计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
43(3): 503~508, 2006

考虑测试与运行差别的软件可靠性增长模型

赵 靖 刘宏伟 崔 刚 杨孝宗
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)
(zhaoj@ftcl.hit.edu.cn)

A Software Reliability Growth Model Considering Differences Between Testing and Operation

Zhao Jing, Liu Hongwei, Cui Gang, and Yang Xiaozong
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)

Abstract The testing and operation environment may be essentially different, and thus the fault detection rate of testing is different from that of the operation phase. Based on the G-O model, the representative of non-homogeneous Poisson process (NHPP), the fault detection rate from testing to operation is transformed considering the differences of profile of these two phases, and then a more precise NHPP model (TO-SRGM) considering the differences of fault intensity of testing and operation phases is obtained. Finally, the unknown parameters are estimated by the least-squares method based on normalized data set. Experiments show that the goodness-of-fit of the TO-SRGM is better than those of the G-O model and PZ-SRGM on a data set.

Key words software reliability growth model; non-homogeneous Poisson process; software fault detection; learning curve; goodness-of-fit

摘 要 软件可靠性增长模型中测试阶段和操作运行阶段环境的不同导致了两个阶段故障检测率的不同.在随机过程类非齐次泊松过程(NHPP)中的经典模型G-O模型基础上,考虑运行剖面和测试剖面的不同,对测试阶段和操作运行阶段的故障检测率进行了转化,得到了较好的刻画测试阶段和操作阶段失效率差别的模型(TO-SRGM).最后,通过实例用最小二乘法对此模型的参数进行了估计.实验结果表明,在某些失效数据集上TO-SRGM的拟和效果比G-O模型和PZ-SRGM好.

关键词 软件可靠性增长模型;非齐次泊松过程;软件故障检测;学习曲线;拟和效果

中图法分类号 TP311.5

1 引 言

软件系统是计算机系统的神经中枢,当一个软件开发完成后,能否投入运行,软件的质量要求是关键问题,软件质量的重要特性之一就是软件的可靠性.软件可靠性增长模型可用来估计与软件可靠性直接有关的量,如:残存的故障数以及失效率等[1].一个好的软件可靠性模型不但能够预测发布后的软件在运行阶段是否可靠,而且为软件的成本模型提供了很好的依据[2].绝大多数软件可靠性模型假设软件的测试环境和操作环境是相似的,利用测试获得的软件失效信息,对软件系统的失效过程进行建模,评估软件系统的可靠性,预测软件实际工作时的现场行为.实际上,软件的性能依赖于它的执行环境,软件的执行环境包括操作系统,硬件平台,以及操作剖面.软件的测试剖面很难真实地反映运行剖面,因而在运行和测试阶段软件的故障检测率是不

收稿日期:2005-01-12;修回日期:2005-08-08
基金项目:国家自然科学基金项目(60503015);教育部博士点基金项目(20020213017)

同的[3~7],但很少有软件可靠性模型会考虑到这一点,所以不能够精确的预测软件可靠性.

2 软件运行阶段和测试阶段的故障检测率

软件可靠性增长模型能够如实地反映运行阶段失效率对于维护软件可靠度与软件费用的平衡,决定何时发布软件是非常重要的. 运行阶段和测试阶段的环境不同,检错情况也不同,也即故障检测率不同. 运行阶段的 FDR 由于操作环境、操作剖面和硬件平台以及不同的应用程序等诸多原因很难预测. 有些学者认为建立软件可靠性增长模型时应考虑测试资源的消耗[8~11],并从软件可靠性评价和开发管理方面考虑,根据实测数据和经验讨论了测试阶段和运行阶段的故障检测率的变化趋势. 如图 1 所示:

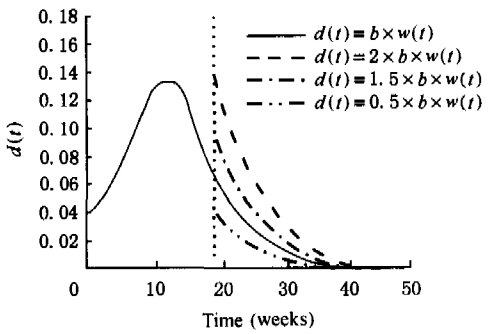

Fig. 1 FDR $d(t)$ of the testing phase and the operation phase.

图 1 测试和运行阶段的故障检测率 $d(t)$

这条曲线描述了从测试到运行阶段故障检测率曲线是先增后减的变化趋势. 许多学者认为软件系统的运行剖面和测试剖面有根本的不同[3~7],在建立软件可靠性增长模型时从测试阶段到运行阶段可以建立一定的联系. 在众多的非齐次泊松可靠性增长模型中,两个最重要的参数就是软件初始故障数和故障检测率. 如果排错过程中不引进新的故障,故障总数是不变的,因此,最重要的参数就是故障检测率[11]. 那么如何从测试阶段的故障检测率(fault detection rate, FDR)得到运行阶段的故障检测率呢? Pham 等人在讨论操作阶段的 FDR 时引进了 β 测试阶段,在 β 测试中引入服从 λ 分布的参数 η 讨论 FDR,不同的应用程序和执行环境导致了 η 的不同[12]. 绝大多数的研究者都假定运行阶段的均值函数和测试阶段均值函数具有同样的结构,所不同的只是对参数的估计[3~7]. 软件测试是软件生命周期中发现故障,并且排除故障,提高软件可靠性的重要手段. 考虑到测试阶段和运行阶段故障检测率的不同,为了使软件可靠性增长模型能够如实地反映运行阶段失效率,并且能为更准确地描述软件成本模型提供依据[12],本文采用 Pham 等人提出的软件生命周期,划分方法如图 2 所示,其中运行阶段包括正式发布软件前给一般用户的 β 测试阶段和发布软件后的最终用户操作阶段,用 β 测试环境模拟实际的终端用户的环境,以 β 测试环境的 FDR 近似替代操作阶段的 FDR,这个 FDR 和净室测试阶段的 FDR 有根本的不同.

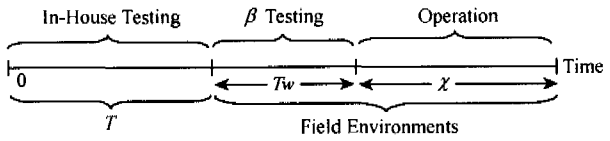

Fig. 2 The developing life cycle of software.

图 2 软件发展生命周期

多数 NHPP 类软件可靠性增长模型中的经典模型假定测试环境和操作环境是相同的,其故障检测率函数主要有下面 3 种[13]:

1) 常数 b;

2) 关于测试时间的增函数;

3) 关于测试时间的减函数.

出现较早的 NHPP 类的软件可靠性增长模型一般假设故障检测率函数是一个常数. 新近提出的一些模型则假设故障检测率函数是一个增函数,这个增函数体现了测试者的学习过程. 几乎所有的研究者都认为在实际测试时,学习的影响非常重要. 如果只考虑故障本身的性质,软件中残留的故障被检测的概率越来越低,故障检测率函数应该是一个递减的函数. 实际上,测试阶段软件的故障检测率变化应该是这两方面作用的结果,从而呈现出先增后减的形状. 测试阶段软件固有的故障检测率的这种变化趋势可表示如下[14]:

$$p_0(t) = b\exp(-\beta_1 t). \tag{1}$$

另一方面,表现测试者学习过程用数学模型表示如下[15]:

$$p_1(t) = \frac{1+\beta}{1+\beta\exp(-b(1+\beta)t)}. \tag{2}$$

测试阶段软件的故障检测率是这两个过程综合作用的结果. 被测软件的故障检测率可用下面的方程表示:

$$b(t) = p_0(t)p_1(t) = \frac{b(1+\beta)\exp(-\beta_1 t)}{1+\beta\exp(-b(1+\beta)t)}. \tag{3}$$

令 $\beta_0=b(1+\beta)$,对 $b(t)$进行讨论:

1) 若 $\beta_0\leqslant\beta_1$ 或 $\beta_0>\beta_1$ 并且 $\beta(\beta_0-\beta_1)-\beta_1\leqslant 0$,则 $t=0$ 时,函数 $b(t)$取得最大值 b. FDR 是一个递减的函数.

2) 若 $\beta_0>\beta_1$ 并且 $\beta(\beta_0-\beta_1)-\beta_1>0$,则当

$$t=\frac{1}{\beta_0}\ln\frac{\beta(\beta_0-\beta_1)}{\beta_1} \tag{4}$$

时,$b(t)$取得最大值,其值为

$$b_{\max}=(\beta_0-\beta_1)\left(\frac{\beta_1}{\beta(\beta_0-\beta_1)}\right)^{\frac{\beta_1}{\beta_0}}. \tag{5}$$

故障检测率是一个先增后减的函数. 下面对测试阶段和运行阶段的故障检测率分别讨论如下:

测试阶段的故障检测率 FDR_c 为

$$FDR_c=b(t)=\frac{b(1+\beta)\exp(-\beta_1 t)}{1+\beta\exp(-b(1+\beta)t)},t\leqslant T. \tag{6}$$

从测试的过程来看,测试阶段是根据测试剖面,开发测试用例,使用多种测试方法,如边界值测试、等价类测试,考虑到测试覆盖率的基于路径测试等,测试者的学习能力对测试阶段故障检测率有一定的影响,当预测的现场行为达到用户所确认的指标时就发布给用户到现场运行的操作阶段. 实际上,软件测试致力于发现软件中的故障,测试技术的有效使用导致非随机测试软件的失效强度比现场情况下要高[13]. 有些学者建立软件可靠性模型考虑测试资源消耗时,都假设测试阶段故障检测率和操作阶段的故障检测率 $d(t)$有如下性质[8]:

$$b\times w(t)\geqslant b_{op}\times w_{op}(t), \tag{7}$$

其中 b 和 b_{op}分别代表测试阶段和操作阶段每单位测试资源的故障检测率,$w(t)$和 $w_{op}(t)$分别代表到时间 t 所消耗的资源. 因此运行阶段(β 测试,操作阶段)的故障检测率 FDR_o:

$$FDR_o=b(t)=b^{\circ}\exp(-\beta_2 t),t>T. \tag{8}$$

3 运行阶段和测试阶段故障检测率的软件可靠性模型

G-O 模型于 1979 年由 Goel 和 Okumoto 提出,是关于连续时间的 NHPP(non-homogeneous Poisson process)模型中的经典模型. 在很多具体的应用中,G-O 都工作得很好[2]. G-O 的基本假设如下:

1) 软件在与预期的操作环境相似的条件下运行;

2) 软件的失效遵循 NHPP 过程;

3) 故障一旦被发现,立即被排除,并且不引入新的故障;

4) 在任何时候软件的失效强度与软件中隐藏的故障数成正比;

5) 每个故障的严重性和被检测到的可能性大致相同.

根据前面的讨论,软件测试阶段和运行阶段的环境是不同的,并且测试阶段和运行阶段的故障检测率是不同的,因此本模型修改 G-O 模型的假设 1)和 5),测试阶段和运行阶段除满足 G-O 模型的假设 2)3)4)以外,对其进行了分别的假设.

测试阶段的假设满足:

① 测试阶段测试人员测试能力的提高与测试时间的关系满足一个非递减的 S-形曲线;

② 软件中潜伏故障的固有检测率是一个随时间递减的函数.

运行阶段的假设:

① 软件运行阶段严格按照操作剖面进行;

② 故障检测率只表现故障本身固有的故障检测率,是一个随时间递减的函数;

根据假设条件,可以推断测试阶段均值函数如下:

$$\frac{dm(t)}{dt}=b(t)[a-m(t)]=\frac{b(1+\beta)\exp(-\beta_1 t)}{1+\beta\exp(-b(1+\beta)t)}[a-m(t)], \tag{9}$$

解方程(9)可得:

$$m(t)=a(1-\exp(-B^*(t))),t\leqslant T, \tag{10}$$

其中,$B^*(t)=\int_0^t b(\tau)d\tau,\tau\leqslant T.$ (11)

将式(6)代入式(11),可得

$$B^*(t)=\int_0^t\frac{\beta_0\exp(-\beta_1\tau)}{1+\beta\exp(-\beta_0\tau)}d\tau=\sum_{k=0}^{+\infty}\frac{\beta_0(-\beta)^k(1-\exp(-(\beta_1+k\beta_0)t))}{\beta_1+k\beta_0},t\leqslant T, \tag{12}$$

因此 $$m(t)=a\left(1-\exp\left(-\sum_{k=0}^{+\infty}\frac{\beta_0(-\beta)^k(1-\exp(-(\beta_1+k\beta_0)t))}{\beta_1+k\beta_0}\right)\right), \tag{13}$$

其中,$t\leqslant T$.

那么,对于运行阶段的均值函数,假设运行阶段也可以被看做一个故障检测过程,而且此过程可以用 NHPP 表示[16],假设运行阶段的故障率为$\lambda^{\circ}(t)$,那么运行阶段的均值函数 $m^{\circ}(t)$为

$$m^{\circ}(t) = \int_{T}^{t}\lambda^{\circ}(t)\mathrm{d}t. \tag{14}$$

推导如下:

$$\lambda^{\circ}(t) = \frac{\mathrm{d}m^{\circ}(t)}{\mathrm{d}t} = b(t)(a - m(T) - m^{\circ}(t)), t > T, \tag{15}$$

将式(8)代入式(15)得到:

$$\frac{\mathrm{d}m^{\circ}(t)}{\mathrm{d}t} = b^{\circ}\exp(-\beta_2 t)(a - m(T) - m^{\circ}(t)), t > T, \tag{16}$$

$$m^{\circ}(t) = a - m(T) + c\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 t)\right). \tag{17}$$

因为

$$\lim_{t\to 0} m^{\circ}(T + x) = 0,$$

从而,得

$$c = \frac{m(T) - a}{\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 T)\right)}, \tag{18}$$

从而,可得测试和运行阶段的均值函数如下:

$$m(t) = \begin{cases} a\left(1 - \exp\left(-\sum_{k=0}^{+\infty}\frac{\beta_0(-\beta)^k(1 - \exp(-(\beta_1 + k\beta_0)t))}{\beta_1 + k\beta_0}\right)\right), t \leqslant T, \\ a + \frac{m(T) - a}{\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 T)\right)}\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 t)\right), t > T. \end{cases} \tag{19}$$

式(19)中包含了一个从 0~+∞的累加表达式,很难用数值的方法进行估算,不过容易知道,当 $\beta \leqslant 1$ 时:

$$\lim_{k\to\infty}\frac{\beta_0(-\beta)^k(1 - \exp(-(\beta_1 + k\beta_0)t))}{\beta_1 + k\beta_0} = 0. \tag{20}$$

因此,在满足方程(19)和用户精度的前提下,在实际应用中可以用下面的方程近似的表示 TO-SRGM 的故障累计数期望函数方程.

$$m(t) = \begin{cases} a\left(1 - \exp\left(-\sum_{k=0}^{M}\frac{\beta_0(-\beta)^k(1 - \exp(-(\beta_1 + k\beta_0)t))}{\beta_1 + k\beta_0}\right)\right), t \leqslant T, \\ a + \frac{m(T) - a}{\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 T)\right)}\exp\left(\frac{b^{\circ}}{\beta_2}\exp(-\beta_2 t)\right), t > T. \end{cases} \tag{21}$$

4 实例验证

为了验证我们建立的软件可靠性模型,应用标准化数据[12]进行分析,对这个模型的拟和效果进行了分析,分析中使用误差平方和 *SSE* 和回归曲线方程的相关指数 *R-square* 度量曲线拟合效果,这两个指标分别定义如下:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{m}(t_i))^2, \tag{22}$$

$$R\text{-}square = \frac{\sum_{i=1}^{n}(\hat{m}(t_i) - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{23}$$

其中:n 表示失效数据集中失效样本的数量;$\hat{m}(t_i)$表示到 t_i 时刻为止故障累计数的估算值;y_i 表示到 t_i 时刻为止故障累计数的实测值.

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n}y_i. \tag{24}$$

SSE 的值越小,曲线拟合得越好;*R-square* 的值越接近于 1,曲线拟合得越好.

首先在标准化数据集上通过实验确定 M 的值.实验结果如表 1 所示.从表 1 可以看出,通过最小二乘法估算参数,当 $M \geqslant 9$ 时,TO-SRGM 模型的参数趋于稳定,因此取 $M = 9$,这个软件系统的参数表示为

$$a = 1.0136, \beta_0 = 0.000108, \beta = 0.1065,$$
$$\beta_1 = 0.0983, \beta_2 = 0.01, b^{\circ} = 0.000005423.$$

在这组数据上应用 G-O 模型和 PZ-SRGM[17]模型进行曲线拟合,这两个模型的数据拟合情况如表 2 所示.图 3 表示模型 G-O,PZ-SRGM,TO-SRGM 以及标准化的实测数据的比较,分析结果表明,TO-SRGM 的拟合情况比 G-O 模型和 PZ-SRGM 的好.

Table 1 Influence of M on TO-SRGM

表 1 M 对 TO-SRGM 参数影响

Parameter	$M=0$	$M=1$	$M=2$	$M=3$	$M=4$	$M=5$	$M=6$	$M=7$	$M=8$	$M=9$	$M=10$	$M=11$	$M=12$
a	0.9876	0.9975	0.8734	0.8234	0.7541	1.0031	1.0026	1.0018	1.0028	1.0136	1.0136	1.0136	1.0136
β_0	9.653E-5	7.539E-5	7.500E-5	7.523E-5	7.523E-5	7.694E-5	9.978E-5	9.969E-5	9.959E-5	1.008E-4	1.008E-4	1.008E-4	1.008E-4
β	0.1976	0.1945	0.3879	0.1856	0.1432	0.1032	0.1354	0.1074	0.1075	0.1065	0.1065	0.1065	0.1065
β_1	0.1000	0.9856	0.9650	0.7856	0.7963	0.7847	0.7857	0.0999	0.0991	0.0993	0.0983	0.0983	0.0983
β_2	0.0096	0.0101	0.0121	0.0099	0.0056	0.0052	0.0099	0.0101	0.0101	0.0100	0.0100	0.0100	0.0100
b^0	8.549E-6	8.200E-6	7.489E-6	7.520E-6	5.356E-6	6.869E-6	5.424E-6	5.426E-6	5.348E-6	5.422E-6	5.425E-6	5.423E-6	5.422E-6

Table 2 Comparison Between the Goodness-of-Fit of the Three Models

表 2 3个模型数据拟合情况对比

Comparison Criteria	G-O model	PZ-SRGM	TO-SRGM
SSE	0.3182	0.3198	0.3115
R-square	0.9935	0.9758	0.9968

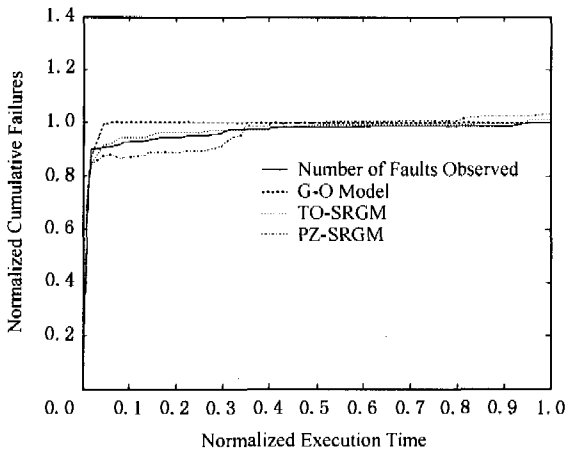

Fig. 3 Comparison among the normalized measured data, the G-O model, the PZ-SRGM and the TO-SRGM.

图 3 标准化的实测数据、G-O 模型、PZ-SRGM、TO-SRGM 比较

5 结 论

本文考虑了运行剖面和测试剖面的不同,通过测试阶段到操作阶段故障检测率的变换,提出了一个更为精确的考虑测试阶段和运行阶段故障率不同的软件可靠性增长模型(TO-SRGM),最后以一个实例比较了经典模型 G-O,PZ-SRGM,TO-SRGM,结果表明,TO-SRGM 模型的拟合情况比 G-O 模型和 PZ-SRGM 的好.

参 考 文 献

1 Xuemei Zhang, Xiaolin Teng, Hoang pham. Considering fault removal efficiency in software reliability assessment. IEEE Trans. Systems, Man, and Cybernetics, 2003, 33(1): 114~120

2 H. Pham, X. Zhang. A software cost model with warranty and risk costs. IEEE Trans. Computers, 1999, 48(1): 71~75

3 J. D. Musa. Operational profiles in software reliability engineering. IEEE Software Magazine, 1993, 10(2): 14~32

4 J. D. Musa. Sensitivity of field failure intensity to operational profile errors. The 5th Int'l Symposium on Software Reliability Engineering, Montery, CA, 1994

5 G. Q. Kenney. Estimating defects in commercial software during operational use. IEEE Trans. Reliability, 1993, 42(1): 107~115

6 Y. Chen. Modeling software operational reliability via input domain-based reliability growth model. The 28th Int'l Symposium on Fault-Tolerant Computing (FTCS), Munich, Germany, 1998

7 J. D. Musa. Adjusting measured field failure intensity for operational profile variation. The 5th Int'l Symposium on Software Reliability Engineering, Montery, CA, 1994

8 C. Y. Huang, S. Y. Kuo. Analysis of a software reliability growth model with logistic testing-effort function. The 8th Int'l Symposium on Software Reliability Engineering (ISSRE 1997), Albuquerque, NM, USA, 1997

9 Sy-Yen Kuo, Chin-Yu Huang, Michael R. Lyu. Framework for modeling software reliability, using various testing-efforts and fault-detection rates. IEEE Trans. Reliability, 2001, 50(3): 310~320

10 C. Y. Huang, J. H. Lo, S. Y. Kuo. Pragmatic study of parametric decomposition models for estimating software reliability growth. The 9th Int'l Symposium on Software Reliability Engineering (ISSRE 1998), Paderborn, 1998

11 Chin-Yu Huang, Sy-Yen Kuo, Michael R. Lyu, *et al*. Quantitative software reliability modeling from testing to operation. The 11th Int'l Symposium on Software Reliability Engineering (ISSRE'00), San Jose, CA, 2000

12 Xiaolin Teng, H. Pham. A software cost model for quantifying the gain with considering of random field environments. IEEE Trans. Computers, 2004, 53(3): 380~384

13 Liu Hongwei. Research on the software reliability growth model of non-homogeneous Poisson process: [Ph. D. disseratation].

Harbin: Harbin Institute of Technology, 2004 (in Chinese)
(刘宏伟. 非齐次泊松过程类软件可靠性增长模型研究: [博士学位论文]. 哈尔滨: 哈尔滨工业大学, 2004)

14 S. Yamada, M. ohba, S. Osaki. S-shaped software reliability growth modeling for software error detection. IEEE Trans. Reliability, 1983, 32(5): 475~484

15 H. Pham, L. Nordmann, X. M. Zhang. A general imperfect-software-debugging model with S-shaped fault-detection rate. IEEE Trans. Reliability, 1999, 48(2): 169~175

16 Hiroshi Ohtera, Shigeru Yamada. Optimal software-release time considering an error-detection phenomenon during operation. IEEE Trans. Reliability, 1990, 39(5): 596~599

17 H. Pham, X. Zhang. An NHPP software reliability model and its comparison. International Journal of Reliability, Quality and Safety Engineering, 1997, 4(3): 269~282

Zhao Jing, born in 1973. Ph. D. candidate. Her main research interests include software testing, software reliability evaluation, fault tolerance computing and wearable computing.

赵婧,1973 年生,博士研究生,主要研究方向为软件测试、软件可靠性评估、容错计算、可穿戴计算.

Liu Hongwei, born in 1971. Ph. D. associate professor. His main research interests include software testing, software reliability evaluation, fault tolerance computing and wearable computing.

刘宏伟,1971 年生,博士,副教授,主要研究方向为软件测试、软件可靠性评估、容错计算、可穿戴计算.

Cui Gang, born in 1949. Professor and Ph. D. supervisor. His main research interests include fault tolerance computing, wearable computing, software testing, and software reliability evaluation.

崔刚,1949 年生,教授,博士生导师,主要研究方向为容错计算、可穿戴计算、软件可靠性评估.

Yang Xiaozong, born in 1939. Professor and Ph.D. supervisor. His main research interests include wearable computing, fault tolerance computing, software testing, and software reliability evaluation.

杨孝宗,1939 年生,教授,博士生导师,主要研究方向为可穿戴计算技术、容错计算技术、软件可靠性技术等.

Research Background

This work is supported by the National Nature Science Foundation of China (60503015). The main objective of this project is to evaluate software reliability considering differences between testing and operation, and influence on software reliability when testing environment does not match operational environment.

"System Fault Tolerance Ability Evaluation (grant No. 41316.1.2)" is a pre-research project for the Tenth Five Year Plan of PLA General Equipped Department. The main objective of this project is to evaluate the fault tolerance ability, fault tolerance mechanism, and reliability of computer system. Software reliability evaluation technique is one of the main areas. In the middle of examining held by PLA General Equipped Department, this project is ranked as excellent.

Dependable computing is highly considered by International researchers. This research area includes fault tolerance, reliability, availability, etc. Software reliability is an important aspect in this area. Supported by the Ph.D. Programs Foundation of Ministry of Education of China (research on dependability evaluation algorithms of computer system, grant No. 20020213017), we have worked on the evaluation algorithms for fault-tolerance and dependability. Some progress has been achieved. Furthermore, we will develop evaluation algorithm for dependability of very large scale network system.

This paper aims at software reliability evaluation and prediction.