ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1379−1385
DOI: 10.3724/SP.J.1001.2008.01379

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

统计测试的软件可靠性保障能力研究*

赵 亮[1,2+], 王建民[1], 孙家广[1]

[1](清华大学 软件学院,北京 100084)

[2](太原卫星发射中心,山西 太原 030027)

Study on the Assurance Ability of Statistical Test on Software Reliability

ZHAO Liang[1,2+], WANG Jian-Min[1], SUN Jia-Guang[1]

[1](School of Software, Tsinghua University, Beijing 100084, China)

[2](TaiYuan Satellite Launching Center, Taiyuan 030027, China)

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

Zhao L, Wang JM, Sun JG. Study on the assurance ability of statistical test on software reliability. *Journal of Software*, 2008,19(6):1379−1385. http://www.jos.org.cn/1000-9825/19/1379.htm

Abstract: This paper studies statistical test's assurance ability on different programs and brings forward to incorporating effectiveness information into software reliability assessment. This can improve the accuracy of reliability assessment. This paper empirically investigates the reasonableness of this method. The conclusion provides a new way for high reliable software assurance.

Key words: statistical testing; software reliability; effectiveness

摘 要: 研究了统计测试对不同软件的测试能力,提出将测试有效性信息综合到软件可靠性评估模型中,以提高可靠性估计的针对性和精度.通过实验证明了该方法的合理性.该方法为高可靠性的软件质量保障要求提供了途径和可能.

关键词: 统计测试;软件可靠性;有效性

中图法分类号: TP311 **文献标识码**: A

统计测试是软件可靠性评估的经典方法,其基本原理是按照软件的实际操作剖面来组织测试序列,根据测试过程中软件所表现出的失效特征来预测软件在实际操作环境下的可靠性特征[1,2].但是,这种可靠性评估方法存在一些理论上的缺陷,限制了其在工程实际中的广泛应用:首先,它不考虑统计测试对被测软件的缺陷检测能力.测试过程中,有些测试用例既不增加缺陷覆盖率,也不增加代码覆盖率,但能延长软件无故障运行的次数(或时间).用这样的样本数据进行可靠性评估,估计结果过于乐观[3];其次,该方法不考虑被测软件的具体特征,不同复杂度的软件达到相同的可靠性保障水平所需要的测试数量是一样的,仅依赖于给定的可靠性目标,结论具有一定的盲目性[4];第三,统计测试的估计结果与操作剖面紧密相关,这种相关性使得不同软件的可靠性估计结果作为软件的一个质量指标缺乏可比性;另外,对于高可靠应用,这种方法需要测试的数量远远超过了现实可行的

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2003AA4Z3040 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2002CB312000 (国家重点基础研究发展计划(973))

Received 2006-12-03; Accepted 2007-04-25

范围,因此,不适用于高可靠软件的质量保障过程[5].进入 21 世纪,随着软件在社会生活中的应用日趋广泛和角色的日益关键,越来越多的应用领域要求信息系统是超高可靠的[6],作为信息系统的关键组成部分,保证软件的超高可靠成为必然和迫切的要求.这些需求推动了软件工程研究人员对高可靠、更精确的软件质量保障方法和模型的研究.

为使统计测试的结果更准确地反映软件的可靠性水平,越来越多的研究者提出在可靠性模型中增加软件产品本身的可度量特征[4]或测试的覆盖水平[3,7−9]等反映软件产品或测试过程特征的因素来进行可靠性评估.我们认为,统计测试对不同的软件具有不同的测试能力,因此,相同数量的统计测试对不同的软件应达到不同的可靠性保障水平.本文通过实验的方式验证了这个假设.本文的结论一方面克服了传统统计模型的盲目性,增加了可靠性评估的针对性,提高了可靠性的预测精度;另一方面,该结论支持通过加强测试的有效性来减少达到一定可靠性目标所需要的统计测试的数量,从而使高可靠性要求的软件质量保障成为可能.

1 统计测试的测试能力

统计测试(ST)是一种特殊的随机测试,其测试用例按照操作剖面的要求从软件输入域中随机选择.对测试而言,测试的有效性在于能够发现软件中的缺陷.一种测试方法能够检测到软件中的缺陷越多,越能够有效地提高软件的可靠性.根据软件测试的接力模型[10],一次有效的测试对应软件测试过程中的一次失效.因此,通过对软件在测试过程中的失效率进行量化分析,可以表示出一种测试方法对于被测软件的有效性.

就一个具体的软件缺陷而言,它存在于特定的软件上下文中.软件特征、缺陷特征以及测试方法的特征共同决定了测试发现该缺陷的概率.相同类型的缺陷在不同的软件中具有不同的检测难度.从测试方法来看,每种测试方法都不会拒绝一个具体的测试用例.从满足某种测试方法要求的测试用例中选择一个能够检测到缺陷的用例具有一定的概率特征,因此,测试方法对单个缺陷的检测能力可以用一个概率值来表示.即测试方法 T 检测到软件 P 中的缺陷 f 的概率为 $x(0\le x\le 1)$.

软件可靠性是针对整个软件而言的,而不是单个缺陷.通常,软件不只包含一个缺陷.不同缺陷在测试过程中导致的失效率是不同的,即:

设软件 P 包含缺陷$\{f_1,f_2,\ldots,f_n\}$,则:

T 检测到软件 P 中各缺陷的概率为$\{x_1,x_2,\ldots,x_n\}$;一般地,有 $x_i\ne x_j(1\le i,j\le n$ 且 $i\ne j)$.

此时,用任何一个概率来表示测试方法对软件 P 的测试能力都是不准确的.综合考虑多个缺陷引起的软件执行失效,其特征可以用一个分布来表示.本文选用β分布来表示软件包含多个缺陷时,统计测试下的失效率特征,即 $X\sim\beta(a,b)$,主要原因包括:

(1) β分布能够描述值域限定在一定范围内的变量的分布情况,当参数 a 和 b 变化时可以表现出多种不同的分布形状,有较好的普适性[11].统计测试过程中软件的失效率在[0,1]范围内变化,因此可以用β分布来表示.

(2) β 分布能够综合多种数据源对变量的变化情况进行分析,这种特征为综合软件特征数据和测试有效性数据提供了基础,即使在实验样本数据较少的情况下也能得到较好的估计效果[12].

(3) 在Bayesian框架下,β分布的后验函数也是β分布,能够表示前一阶段的测试对后面测试的影响.这样,通过收集历次测试的信息可以对软件的可靠性进行更准确的估计[13].在软件开发的各种过程模型中(螺旋模型、极限(敏捷)编程和 RUP 方法),测试是一个不断持续的过程,前面完成的测试为后面进行的测试提供先验知识,并指导后续的测试.

根据以上分析,有如下定理:

定理[13]. 设统计测试过程中,软件失效率分布满足分布$\beta(a,b)$,则为保障可靠性水平(θ_0,α)(θ_0 为目标失效率,α为风险水平)所需的最小测试数量 N 应满足:

$$\int_0^{\theta_0}\frac{\theta^{a-1}(1-\theta)^{b+N-1}}{B(a,b+N)}\mathrm{d}\theta\ge\alpha \tag{1}$$

根据经典统计模型,为保障可靠性水平(θ_0,α)所需的最少统计测试数量 N 应满足:

$$1-\alpha=1-(1-\theta_0)^N \tag{2}$$

对式(1),若忽略统计测试对被测软件的测试有效性,即模型参数取 a=1,b=1,则式(1)的计算结果与式(2)的计算结果一致.下文将式(1)称为基于测试能力的 Beta 模型(Beta 模型),式(2)称为经典统计模型.

根据以上理论,本文要验证的假设可以表示为:

$H1$:程序 P 在 ST 过程中因不同缺陷造成的失效率满足 β 分布;

$H2$:ST 对不同软件 P 的测试能力不同,表现为进行测试时,其失效率的 β 分布不同;

$H3$:在用 ST 评估不同的软件时,达到相同的可靠性目标需要不同的测试数量.

2 实验验证

2.1 实验对象

为了验证我们的假设,我们选用 Siemens 程序[14]进行实验验证.该程序包包含 7 个不同类型的程序.实验程序的基本特征见表 1,表中的"*"表示实验中实际的缺陷数与程序发布文档中声明的不一致.

Table 1 The feature of subject programs

表 1 实验程序基本特征

Program	Loc	Executable		# Vers	Test pool size	Descriptions
		Edges	Dus			
Totinfo	346	83	292	23	105 2	Information measure
Schedule 1	299	62	294	9	265 0	Priority scheduler
Schedule 2	297	80	217	9*	271 0	Priority scheduler
Tcas	138	46	57	40*	160 8	Altitude separation
Printtoken 1	402	97	268	7	413 0	Lexical analyzer
Printtoken 2	483	159	240	10	411 5	Lexical analyzer
Replace	516	191	664	32	554 2	Pattern replacement

2.2 假设1的检验

$H1$ 是 3 种假设中的根本,其检验是一个典型的分布簇检验问题,可将对每个程序的检验转换为所有程序的失效率分布的检验,即所有程序在测试中的运行失效率样本数据是否满足 β 分布[15].这样既可以增加样本数据,提高了检验结果的准确性,也避免了逐个检验的麻烦.根据分布簇的检验方法,检验步骤包括:

(1) 对每个程序,从测试池中随机选择一定数量的测试用例组成测试用例序列;

(2) 分别打开各程序的各个缺陷,使用测试序列进行测试,统计失效率,得到样本数据 $x_1,\ldots,x_r$.

(3) 设假设成立,利用最大似然法,计算总体 β 分布的分布参数 a 和 b;

(4) 根据$\chi 2$ 检验的要求,将失效率样本数据分成不同的区间 $L_j,j=1,\ldots,k$;分别统计落在每个区间内的样本个数 $n_j,j=1,\ldots,k$.

(5) 根据分布计算总体变量落入各区间的估计概率 $\tilde{p}_j = P(X \in L_j)$,$j=1,\ldots,k$;

(6) 构造χ^2 统计量:

$$K_n^2 = \frac{1}{n}\sum_{j=1}^{k}\frac{n_j^2}{\tilde{p}_j} - n,$$

比较χ^2 统计量与指定置信度水平 1−α时的标准分布值.

若 $K_n^2 > \chi_\alpha^2(k-u-1)$,则拒绝原假设;否则接受原假设.其中,$u$ 为待检验分布包括的参数个数,β 分布的检验中 u=2,风险水平α指定为 0.05.

为检验 $H1$,我们从测试池中随机选择不同的测试用例组成测试序列,各程序使用的测试用例序列长度见后文表 3 中"测试序列长度"列.各程序的失效率数据如图 1 中对应的曲线所示.图 1 中,横坐标 Fx 表示对应的缺陷号,纵坐标表示测试过程中因该缺陷导致的失效率.

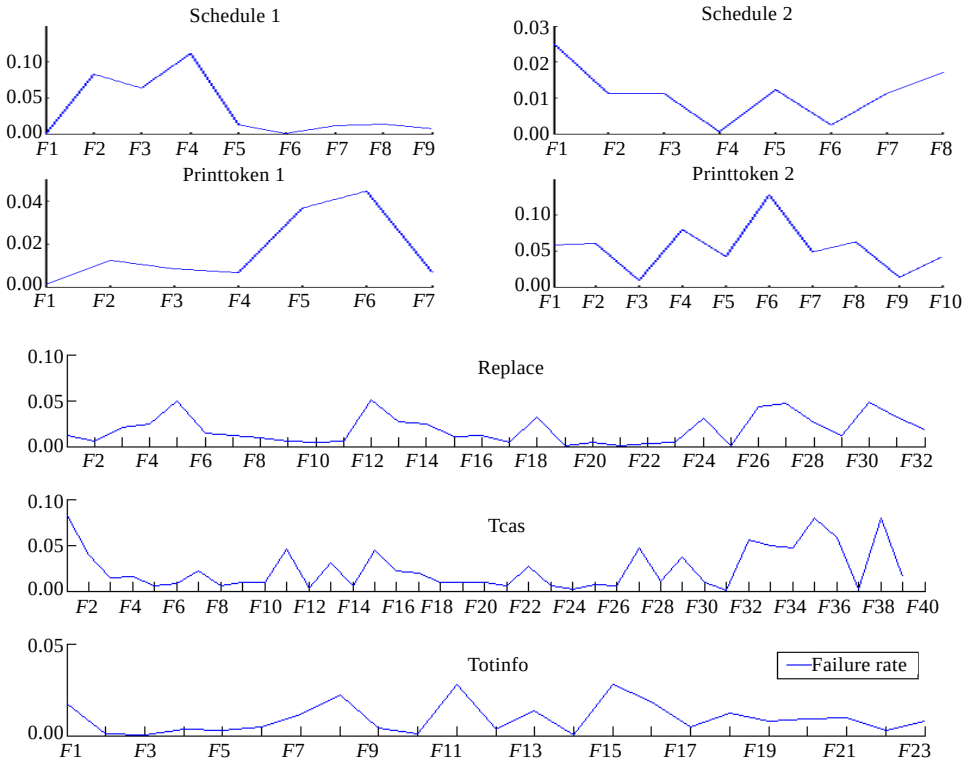

Fig.1 Program failure rate owing to different faults under statistical test

图 1 各程序因不同缺陷造成的统计测试下的失效率

数据分析:实验中,样本总数 n=129,区间数 k=12.

总体分布参数估计结果:a=0.736332,b=19.01934.

失效率区间划分情况和各区间样本数、样本概率和估计概率见表 2.

Table 2 Distribution test of statistical test sequence

表 2 统计测试序列分布检验

Failure rate range	#Samples	Sample probability	Estimated probability
[0,0.000549]	3	0	0.037 6
(0.000549,0.00111)	3	0.046 512	0.025 27
(0.00111,0.00222)	7	0.054 264	0.040 99
(0.00222,0.004502)	8	0.062 016	0.067 956
(0.004502,0.006713)	10	0.077 519	0.549 88
(0.006713,0.009989)	12	0.093 023	0.069 786
(0.009989,0.01865)	24	0.186 047	0.144 35
(0.01865,0.02763)	10	0.077 519	0.111 751
(0.02763,0.048875)	25	0.193 798	0.176 747
(0.048875,0.097699)	17	0.131 783	0.182 46
(0.097699,0.277469)	10	0.077 519	0.087 059
(0.277469,1)	0	0	0.001 034

根据上表计算得出: K_n^2=8.739874.

该χ^2检验的自由度为 k–2–1=12–2–1=9.

在显著性水平为 0.05 时, $\chi^2_{0.05}(9)$ =16.919>8.739874.

因此,我们有 95%的置信度接受原假设,即随机序列下各程序测试失效率的分布满足 β 分布,$H1$ 成立.

2.3 假设2的检验

根据测试获得的数据,对统计测试下各程序的失效率分布参数进行估计.表 3 列出了分别用 3 种不同的方法计算得到的分布参数.Max-Likelihood 表示最大似然法,s2-Based Method 表示基于方差的估计方法,s2-Based Method 表示基于样本差的估计方法.从结果可以看出,对同一个程序,用不同的方法计算得到的结果基本上是一致的或接近的.这一方面验证了计算方法的正确性,另一方面也说明,不同的统计方法具有不同的统计解释能力,实际应用中应综合考虑.

Table 3 Statistical test effectiveness for different programs

表 3 统计测试对不同程序的测试效果

Program	Max-Likelihood		σ^2-Based method		s^2-Based method		Length of ST sequence
	a	*b*	*a*	*b*	*a*	*b*	
Schedule 1	0.66	18.32	0.75	20.61	0.7	19.34	253 8
Schedule 2	1.37	116.75	2.52	214.51	2.36	201.04	257 7
Printtok 1	1.12	65.37	1.11	64.96	1.03	60.25	402 2
Printtok 2	2.2	37.77	2.69	46.14	2.56	43.78	387 1
Peplace	1.04	55.06	1.36	71.76	1.34	70.62	546 5
Totinfo	0.89	8.69	1.06	10.24	1.03	10	901
Tcas	1.04	41.73	1.04	42.16	1.03	41.61	155 5

对假设 $H2$,从表 3 可以看出,用任何一种方法计算得到的分布参数的结果,各程序都不一样,即 $a_i \neq a_j, b_i \neq b_j$ $(1 \leq i,j \leq n$ 且 $i \neq j)$.因此,我们可以确认统计测试对不同的软件具有不同的测试能力,即 $H2$ 成立.

根据估计结果可以画出各程序在统计测试下的失效率密度函数曲线.为了表示清晰,我们把 7 个程序的失效率密度函数曲线分别画在图 2(a)和图 2(b)中.每幅图都包含一条经典统计模型假设的失效率密度函数曲线(图中 Standard 曲线所示)作对比.

从图中可以看出,统计测试过程中每个程序实际的失效率曲线和经典统计模型假设的曲线是完全不同的,即经典统计测试的假设在实际测试中往往是不成立的.

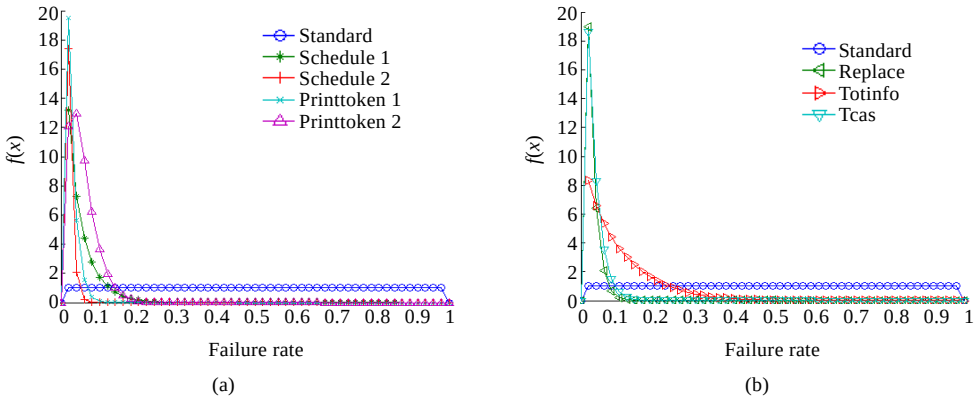

Fig.2 Program failure rate density curve under statistical test

图 2 统计测试下各软件的失效率密度函数曲线

2.4 假设3的检验

设定可靠性目标为(0.001,0.01),分别计算经典统计模型(N_{ST})、基于测试能力的模型(下文简称为 Beta 模型,N_{Beta})和 PAC 模型(N_{PAC})所需要的测试的数量.结果见表 4.其中,“*”表示实际的伪维数与程序文档中发布的不一致.

Table 4 The necessary test numbers for given reliability goal from different models

表 4 设定可靠性目标各可靠性估计模型所需测试数量

Program	N_{ST}	Beta model			PAC model	
		a	b	N_{Beta}	$dimp(p)$	N_{PAC}
Schedule 1	460 2	0.66	18.32	375 1	9	6.2187e+006
Schedule 2	460 2	1.37	116.75	528 8	9*	6.2187e+006
Printtoken 1	460 2	1.12	65.37	480 7	7	4.8516e+006
Printtoken 2	460 2	2.2	37.77	696 5	10	6.9022e+006
Replace	460 2	1.04	55.06	463 9	32	2.1940e+007
Totinfo	460 2	0.89	8.69	433 8	23	1.5788e+007
Tcas	460 2	1.04	41.73	465 2	40*	2.6725e+007

PAC 模型的计算公式为(具体参见文献[4])

$$N \geq \frac{32}{\theta}\left[2\dim_p(P)\ln\left[\frac{16e}{\theta}\right]+\ln\left[\frac{8}{1-\alpha}\right]\right].$$

从表 4 可以看出,为保障可靠性目标(0.001,0.01):

- 经典统计模型需要的测试数量对每个程序都一样,N=4602.因此,我们说该模型忽略了软件复杂性和统计测试对软件的测试能力,所得结论具有一定的盲目性[16].对特定软件而言,若统计测试有较高的有效性,则其结果过于保守,浪费了测试资源;若统计测试的有效性较低,则其结果又过于乐观,加大了软件的质量风险.
- PAC 模型仅考虑了不同程序的复杂性而未考虑测试方法的测试能力,其结果偏于保守.对同一个软件,该模型所需要的测试数量远远超出其他两种模型的计算结果.
- Beta 模型对不同程序要求的测试数量依赖于统计测试对各程序的测试能力,能力越强,则需要的测试数量越少.Beta 模型把测试方法的测试能力和软件的可靠性保障能力统一起来.

经典统计模型、PAC 模型和 Beta 模型的比较见表 5.

Table 5 Comparison of different software reliability models

表 5 3 种软件可靠性模型比较

Items	Classical statistical model	PAC model	Beta model
Principle	Operation profile	PAC learning machine	Bayesian analysis, Beta distribution
Assumption	Every fault has the same probability to cause failure	Every fault has the same probability to cause failure	Different faults have independent probability to cause fault
Factor	Construction of operation profile	Software complexity	Effectiveness of test method
Reliability assurance	All software need the same test number	Different software with different complexity need different test number	Different effectiveness need different test number

PAC 模型用程序的伪维数来表示程序的复杂性.直觉上看,对于越复杂的软件,同一种测试方法的测试能力也越差.但计算 Beta 模型和 PAC 模型需要的统计测试数量的 Spearman 相关系数,仅为–0.286,表明程序伪维数与统计测试的测试能力没有直接的关系.可能的原因包括:一方面,软件复杂性对测试方法的测试能力没有直接的影响;另一方面,除了复杂性,还有其他因素影响测试方法的测试能力.

3 结束语

本文通过实验的方式,验证了统计测试对不同软件具有不同的测试能力以及这种测试能力对可靠性评估的影响.本文的结论表明:进行准确的可靠性预测需要综合考虑统计测试对被测软件的测试能力和软件产品的特征,使用测试能力更强的测试方法进行软件质量保证可以用更少的测试数量来达到更高的软件可靠性要求.未来的研究还包括对不同测试准则测试能力的量化研究.

致谢 向对本文的工作给予支持和建议的同行表示感谢.

References:

[1] Musa J, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction and Application. New York: McGraw-Hill, 1987.

[2] Howden WE. Functional Program Testing and Analysis. New York: McGray Hill, 1987.

[3] Chen MH, Lyu MR, Wong WE. Incorporating code coverage in the reliability estimation for fault-tolerant software. In: Proc. of the 16th Symp. on Reliable Distributed Systems. 1997. 45−52.

[4] Zhu H. Toward a relationship between software reliability estimation and complexity analysis. Journal of Software, 1998,9(9): 713−717 (in Chinese with English abstract).

[5] Fenton NE, Pfleeger SL. Software Metrics: A Rigorous & Practical Approach. 2nd ed., Int'l Thomson Computer Press, 1996.

[6] Littlewood B, Lorenzo S. Software reliability and dependability: A roadmap. In: Proc. of the Conf. on the Future of Software Engineering. New York: ACM Press, 2000. 175−188.

[7] Kuball S, May J. Test-Adequacy and statistical testing: Combing different properties of a test-set. In: Proc. of the 15th Int'l Symp. on Software Reliability Engineering (ISSRE 2004). 2004. 161−172.

[8] Zhang YQ, Sun SJ. Software reliability modeling based on unascertained theory. Journal of Software, 2006,17(8):1681−1687 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1681.htm

[9] Chen MH, Mathur AP, Rego VJ. Effect of testing techniques on software reliability estimates obtained using a time_domain model. IEEE Trans. on Reliability, 1995,44(1):97−103.

[10] Richardson DJ, Thompson MC. An analysis of test data selection criteria using the relay model of fault detection. IEEE Trans. on Software Engineering, 1993,19(6):533−553.

[11] Hahn GJ, Sharpiro S. Statistical Models in Engineering. New York: John Wiley & Sons, 1967.

[12] Savchuk VP, Martz HF. Bayes reliability estimation using multiple sources of prior information: Binomial sampling. IEEE Trans. on Reliability, 1994,43(1):138−144.

[13] Littlewood B, Wright D. Some conservative stopping rules for the operational testing of safety-critical software. IEEE Trans. on Software Engineering, 1997,23(11):673−683.

[14] Aristotle Research Group. 2006. http://www.cc.gatech.edu/aristotle/Tools/dist.html

[15] Sheng Z, Xie SQ, Pan CY. Probability Theory and Statistical Methods. Beijing: Higher Education Press, 2002 (in Chinese).

[16] Cai KY. Software Defect and Operational Profile Modeling. Norwell Massachusetts: Kluwer Academic Publishers, 1998.

附中文参考文献:

[4] 朱鸿.软件可靠性估计与计算复杂性的关系浅析.软件学报,1998,9(9):713−717.

[8] 张永强,孙胜娟.基于未确知理论的软件可靠性建模.软件学报,2006,17(8):1681−1687. http://www.jos.org.cn/1000-9825/17/1681.htm

[15] 盛骤,谢式千,潘乘毅.概率论与数理统计.北京:高等教育出版社,2002.

赵亮(1971－),男,山西太原人,博士,工程师,主要研究领域为软件工程,软件测试.

孙家广(1946－),男,教授,博士生导师,主要研究领域为 CAD,软件工程,数据库.

王建民(1968－),男,博士,教授,CCF 高级会员,主要研究领域为工作流技术,软件测试,信息系统.