 5:3:2 比例关系设定核心模块、一般模块和辅助模块的关键级别等级,则有:

$$z_1\theta_1+z_2\theta_{z_1+1}+z_3\theta_{z_1+z_2+1}=5z_1\theta_0+3z_2\theta_0+2z_3\theta_0=1,$$

其中,θ_0 为临时变量.若各关键级别模块数量分别为 $z_1=5,z_2=8,z_3=3$,可求得:θ_0=1/55,进一步得到所有测试影响因子:

$$\begin{cases}\theta_1=\theta_2=...=\theta_{z_1}=5\theta_0=1/11\\ \theta_{z_1+1}=\theta_{z_1+2}=...=\theta_{z_1+z_2}=3\theta_0=3/55\\ \theta_{z_1+z_2+1}=\theta_{z_1+z_2+2}=...=\theta_z=2\theta_0=2/55\end{cases}.$$

2.3 综合测试覆盖率

2.3.1 综合测试覆盖率的定义

多维度测试覆盖率虽然从多个角度度量了软件测试充分程度,但缺少综合度量方法对软件测试充分程度给予总体评价.由此,本节提出综合测试覆盖率的概念,并给出了经验公式.

定义 2. 综合测试覆盖率 C_{syn}.

综合测试覆盖率 C_{syn} 指基于软件多维度测试覆盖率的所有特征属性和度量参数,对软件测试充分程度进行总体评价和度量.基于第 2.1 节和第 2.2 节多维度测试覆盖率模型所有度量参数,综合测试覆盖率可表示成如下函数式:

$$C_{syn}=f(n_{case},t,C,\hat{C},\omega,Type,\theta,Sat) \tag{4}$$

其中:C 表示所有维度测试覆盖率参数:$C_1,C_2,\ldots,C_m$;

$\hat{C}$ 表示所有维度测试覆盖率期望值参数: $\hat{C}_1,\hat{C}_2,...,\hat{C}_m$;

ω表示所有维度测试覆盖率的优先级因子: $\omega_1,\omega_2,...,\omega_m$;

Type 表示测试覆盖率的类型,主要包括基于代码和基于需求两类;

θ表示所有软件模块关键性测试影响因子: $\theta_1,\theta_2,...,\theta_n$;

Sat 表示所有维度测试覆盖率满意度参数: $Sat_1,Sat_2,...,Sat_m$;

其他符号的含义解释同前.

2.3.2 综合测试覆盖率的经验公式

基于代码和基于需求两类测试覆盖所处的主要测试阶段、采用的测试方法和特点均有不同.基于需求的测试覆盖,如功能覆盖、需求覆盖和接口覆盖分别从软件需求、软件功能点和软件接口三方面对整个软件测试充分性情况进行度量,因此可以不考虑模块测试影响因子θ.下面分别讨论基于代码和基于需求两类综合测试覆盖率的经验计算公式.

a) 基于代码的综合测试覆盖率 C_{syn}^{code}

i) 软件测试中,假设当前已执行完第 n_{case} 个测试用例,软件被划分为 n 个模块,有 m_1 种基于代码的测试覆盖率,结合式(4),给出被测软件基于代码的综合测试覆盖率 C_{syn}^{code} 的经验公式:

$$\begin{aligned}C_{syn}^{code}(x)&=\sum_{k=1}^{n}\theta_k\times\left(\sum_{j=1}^{m_1}C_j^k(x)\times Sat_j^k(x)\times\omega_j^k\right)\\&=\sum_{k=1}^{n}\theta_k\times\left(\sum_{j=1}^{m_1}C_j^k(x)\times\left(\frac{C_j^k(x)}{\hat{C}_j^k}\times100\%\right)\times\omega_j^k\right)\end{aligned} \tag{5}$$

其中,$\hat{C}_j^k$, $C_j^k(x)$, $Sat_j^k(x)$, θ_k 和 ω_j^k 分别表示执行完第 n_{case} 个测试用例后,模块 k 的第 j 种测试覆盖率的期望值、当前值、满意度、软件模块关键性测试影响因子和测试覆盖率优先级因子.

ii) 如果单独计算被测模块 k 的综合测试覆盖率,可用式(6):

$$C_{syn}^{k}(x)=\sum_{j=1}^{m_1}C_j^k(x)\times Sat_j^k(x)\times\omega_j^k \tag{6}$$

基于式(5)、式(6)计算综合测试覆盖率,能够体现其以下特性:

i) 某种测试覆盖率达到的满意度越高,其在综合测试覆盖率中的权重就越高;

ii) 某种测试覆盖率的优先级因子越高,其在综合测试覆盖率中的权重就越高;

iii) 某种测试覆盖率所在模块的关键性测试影响因子越高,其在综合测试覆盖率中的权重就越高.

例如,表 2 为 3 个被测模块 k=1,2,3 在执行完若干个测试用例时综合覆盖率的计算示例.j=1,2,3 分别表示语句覆盖、分支覆盖和条件覆盖.

Table 2 Illustration for synthetic test coverage (STC)

表 2 综合测试覆盖率示例

	k=1			k=2			k=3			$C_{syn}^{k}(x)$			C_{syn}^{code}
	C_1^1	C_2^1	C_3^1	C_1^2	C_2^2	C_3^2	C_1^3	C_2^3	C_3^3	k=1	k=2	k=3	
$\hat{C}_j^k$	95%	85%	70%	100%	90%	85%	85%	75%	65%				
ω_j	0.4	0.32	0.28	0.4	0.32	0.28	0.4	0.32	0.28				
θ_k		0.3			0.5			0.2		0.381	0.396	0.403	0.392
$C_j^k(x)$	78%	44%	36%	83%	46%	37%	76%	43%	35%				
Sat^k	0.82	0.518	0.514	0.83	0.511	0.435	0.894	0.573	0.538				

从表 2 的计算结果可以看出,模块 3 的各维度覆盖率相对较低,但其达到的满意度相对比较高;模块 2 的覆盖率较高,但满意度较低,计算得到模块的综合测试覆盖率 $C_{syn}^3(x)>C_{syn}^2(x)>C_{syn}^1(x)$.可见,利用模块的综合覆盖率可以比较各模块总的测试充分性程度.考虑模块测试影响因子,当前整个软件的综合测试覆盖率为 39.2%.

b) 基于需求的综合测试覆盖率

软件测试中,假设当前已执行完第 n_{case} 个测试用例,不考虑模块测试影响因子,有 m_2 种基于需求的测试覆盖率,被测软件基于需求的综合测试覆盖率 C_{syn}^{req} 由经验公式(7)给出,其中的变量含义同前.

$$C_{syn}^{req}(x)=\sum_{j=1}^{m_2}C_j(x)\times Sat_j(x)\times\omega_j \tag{7}$$

显然, $0\leqslant C_{syn}^{code}(x),C_{syn}^{req}(x)\leqslant 1$.

c) 综合测试覆盖率的期望值

如果测试中各维度覆盖率均达到期望值,则由式(5)和式(7)可分别推出对应的综合测试覆盖率期望值 $\hat{C}_{syn}$ 的计算式,即满意度均为 1 时的综合测试覆盖率:

$$\hat{C}_{syn}^{code}=\sum_{k=1}^{n}\theta_k\times\left(\sum_{j=1}^{m_1}\omega_j^k\times\hat{C}_j^k\right) \tag{5'}$$

$$\hat{C}_{syn}^{req}=\sum_{j=1}^{m_2}\hat{C}_j\times\omega_j \tag{7'}$$

综合测试覆盖率期望值 $\hat{C}_{syn}$ 可用作定义软件综合测试覆盖率充分性准则,这里不作详细讨论.

补充说明:虽然多维度覆盖率之间有一定的交叉和包含关系[8,9],但由于在测试中对各种测试覆盖率是独立进行统计的,所以在建立多维度测试覆盖率模型和计算综合测试覆盖率时,均不考虑其交叉和包含关系.

3 软件测试动态分析和评价模型

测试覆盖率是定量度量软件测试的重要手段,并且是测试时间(或测试用例数)的增长函数[12](coverage growth function,简称 CGF),因此,利用多维度覆盖率和综合测试覆盖率随测试用例数或时间的动态变化特性,可对软件测试进行动态、定量地分析和评价.文献[13]通过大量测试用例实验得出,单个测试覆盖率不能完全说明

测试用例集的测试能力,多种测试覆盖率应相互补充.同时还指出,使覆盖率达到较高的测试相比同样数量的随机测试用例集的测试,发现缺陷的效率要高.本文侧重讨论如何利用多维度测试覆盖率随 x 的各种增长曲线及增长率分布曲线,动态分析和评价软件测试效率及薄弱点,为持续优化软件测试提供决策依据.

本节使用的测试实例情况介绍如下:

在 Java 集成开发环境 Eclipse 下,载入用 Java 实现的开放源码项目 CDT 及其在 Eclipse 环境中,使用 JUNIT 开发的单元测试程序[14].使用 Eclipse 开源插件 EclEmma 软件测试工具实现对 CDT 的覆盖测试,并在测试中动态统计语句覆盖、模块覆盖和指令覆盖等代码测试覆盖率.在上述测试环境下,可根据需要分别加载不同的测试用例进行测试,并可将多次测试的结果进行合并.

CDT 是完全用 Java 实现的开放源码项目[14],作为 Eclipse SDK 平台的一组插件,使得 Eclipse 平台(Eclipse platform)提供对 C/C++开发的支持.由于其复杂性,CDT 被分为 CDT 核心(CDT Core)、CDT 插件(Primary CDT plug-in)、CDT 功能 Eclipse(CDT Feature Eclipse)等 9 个独立插件.载入 CDT Core test 测试资源,可实现对 CDT Core 插件的单元测试.CDT Core test 提供了 misc,model,parser,suite,regression 和 templateengine 等模块,目前共带有 12 361 个测试用例.本文主要选择 CDT Core test 中的 misc (36 个测试用例)、model(320 个测试用例)、parser(8 613 个测试用例)作为实例使用,在此平台下,所有模块或子模块及测试用例均可独立执行,并采集其覆盖率数据.测试实例程序“org.eclipse.cdt.core.tests”模块组成如图 3 所示,测试平台下覆盖率采集实例如图 4 所示.

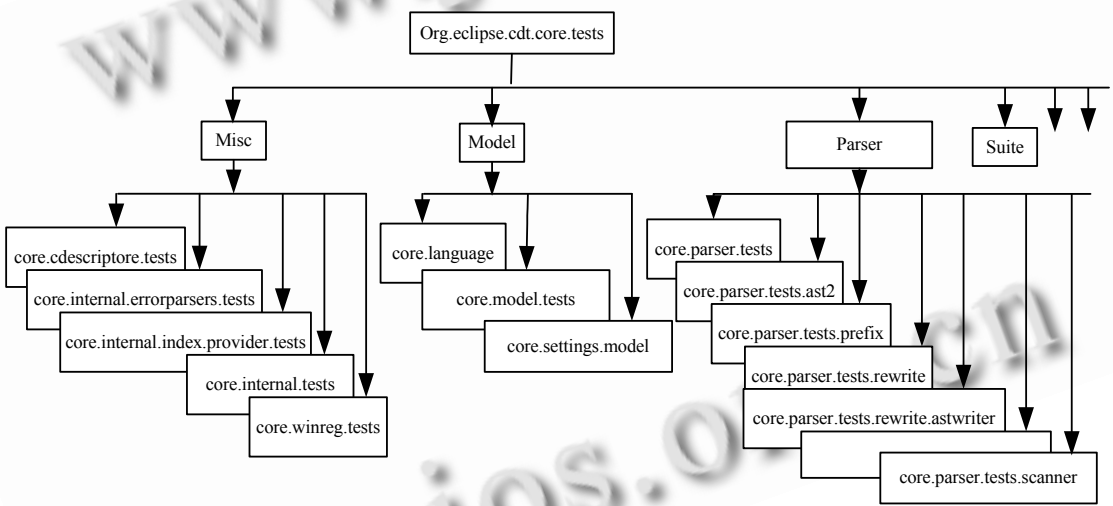

Fig.3 Components of “org.eclipse.cdt.core.tests”

图 3 “org.eclipse.cdt.core.tests”的模块组成

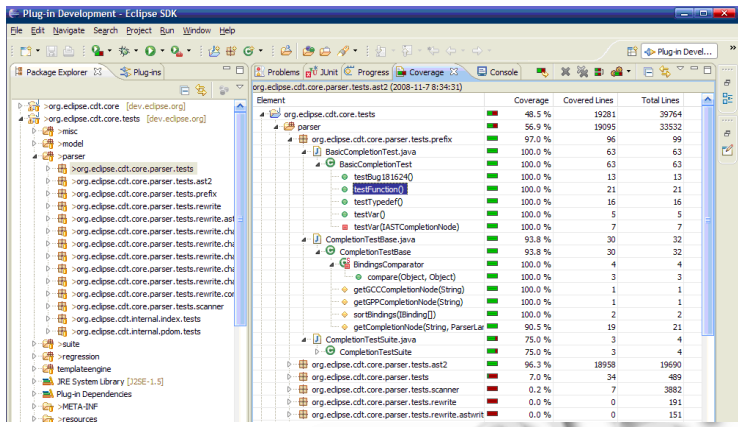

Fig.4 Example of coverage data collecting during test under Eclipse platform

图 4 测试平台下覆盖率采集实例图

3.1 多维度测试覆盖率增长曲线

1) 利用多维度测试覆盖率满意度增长曲线,分析测试能力和测试薄弱点

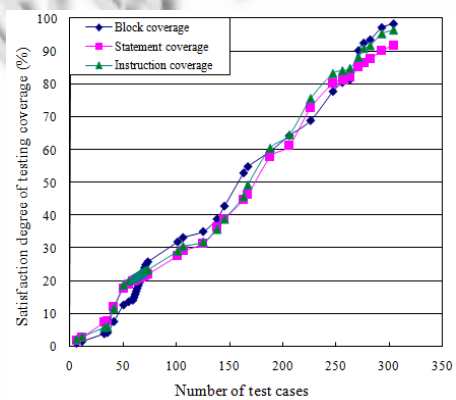

Fig.5 Growth curves of satisfaction degree of multi-dimensional test coverage during the test of "cdt.core.model"

图 5 模块"cdt.core.model"的多维度测试覆盖率满意度增长曲线

软件测试过程中,动态、同步统计并绘制各维度的测试覆盖率满意度随 x 的增长曲线.通过曲线之间的比较,测试覆盖率满意度相对明显偏低的覆盖测试是当前测试的薄弱点,所执行的那些测试用例针对该维度测试覆盖的能力较差,应改进测试策略.分析范围可以是某个被测模块内部所有维度的测试覆盖率满意度之间的比较,或各模块之间某一维度的测试覆盖率满意度的比较,也可以是整个被测软件的各维度测试覆盖率满意度间的比较.图 5 是被测模块"cdt.core.model"的 320 个测试用例测试情况.由图 5 可以直观看出,测试中块覆盖率和指令覆盖率的满意度比较高,最终分别达到了 99.5%和 97.3%,语句覆盖率满意度相对较低,最终只达到 93.3%.因此,需要进一步增加语句覆盖测试.

2) 利用多维度测试覆盖率增长量,评价测试用例的测试能力和效果

用 $\Delta C_j(i)=C_j(i)-C_j(i-1)$ 表示执行第 i 个测试用例某种测试覆盖率 C_j 的增长量;用 $\Delta N^i_{D_{find}}=N^i_{D_{find}}-N^{i-1}_{D_{find}}$ 表示执行该测试用例期间发现的软件缺陷数的增长量;$C_j(i)$、$N^i_{D_{find}}$ 分别表示执行完 i 个测试用例后,覆盖率 C_j 的值和发现的软件缺陷总数.

如果执行了第 i 个测试用例,对于任意 $j, C_j(i), N^i_{D_{find}}$ 均变化很小,即 $\forall j\in\{1,2,...,m\}$, $0\leqslant\Delta C_j(i)\leqslant a$ & $0\leqslant\Delta N^i_{D_{find}}\leqslant b$,a,b 均为很小的值,如 a=0.05,b=0,则说明该测试用例的测试能力和测试效果较差,属于冗余测试/重复测试或无效测试.实际测试过程中,如果分析单个测试用例工作量太大,最好选择连续执行的 $N(N\geqslant 1)$个测试用例为单位,统计其各种覆盖率和发现软件缺陷数的变化情况.

3) 被测模块综合测试覆盖率增长曲线

软件测试过程中,动态统计并绘制各被测软件模块的综合测试覆盖率 $C_{syn}^{k}(x)$ 随 x 的增长曲线.通过曲线之间的比较,结合模块测试影响因子 θ_k,可分析当前所有被测模块,尤其关键核心模块的综合测试覆盖率是否明显偏低.如果 θ_k 相对较高,但 $C_{syn}^{k}(x)$ 相对偏低,则针对该模块的测试是当前的测试薄弱点,应调整当前测试策略,优先保证对核心模块的测试.

4) 软件综合测试覆盖率增长曲线

软件测试过程中,动态绘制软件综合测试覆盖率随 x 的增长曲线.可以了解当前软件整体的测试充分程度,并结合测试进度和软件综合测试覆盖率期望值评价软件整体测试进展情况.

5) 软件被测模块中发现的缺陷密度及其综合测试覆盖率增长曲线

Pareto 原则(二八定理)[3]指出,测试发现错误中的 80%很可能起源于 20%的模块中.如果测试中,通过曲线发现某被测软件模块的缺陷密度相对较高,但综合测试覆盖率相对偏低,则该模块为当前测试的薄弱点.应该调整测试策略,加强针对薄弱点的测试.

3.2 多维度测试覆盖率的增长速率曲线,度量测试效率和薄弱点

如何定量度量软件测试效率是提高测试效率的前提,也是软件测试领域关注的重点.目前常用的方法[15]是利用缺陷的发现趋势(多少)来度量测试效率,但软件中存在多少缺陷和执行测试能够发现多少缺陷存在一定的不确定性,仅考虑缺陷不能从普遍意义上很好地度量测试的有效性.本文提出利用多维度测试覆盖率(包括缺陷覆盖率)增长函数,计算连续执行的若干测试用例期间的覆盖率增长速率,并结合发现的缺陷数来度量测试效率和测试薄弱点.具体方法如下:

用 Δt_i 表示执行第 i 个测试用例所用的时间,$\Delta C_j(i)=C_j(i)-C_j(i-1)$ 表示执行第 i 个测试用例某种测试覆盖率 C_j 的变化量.如果从执行第 l 个测试用例开始,连续执行 k 个测试用例,则 k 个测试用例的测试执行时间 Δt 为

$$\Delta t=\sum_{i=l}^{i=l+k}\Delta t_i \tag{8}$$

C_j 在 Δt 时间内随测试执行时间的平均增长率 $V_{C_j}^{k-time}$ 为

$$V_{C_j}^{k-time}=\frac{(\Delta C_j)_k}{\Delta t}=\frac{\sum_{i=l}^{i=l+k}\Delta C_j(i)}{\sum_{i=l}^{i=l+k}\Delta t_i}=\frac{C_j(l+k)-C_j(l)}{\sum_{i=l}^{i=l+k}\Delta t_i} \tag{9}$$

相应的,可计算 C_j 随测试用例数的平均增长率:

$$V_{C_j}^{k-case}=\frac{(\Delta C_j)_k}{k}=\frac{C_j(l+k)-C_j(l)}{k} \tag{9'}$$

进一步可计算出 m 种测试覆盖率的平均增长率为

$$V_C^{Ave}=\sum_{j=1}^{m}V_{C_j}\Big/m \quad ,V_{C_j}\text{ 代表 }V_{C_j}^{k-time}\text{ 或 }V_{C_j}^{k-case} \tag{10}$$

用 $\Delta N_{D_{find}}^{k}$ 表示连续执行 k 个测试用例期间发现和排除的软件缺陷数的增长量,即

$$\Delta N_{D_{find}}^{k}=N_{D_{find}}^{l+k}-N_{D_{find}}^{l} \tag{11}$$

测试中,每间隔 k(k 可取≥1 的任意整数)个测试用例,统计和计算一次 $V_{C_j}, j\in\{1,2,...,m\}$,动态绘制 $V_{C_j}^{k-case}$、$\Delta N_{D_{find}}^{k}$ 随测试用例数和 $V_{C_j}^{k-time}$、$\Delta N_{D_{find}}^{k}$ 随测试执行时间的变化曲线.通过曲线可以很直观地看出各维度测试覆盖率的增长率及发现的缺陷数的变化情况.在测试未达到测试目标前,如果测试中发现最近连续执行 k 个测试用例后,所有测试覆盖率的 $V_{C_j}^{k-time}$ 和 $V_{C_j}^{k-case}$ 均小于某个值 r,即 $\forall j\in\{1,2,...,m\}$, $V_{C_j}^{k-time}\leqslant r$ & $V_{C_j}^{k-case}\leqslant r$,并且

$\Delta N^k_{D_{find}} \leqslant e$,如 $k \geqslant 5$,r=0.2%,e=1,可认为最近执行的 k 个测试用例及测试方法的测试效率偏低,可能存在比较严重的冗余性测试/重复性测试或无效测试,需要改进测试方法或测试用例集,提高测试效率,尤其是优先级高,但 V_{C_j} 偏低的覆盖测试.

实例如图 6 所示,执行模块"cdt.core.model"的 320 个测试用例,动态采集其块覆盖率、语句覆盖率和指令覆盖率的变化情况,并分别计算其覆盖率增长率.测试中未发现软件缺陷.若以 $k \geqslant 5$,r=0.2%为判断标准,则由图 6 发现有 4 处(用例数 7~11、125~138、263~271、293~320)所执行的测试用例的测试效率比较低.

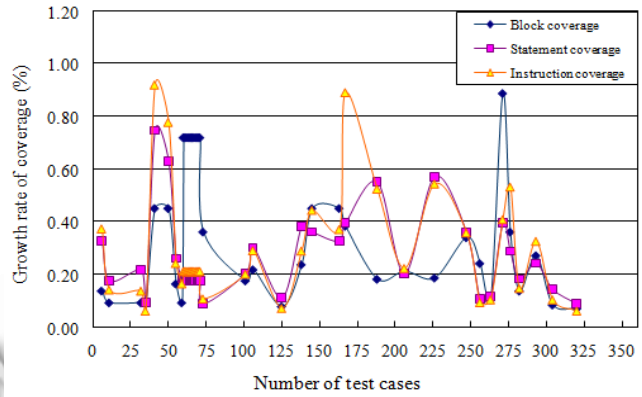

Fig.6 Curves of test coverage growth rate with the number of test cases for model "cdt.core.model"

图 6 模块"cdt.core.model"的测试覆盖率增长率随测试用例数的变化曲线

4 测试优化策略

对软件测试进行动态评价的目的在于持续优化测试,提高测试效率.基于上节软件测试分析和评价模型,本节讨论如何根据软件测试评价结果,制定测试优化策略,实现软件测试的动态优化.

首先,给出制定软件测试优化策略的原则和目标:

1) 在时间和成本等测试资源约束条件下,优化原有测试策略,优先保证 θ_k 和 ω_j 较高的覆盖测试,并针对性地改进测试方法和测试用例集,减少冗余测试和无效测试,提高测试效率.

2) 在相同测试资源约束条件下,通过多层次的测试策略优化,持续提高测试效率,使测试更加充分.

然后,本文分别从基于多维度测试覆盖率满意度、模块综合测试覆盖率和测试效率几方面讨论如何优化测试策略.

4.1 基于多维度测试覆盖率满意度的测试优化策略

1) 如图 7 所示,在被测软件模块内,动态分析各维度测试覆盖率(主要是基于代码的测试覆盖率)的满意度,优先针对满意度最低的覆盖测试改进测试策略(测试优化策略 1).

2) 如图 8 所示,动态分析整个软件的各维度测试覆盖率满意度,优先针对满意度最低的覆盖测试改进测试策略(测试优化策略 2).

4.2 基于各模块综合测试覆盖率的测试优化策略

如图 9 所示,比较各模块综合测试覆盖率,结合 θ_k,优先优化 θ_k 相对较高,但综合测试覆盖率最低的被测模块的测试策略(测试优化策略 3).

4.3 基于测试效率的测试优化策略

如图 10 所示,依据第 3 节中软件测试能力和测试效率的评价方法,当发现连续执行若干测试用例后,测试用例的测试效率和测试能力较差,则需要分析采用的测试方法以及测试用例的针对性,优化测试用例集和测试方

法,实现对测试策略的优化(测试优化策略 4).

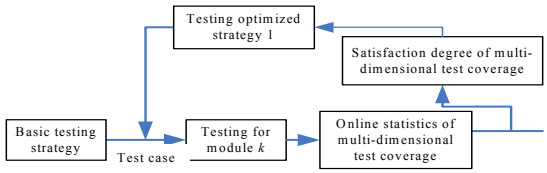

Fig.7 Testing optimized strategy for the module test based on the satisfaction degree of its multi-dimensional test coverage

图 7 被测软件模块内基于各维度测试覆盖率满意度的测试优化策略

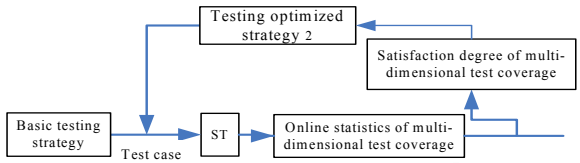

Fig.8 Testing optimized strategy for the software test based on the satisfaction degree of its multi-dimensional test coverage

图 8 基于整个软件各维度测试覆盖率满意度的测试优化策略

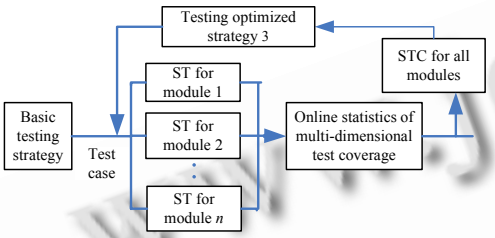

Fig.9 Testing optimized strategy based on the STC

图 9 威者基于各模块综合测试覆盖率的测试优化策略

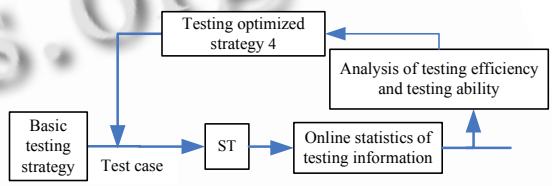

Fig.10 Testing optimized strategy based on test efficiency

图 10 基于软件测试效率的测试优化策略

4.4 测试优化策略的分类与实现方法

测试优化策略包括以下几类:

1) 优化测试用例集.依据软件测试动态评价结果和图 7~图 10 中的测试优化策略需求,针对性地分析测试用例集,通过调整测试用例顺序,改进测试用例,增加或删减测试用例等方法,改进测试用例集的针对性、有效性和分布的合理性.

2) 改进测试方法.为了提高测试效率和测试的针对性,依据软件测试动态评价结果和测试优化策略需求,改进测试方法.如采用面向软件缺陷的测试方法来提高针对软件缺陷类型的覆盖测试,;采用面向对象的软件测试方法来提高面向对象的测试覆盖率.

3) 针对性地使用各种自动化测试工具[1−4](如 Parasoft 公司的 C++Test,Telelogic 公司的 LogiScope 及 Compuware 公司的 TureCoverage 等),提高测试的自动化能力和信息动态采集能力,进而提高测试效率.

由于针对测试策略的研究不是本文研究重点,这里只简单讨论优化策略的分类和方法,后续我们将专门针对测试优化策略,研究测试用例集分析、测试用例顺序调整、测试用例针对性改进以及测试用例增删减的方法.2007 年专门成立的第 1 个国际软件测试评价组织[5],软件测试数据(test Oracle)的分析和改进方法是其研究热点之一.

4.5 测试优化实例

本节选择图 3“core.parser.tests.scanner”模块中针对其子模块“Portedscanner”的 134 个测试用例作为实例,测试中动态统计其语句覆盖率.分 3 种情况执行测试用例:(a) 按随机顺序执行测试用例集中的用例;(b) 按随机执行顺序执行了 50 个测试用例后,分析测试用例集中剩余测试用例的特点,优先执行可以尽快提高语句覆盖率的测试用例;(c) 在前两个测试结果的基础上,总结分析测试用例特点,按最优顺序,即按语句覆盖率大小排序,按最理想执行顺序执行测试用例.执行结果如图 11 所示.可以看出,同样一个测试用例集,采用不同的执行顺序,其

测试效率不同.如在执行到第 80 个测试用例,最优测试(c)可使语句覆盖率达到 84.31%,(b)情况为 78.89%,(a)情况只有 52.43%.当测试用例集中的用例数量较多时,按随机顺序执行测试用例,很难保证和控制测试效率.若采取一定的测试优化策略,在相同的时间约束条件下,能够在一定程度上改进测试效率.因此,动态监测和分析测试覆盖率情况,针对性地优化测试,可使测试效率得到持续提高.

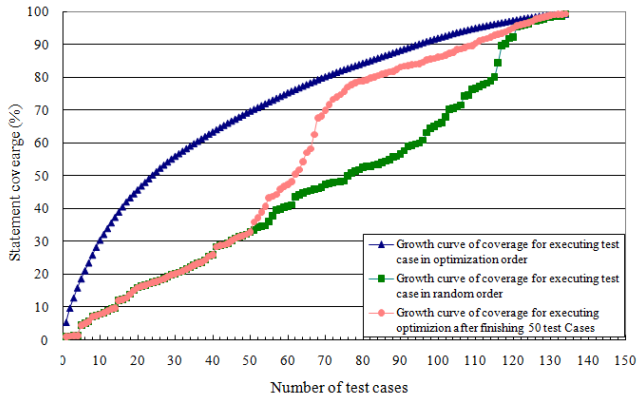

Fig.11 Examples of software test optimization based on growth curves analysis of test coverage

图 11 依据测试覆盖率增长曲线分析来优化软件测试的实例

5 结束语

软件测试中,软件管理者和测评人员都希望能够动态跟踪和定量评价软件测试,及时发现测试薄弱点,并有针对性地优化测试,以保证测试的有效性和充分性.本文提出了一种基于多维度测试覆盖率的软件测试动态评价方法,并从测试监测信息、动态分析和评价模型、测试优化策略几方面开展讨论.在 Eclipse 平台下,使用开源软件 CDT、开源插件 EclEmma 软件测试工具及其测试用例库,通过实例说明了上述方法的有效性.该方法将被试用于新一代飞机综合化航空电子系统大型应用软件工程中,今后将对相关评价模型和经验公式作进一步改进和完善.

本文讨论的多维度测试覆盖率及综合测试覆盖率的概念和计算模型适于推广应用,例如,文献[12,16]等均指出软件可靠性模型中应该考虑软件测试覆盖率因素.但由于目前有 10 余种覆盖率指标,文献中均未提及具体如何选择或使用现有多种覆盖率.后续工作将利用本文中多维度覆盖率模型及综合测试覆盖率指标研究基于多维度测试覆盖率的软件可靠性建模和软件质量评价方法.另外,基于综合测试覆盖率可以进一步研究综合测试覆盖率充分性准则等.

致谢 在此,衷心感谢清华大学计算机科学与技术系林闯教授和白晓颖副教授对本文工作所提供的指导性意见和帮助.

References:

[1] Perry WE, Wrote; Gao M, Feng F, Xu L, Trans. Effective Methods for Software Testing. 3rd ed., Beijing: Tsinghua University Press, 2008 (in Chinese).

[2] Lewis WE, Veerapillai G, Wrote; Chen SY, Zhang HT, Liu JH, Jin CJ, Trans. Software Testing and Continuous Quality Improvement. 2nd ed., Beijing: Posts & Telecom Press, 2008 (in Chinese).

[3] Zhao B. Software Testing Technology Classic Course. Beijing: Science Press, 2007 (in Chinese).

[4] Gu Y, Shi JL. Generality for Technology of Software Testing. Beijing: Tsinghua University Press, 2004 (in Chinese).

[5] Mayer J, Beydeda S. Message of the program chairs of STEV'07. In: Proc. of the 7th Int'l Conf. on Quality Software (QSIC 2007). Washington: IEEE Computer Society, 2007. 370−371. http://doi.ieeecomputersociety.org/10.1109/QSIC.2007.35

[6] Chernak Y. Validating and improving test-case effectiveness. IEEE Software, 2001,18(1):81−86.

[7] He YT, Hecht H, Paul RA. Measuring and assessing software test processes using test data. In: Proc. of the 5th IEEE Int'l Symp. on High Assurance Systems Engineering (HASE 2000). 2000. 259−264. http://doi.ieeecomputersociety.org/10.1109/HASE.2000.895470

[8] Zhu H, Hall PAV, May JHR. Software unit test coverage and adequacy. ACM Computing Surveys, 1997,29(4):366−427. [doi: 10.1145/267580.267590]

[9] Li QY, Lu MY, Ruan L. Theoretical research on software reliability testing adequacy. Journal of Beijing University of Aeronautics and Astronautics, 2003,29(4):312−316 (in Chinese with English abstract).

[10] Burnstein I, Suwanassart T, Carlson R. Developing a testing maturity model for software test process evaluation and improvement. International Test Conference, 1996. 581−589.

[11] Miller SD, DeCarlo RA, Mathur AP. Modeling and control of the incremental software test process. In: Proc. of the 28th Annual Int'l Computer Software and Applications Conf. (COMPSAC 2004). 2004. 156−159. http://doi.ieeecomputersociety.org/10.1109/CMPSAC.2004.1342700

[12] Fujiwara T, Park JY, Park JH. Evaluation and application of MVFs in coverage for coverage-based NHPP SRGM frameworks. In: Proc. of the 5th Int'l Conf. on Software Engineering Research, Management and Applications. 2007. 385−392. http://doi.ieeecomputersociety.org/10.1109/SERA.2007.85

[13] Hutchins M, Foster H, Goradia T, Ostrand T. Experiments on the effectiveness of data flow-and control flow-based test adequacy criteria. In: Proc. of the 16th Int'l Conf. on Software Engineering (ICSE). Los Alamitos: IEEE Computer Society Press, 1994. 191−200.

[14] http://www.eclipse.org/cdt, 2008.

[15] Briand LC, Pfahl D. Using simulation for assessing the real impact of test-coverage on defect-coverage. IEEE Trans. on Reliability, 2000,49(1):60−70. [doi: 10.1109/24.855537]

[16] Cai X, Lyu MR. Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project. In: Proc. of the 18th IEEE Int'l Symp. on Software Reliability Engineering. 2007. 17−26. http://doi.ieeecomputersociety.org/10.1109/ISSRE.2007.17

附中文参考文献:

[1] Perry WE,著;高猛,冯飞,徐璐,译.软件测试的有效方法.第 3 版,北京:清华大学出版社,2008.

[2] Lewis WE,Veerapillai G,著;陈绍英,张河涛,刘建华,金成姬,译.软件测试与持续质量改进.第 2 版,北京:人民邮电出版社,2008.

[3] 赵斌.软件测试技术经典教程.北京:科学出版社,2007.

[4] 古乐,史九林.软件测试技术概论.北京:清华大学出版社,2004.

[9] 李秋英,陆民燕,阮镰.软件可靠性测试充分性问题的理论研究.北京航空航天大学学报,2003,29(4):312−316.

安金霞(1973－),女,河北邢台人,博士生,高级工程师,主要研究领域为软件测试,软件可靠性.

李树芳(1978－),男,工程师,主要研究领域为软件测试.

王国庆(1956－),男,博士,研究员,主要研究领域为航空电子系统,软件工程.

朱纪洪(1968－),男,博士,教授,博士生导师,主要研究领域为飞行控制,容错计算.