连续执行软件可靠性验证测试方法*)

覃志东 雷 航 桑 楠 熊光泽

(电子科技大学计算机学院实时系统研究室 成都610054)

摘 要 针对基于经典统计假设检验的固定期测试方法测试持续期太长,难以满足现代连续执行软件可靠性验证测试的需要,提出了一种基于经验贝叶斯统计推断的连续执行软件可靠性验证测试方法。该方法通过可靠性增长测试阶段的测试记录获得软件失效强度的先验分布,从而得到所需要的验证测试持续期,并结合验证测试过程中的具体情况,提出了先验动态整合的方法。实验表明,该方法在不降低验证测试结果可信性的条件下,能有效地减少可靠性验证测试的持续期。

关键词 软件可靠性,可靠性验证测试,经验贝叶斯方法,连续执行软件

Reliability Demonstration Testing Method for Continuous Execution Software

QIN Zhi-Dong LEI Hang SANG Nan XIONG Guang-Ze

(Real-time System Lab, Computer College of UESTC, Chengdu 610054)

Abstract To the question that fixed duration testing method, which bases on the classical statistical hypothesis testing, can't fulfill the requirements of reliability demonstration testing for modern continuous execution software due to the long testing duration, a new method to demonstrate the reliability for continuous execution software is presented basing on the empirical Bayesian statistical reference. Firstly, the prior distribution of software failure intensity is derived on the basis of analyzing the testing records of software reliability growth testing. Then, the reliability demonstration testing duration can be calculated. Finally, the prior knowledge dynamic integration method taking the specific testing information into account is provided. Experiment shows that the method provided above is effective to reduce the testing duration without decreasing the confidence level in the testing results.

Keywords Software reliability, Reliability demonstration testing, Empirical Bayesian statistical reference, Continuous execution software

1 引言

随着高性能处理器和海量存储介质的不断推陈出新,以及结构化、对象化和构件化等软件技术的应用,各种高度逻辑复杂性的软硬件集成系统日益广泛地被运用在航空、航天、核能、交通、金融等安全关键领域,并起着灵魂与中枢的作用。如,美国航天飞机上的飞行软件达50万行源代码;F-22战斗机上的飞行软件更达150多万行源代码[1]。我国发射的神舟5号飞船上就有×××万行源代码和1000多颗微处理器;而且,鉴于软件的灵活性和易修改性,在神舟二期工程中,软件的比重还要增大[2]。但是,由于软件固有的逻辑复杂性以及人类对这种逻辑复杂性的不可控制性,软件失效已经成为系统瘫痪的主要原因,而由软件失效导致的重大事故也层出不穷。如,亚利安娜5号火箭发射失败[3]、Therac-25放射性治疗仪事故[4]等。美国国防部和NASA统计发现,当今武器系统和航天项目中的软件可靠性比硬件系统的可靠性低一个数量级[5]。如何保证和提高软件的质量,特别是软件的可靠性,日益受到理论界和产业界的重视。围绕这一中心议题,于1990年前后,形成了软件可靠性工程的概念(Software Reliability Engineering),即:预计、度量和管理以软件为基础的系统的可靠性,以最大限度地满足用户需求的应用科学[6]。

软件可靠性工程包括:可靠性指标的确定与分配、可靠性设计、可靠性管理与控制、可靠性估计与预计、可靠性增长测试与验证测试。大量的研究主要集中在利用软件开发阶段的失效数据进行软件可靠性增长建模,从而对软件可靠性进行估计和预测,以支持相应的决策判断[7~13]。但是,这些模型对软件可靠性评价的精度和稳定性差,可信性不高,所以,在软件验收阶段,需要通过统计测试的方法,对软件的可靠性进行可靠性验证测试(Software Reliability Demonstration Testing, SRDT),以判断是否达到了合同上规定的软件可靠性指标。

美国军事手册US MIL-HDBK-781A规定了两种可靠性验证测试方法:序列测试(Probability Radio Sequential Testing, PRST)和固定时间测试(Fixed During Testing, FDT)[14]。但是,在安全关键领域,由于规定的软件可靠性指标是非常高的,验证测试所需要的测试用例量非常大,要求的测试持续期非常长,以至于对于某些生命安全关键软件的可靠性指标无法进行验证测试[15]。因而,如何在不降低SRDT可信性的情况下,减少验证测试用例量是解决高可靠性指标验证问题的有效手段。Tal等人在PRST基础上提出了单风

*)本课题受国防预研基金(41315040106)、国家高技术研究发展863计划专项经费(2003AA1Z2210)和电子科技大学优秀青年教师基金(L08010601YF020806)资助。覃志东 博士生,研究方向为软件测试理论、技术与软件可靠性工程。雷 航 博士,教授,研究方向为软件可靠性工程。桑 楠 副教授,主要研究方向为可信性计算理论与应用。熊光泽 教授,博士生导师,主要研究领域为实时计算系统与应用。

险序列测试方法(Single Risk Sequential Test,SRST)[16,17],能用较少的测试用例获得同样的测试可信性;覃志东等人针对实时多任务软件的结构和运行特性,提出了实时多任务软件的 SRDT 方法[18];同时,覃志东等人根据序列执行软件二项取样的特性,发展了序列执行软件的经验贝叶斯测试方法[19,20]。这些方法,在实践中取得了很好的效果,但是,这些方法都是针对序列执行软件的 SRDT,而有些连续执行软件,如操作系统,就无法采用序列测试方法,只能采用固定时间测试方法。如何改善这类连续执行期软件的 SRDT 方法,有效地缩短验证测试过程的持续期,具备很高的理论和实用价值。

本文在剖析已有的连续执行软件的可靠性验证测试方法的基础上,假定连续执行软件的失效分布模型,根据软件测试的实际,建立了连续执行软件的可靠性验证测试的经验贝叶斯方法,最后给出了实例。

2 基于经典统计假设检验的 FDT 方法

连续执行软件可靠性可以用多种软件可靠性参数来度量,如:失效率、失效强度、平均失效前时间(Mean Time to Failure, MTTF)、平均失效间隔时间(Mean Time Between Failure, MTBF)等,这些参数可以相互转换。在软件可靠性工程实践中,软件可靠性参数的选取必须结合软件的特点和整个系统的要求。

记,θ_0为合同规定 MTBF,θ_1为对 MTBF 可容忍的较低的限制,而且($\theta_1<\theta_0$);E_1表示软件在测试过程中失效数不超过 s 的事件;$\bar{E}_1$表示事件 E_1的补事件。

假设:①在 SRDT 过程中,软件错误不排除,但是,当 SRDT 过程结束以后,所有被发现的错误都被排除。

②软件在时间区间$(0,t]$遵循均值为 t/θ 的泊松分布。

③θ_0和 θ_1的值在测试之前都已经具体指明。

在测试完成以后,当 $\theta=\theta_0$时,事件 $\bar{E}_1$发生的概率为

$$P(\bar{E}_1|\theta_0)=\sum_{i=s+1}^{\infty}\frac{(t/\theta_0)^i}{i!}e^{-\frac{t}{\theta_0}} \tag{1}$$

同样,当 $\theta=\theta_1$时,事件 E_1发生的概率为

$$P(E_1|\theta_1)=\sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-\frac{t}{\theta_1}} \tag{2}$$

在经典统计学中,式(1)表示生产者风险,即当软件的可靠性已经达到指标,但在测试过程中,错误数超过预期数目,而错误地拒绝该软件;式(2)表示消费者风险,表示事实上软件的可靠性没有达到指标,但在测试过程当中,发现的错误数不超过预期数目,从而错误地接受了该软件。当给定生产者和消费者风险值 α 和 β 后,解式(3)便可以求得测试验证相应的可靠性指标所需要的持续时间 t:

$$\begin{cases}\sum_{i=s+1}^{\infty}\frac{(t/\theta_0)^i}{i!}e^{-\frac{t}{\theta_0}}\leqslant\alpha\\ \sum_{i=0}^{s}\frac{(t/\theta_1)^i}{i!}e^{-\frac{t}{\theta_1}}\leqslant\beta\end{cases} \tag{3}$$

在安全关键软件测试中,为了尽最大可能地保证整个软件系统的可靠性,要尽量减少消费者风险,在此基础上再考虑减少生产者风险。另外,对于安全关键软件,规定在 SRDT 过程中,不应该发现失效,如果失效,便拒绝该软件。从而,对于安全关键软件,对于给定的可靠性指标(θ_1,β),所需要的无失效测试持续期为

$$t=-\theta_1\ln\beta \tag{4}$$

针对不同的(θ_1,β),通过式(4)计算出了相应的无失效测试持续时间如表1所示。

表1 无失效测试时间与可靠性指标之间的关系(时间单位:小时)

θ_1 \ β	0.1	0.01	0.001	0.0001
1000	2302.6	4605.2	6907.8	9210.3
10000	23026	46052	69078	92103
100000	230260	460520	690780	921030
…	…	…	…	…

可见,对于连续执行软件,在可靠性验证测试过程中,无失效测试时间随着待验证指标的提高而急剧增大。如:假设待验证的软件可靠性指标为(10000,0.0001),所需要的无失效测试时间为92103小时,约10.5年,即使采用10个软件版本并行运行测试,也需要1.05年的时间。显然,这给实际的软件可靠性验证测试工作带来很大的不方便。如何从根本原理上改变这种状况,具有很大的理论和实用价值。

3 FDT 的经验贝叶斯方法

3.1 泊松分布及其共轭先验分布

设软件的失效强度为 λ,那么在时间区间$(0,t]$,软件发生失效的次数 X 是一个随机变量,且软件失效次数为服从参数为 λt 的泊松分布:

$$P(x=k)=\frac{(\lambda t)^k}{k!}e^{-\lambda t}\quad(k=0,1,2,\cdots) \tag{5}$$

其中 λ 表示软件的失效强度,也就是软件在单位时间失效的平均次数。可以根据软件在软件可靠性增长测试阶段留下的测试记录估计出 λ 的分布。

先验分布的选取有多种方法,其中,共轭先验分布由于其良好的数学表达,具有很广泛的应用基础。

定义 设样本 $x_1,\cdots,x_n$ 对参数 θ 的条件分布为 $p(x_1,\cdots,x_n|\theta)$,先验分布 $\pi(\theta)$称为 $p(x_1,\cdots,x_n|\theta)$的共轭分布,如果 $\pi(\theta)$决定的后验分布密度 $h(\theta|x_1,\cdots,x_n)$与 $\pi(\theta)$是同一个类型。

根据定义,泊松分布的共轭先验分布为 *Gamma* 分布。

证明:对于泊松分布:$p(x=k)=\frac{(\lambda t)^k}{k!}e^{-\lambda t}$,假定其参数 λ 的先验分布为

$$\pi(\lambda)=Gamma(a,b)=\frac{b^a}{\Gamma(a)}\lambda^{a-1}e^{-b\lambda}\propto\lambda^{a-1}e^{-b\lambda} \tag{6}$$

那么,其后验分布为

$$h(\lambda|x_1,\cdots,x_n)\propto\pi(\lambda)p(x_1,\cdots,x_n|\lambda)\propto\lambda^{a-1}e^{-b\lambda}\prod_{i=1}^{n}\frac{(\lambda t)^{k_i}e^{-\lambda t}}{k_i!}\propto\lambda^{a+\sum_{i=1}^{n}k_i-1}e^{-(b+nt)\lambda}=Gamma(a+\sum_{i=1}^{n}k_i,b+nt) \tag{7}$$

所以,泊松分布的共轭先验分布为 *Gamma* 分布。

3.2 先验分布参数的求解

由上面知道,软件的失效强度 λ 遵从 $Gamma(a,b)$分布。但是,如图1所示,对于超参数 a,b 取不同的值,将导致失效强度 λ 的分布发生很大的变化。所以,不能主观武断地给定超参数 a,b 的值,而应该根据软件在进行 SRDT 之前的测试过程中的表现情况,作为先验知识,来确定先验分布的超参数 a,b。在软件验收阶段进行 SRDT 之前,都要经过相应的软件可靠性增长测试,留下大量的测试记录。对于连续执行软件来说,留下的会是每两次失效之间的执行时间 $T_1,\cdots,T_i,\cdots,T_n$。相应地给出了软件失效率的经验样本值,可以采用参数化

方法对超参数 a,b 进行估计。

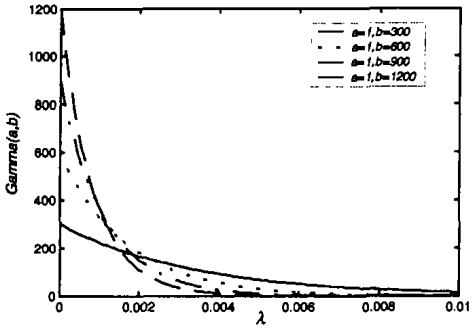

图1　不同超参数对应的先验分布

与经典统计学认为失效率 λ 是常量不同，贝叶斯统计学认为 λ 是随机变量，软件在时间区间$(0,t]$内失效次数为 k 的概率，是随机变量 λ 的条件概率：

$$p(x=k|\lambda)=\frac{(\lambda t)^k}{k!}e^{-\lambda t}\quad(k=0,1,2,\cdots)\tag{8}$$

其边缘分布

$$m(x)=\int_0^{+\infty}\pi(\lambda)p(x|\lambda)\mathrm{d}\lambda=\int_0^{+\infty}\frac{b^a}{\Gamma(a)}\lambda^{a-1}e^{-b\lambda}\frac{(\lambda t)^x}{x!}e^{-\lambda t}\mathrm{d}\lambda=\frac{b^a t^x(a+x)\cdots(a+1)}{x!(b+t)^{a+x}}\tag{9}$$

那么 $m(x)$ 相应的一、二阶矩为

$$E(x)=\sum_{x=0}^{+\infty}xm(x)=\sum_{x=0}^{+\infty}x\int_0^{+\infty}\pi(\lambda)\frac{(\lambda t)^x e^{-\lambda t}}{x!}\mathrm{d}\lambda=\frac{at}{b}\tag{10}$$

$$E(x^2)=\sum_{x=0}^{+\infty}x^2m(x)=\sum_{x=0}^{+\infty}x^2\int_0^{+\infty}\pi(\lambda)\frac{(\lambda t)^x e^{-\lambda t}}{x!}\mathrm{d}\lambda=\frac{at}{b}+\frac{(a+1)at^2}{b^2}\tag{11}$$

取 t 为相对失效样本 $T_1,\cdots,T_i,\cdots,T_n$ 的一个较大的时间，那么$(0,t]$这段时间，软件失效数的经验样本值为 $t/T_1,\cdots,t/T_i,\cdots,t/T_n$。用经验样本值去估计 $E(x)$ 和 $E(x^2)$，联立式(10)、(11)，便可以估计出超参数 a,b 的值为 a_0,b_0。从而得到软件失效强度 λ 的先验分布为

$$\pi(\lambda)=Gamma(a_0,b_0)=\frac{b_0^{a_0}}{\Gamma(a_0)}\lambda^{a_0-1}e^{-b_0\lambda}\tag{12}$$

3.3　先验动态整合

当确定软件失效强度 λ 的先验分布后，就可以对软件的可靠性指标进行验证。本方法选取可靠性指标的形式为(λ_0,β_0)，表示软件的失效强度 $\lambda\leqslant\lambda_0$，且消费者风险 $\beta\leqslant\beta_0$。其它的可靠性指标形式可以转化为本文的表达形式。

当软件持续运行时间 t，观察到 r 次失效，则由式(7)可知，后验分布为

$$h(\lambda|r,t,a_0,b_0)=Gamma(a_0+r,b_0+t)\tag{13}$$

特别地，当 $r=0$，即在验证测试过程中不容忍失效时，软件失效强度 λ 的后验分布为

$$h(\lambda|0,t,a_0,b_0)=Gamma(a_0,b_0+t)\tag{14}$$

那么，满足相应可靠性指标的软件无失效验证测试时间 t_1 取下式中的 t：

$$\int_0^{\lambda_0}=Gamma(a_0,b_0+t)\mathrm{d}\lambda=1-\beta\tag{15}$$

如果软件在验证测试过程中顺利地通过验证测试持续期 t_1 而无失效，说明软件已经达到了规定的可靠性指标，验收合格；相反，如果软件在 tf_1 时刻$(tf_1<t_1)$发生失效，则说明软件达不到规定的可靠性指标，测试工作必须停止下来固定并排除错误，然后进行第二次 SRDT。在决定第二次验证测试持续期之前，可以把第一次测试的结果作为先验信息的一部分综合考虑。所以，对于同样的可靠性指标，第二次无失效验证测试时间 t_2 取满足式(16)中的 t：

$$\int_0^{\lambda_0}gamma(a_0+1,b_0+tf_1+t)\mathrm{d}\lambda=1-\beta\tag{16}$$

这个过程一直持续下去，如果在此过程中已第 j 次发现错误，那么第 $j+1$ 次无失效验证测试时间 t_{j+1} 取下式中的 t：

$$\int_0^{\lambda_0}Gamma(a_0+j,b_0+\sum_{i=0}^{j}tf_i+t)\mathrm{d}\lambda=1-\beta\tag{17}$$

令 $E_{j+1}=\sum_{i=0}^{j}tf_i+t_{j+1}$ 表示在软件可靠性验证测试过程中观察到第 j 次软件失效后，软件必须经历的总的测试时间，包括前面 j 次被中断的测试时间和第 $j+1$ 次需要的无失效验证测试时间 t_{j+1}。那么，式(17)可以表达成：

$$\int_0^{\lambda_0}Gamma(a_0+j,b_0+E_{j+1})\mathrm{d}\lambda=1-\beta\tag{18}$$

在验证测试过程中，可以根据式(18)计算出对应的第 $j+1$ 次软件需要的总的测试时间 E_{j+1}，然后再计算出具体的第 $j+1$ 次无失效验证测试持续期：

$$t_{j+1}=E_{j+1}-\sum_{i=0}^{j}tf_i\tag{19}$$

显然，由式(18)得到的总的测试时间 E_{j+1} 是一定的，所以，由式(19)可知：前面 j 次失效情况直接影响着第 $j+1$ 次的验证测试持续时间的长短。

4　实例研究

表2　先验样本数据(时间单位：小时)

软件失效间隔时间 T_i	软件失效数 t/T_i
909.1	110
990.1	101
1123.6	89
877.2	114
1075.3	93
1000	100
1063.8	94
847.5	118
1010.1	99
757.6	132

本研究室研发的某一监测系统软件，无法确定其每一次操作的平均时间，属于连续执行软件类。其合同可靠性指标为$(\lambda,\beta)=(10^{-3},0.01)$。在系统测试阶段，专门开发了其运行剖面，并严格按照运行剖面抽样产生测试用例，进行了可靠性增长测试。选择最后10组失效间隔时间作为 T_i 的经验样本值，选取 $t=100000$小时并估计出在这段时间软件失效数的经验样本值 t/T_i 如表2所示。

根据以上的数据可以计算出软件失效强度的先验分布的超参数为 $a_0=1,b_0=952.4$。那么，由式(18)可以计算出软件总的验证测试时间与软件失效次数之间的关系如表3所示：

表3　软件总测试时间与失效次数之间的关系(时间单位：小时)

失效次数，j	总测试时间，E_{j+1}
0	3652.8
1	5685.9
2	7453.5
3	9092.7
4	1065.2
5	12156.1
…	…

由表3可知，当进行第一次验证测试时，需要的验证测试持续期为 $t_1=3652.8$小时，当软件无失效通过这么长的测试期，则软件达到规定的可靠性指标，验收合格；如果软件在时刻 $tf_1=2000$小时发生失效，则说明软件可靠性达不到规定的可靠性指标，整个验证测试过程应该停止下来排除错误。那么第二次验证测试所需要的无错测试运行时间为 $t_2=5685.9-tf_1=3685.9$小时。同样，在验证测试过程中，如果软件在时刻 $tf_1=3652$小时发生失效，则第二次验证测试所需要的无错测试运行时间为 $t_2=5685.9-tf_1=2033.9$小时。显然，第一次软件失效时间的早晚直接影响着第二次验证测试所需要的测试持续期。如果软件在进行第二次验证测试时，顺利通过规定的测试持续期 t_2，那么软件验收合格；而如果在测试过程中的 tf_2时刻($tf_2<t_2$)，软件又发生失效，则需要排除错误并确定需要的第三次无失效验证测试持续期 $t_3=7453.5-tf_1-tf_2$。t_3 的具体取值与前面两次失效发生时间的早晚是息息相关的，这充分体现了对前面两次验证测试信息整合的结果。后面的测试过程以此类推。

根据式(4)，我们可以计算出基于经典统计假设检验的FDT方法对同样的可靠性指标进行验证测试时，每次所需要的软件验证测试持续期都固定为4605.2小时。显然，我们所提供的方法不但能根据测试过程中的具体情况动态调整测试持续期，而且由于考虑了软件的先验信息，所需要的测试持续期更短，这都充分体现了该方法的优越性。对于不同的 a_0 和 b_0，都可以方便地根据式(18)产生相应的表，辅助决定在不同情况下需要的验证测试持续期，使得连续执行软件可靠性验证测试过程变得简单易行。

结论 本文提出的基于经验贝叶斯统计推断的连续执行软件可靠性验证测试方法不但考虑了软件在测试过程中的具体表现情况，而且还考虑了软件可靠性的先验信息，客观地反映了被测试软件可靠性的实际，能有效地减少验证测试所需要的测试持续期，为验证高可靠性指标提供了理论和技术支持。同时，本文提供的软件失效强度先验分布超参数的求解方法又使得先验整合方法在工程中的实施成为可能。在本文工作的基础上，加上测试用例的等价类划分方法和形式化的程序转换方法，使得在当前一些不可验证的超高可靠性指标的验证成为可能。当然，一种理论和技术的成熟，需要在实践中接受检验并加以完善，希望更多软件可靠性验证测试专家学者提出指正。

参考文献

1 Michael L, ed. Handbook of software reliability engineering. McGraw Hill and IEEE Society Press, 1996
2 http://www.openedu.com.cn/file_post/display/read.php?FileID=23060
3 Selding P B. Faulty software caused Ariane 5 failure. Space News, 1996, 7(25): 24～30
4 Leveson N G, Turner C S. An investigation of the Therac-25 accident. IEEE Computer, 1993, 26(7): 18～41
5 何国纬，等著. 软件可靠性. 国防工业出版社，1998
6 徐仁佐. 软件可靠性工程的基本概念、任务及实施方法. 软件世界，1993. 2～3
7 Pham H. Software reliability and testing, IEEE Computer Society Press, 1996
8 Cai K Y, Cai L, Wang W D, et al. On the neural network approach in software reliability modeling. Journal of systems and software, 2001, 58(1): 47～62
9 白成刚，俞蒙槐，胡上序，等. 基于 Bayes 网的软件失效预测模型. 计算机科学，2003，30(6)：162～164
10 毛晓光，邓勇进. 基于构件软件的可靠性通用模型. 软件学报，2004，15(1)：27～32
11 邹丰忠，李伟湘. 软件可靠性混沌模型. 计算机学报，2001，24(3)：281～291
12 董成，毛新军，陈磊，等. 基于 Agent 的软件可靠性评估系统. 计算机科学，2000，17(6)：28～31
13 黎忠文，熊光泽. 软件可靠性评估的误差分析. 系统工程与电子技术，2001，23(6)：87～89
14 US Department of Defense. MIL-HDBK-781A: Reliability test methods, plans and environments. 1996
15 Butler R W, Finelli G B. The infeasibility of quantifying the reliability of life-critical real-time software. IEEE Trans. on Software Engineering, 1993, 19(1): 3～12
16 Tal O. Software dependability demonstration for safety-critical military avionics system by statistical testing. PhD Thesis, Nottingham Trent University, 1999
17 Tal O, MoCollin C, Bendell A. Reliability demonstration for safety-critical systems. IEEE Trans. on Reliability, 2001, 50(2): 194～203
18 覃志东，雷航，熊光泽，等. 一种实时多任务软件可靠性验证方法. 系统工程与电子技术(已录用)
19 覃志东，雷航，桑楠，等. 安全关键软件可靠性验证测试方法研究. 航空学报(已录用)
20 Qin Zhidong, Lei Hang, Sang Nan, et al. Safety-critical software reliability demonstration testing by empirical Bayesian method. International Symposium on Computing and Information, Aug. 2004

(上接第201页)

同：(1) 本文是分别导出系统中每个单一构件的 LTS，通过这些 LTSs 的集成，来构造整个系统的 LTS，后者直接导出整个系统的 LTS 以作为下面的测试准则选择的基础。(2) 本文对测试用例的选择是基于结构化测试覆盖标准，后者采用的是未明确定义的观察函数。(3) 本文测试策略是自底向上的集成测试策略，后者主要采用的是自顶向下的集成测试策略。

参考文献

1 Harrold M J, Liang D, Sinha S. An Approach To Analyzing and Testing Component-Based Systems. In: Proc. of First Intl. ICSE Workshop on Testing Distributed Component-Based Systems. Los Angeles, 1999. 134～140
2 Bertolino A, Corradini F, Inverardi P, Muccini H. Deriving Test plans from architectural descriptions. In: Proc. of the 22nd Intl. Conf. on Software Engineering. ACM Press, 2000. 220～229
3 叶俊民，王振宇，曹瀚，赵恒. 基于 CHAM 模型的 LTS 状态树生成算法. 哈尔滨工程大学学报，2003，24(3)：287～291
4 Inverardi P, Wolf A L. Formal Specifications and Analysis of Software Architectures Using the Chemical Abstract Machine Model. IEEE Transactions on Software Engineering, 1995, 21(4): 373～386
5 Cheung S C, Kramer J. Enhancing Compositional Reachability Analysis with Context Constraints. In: Proc. of the First ACM SIGSOFT Symposium on the Foundations of Software Engineering. D Notkin, 1993. 115～125
6 Koppol P, Carver R H, Tai K C. Incremental Integration Testing of Concurrent Programs. IEEE Transactions on Software Engineering, 2002, 28(6): 607～623
7 Weyuker E J. Testing Component-Based Software: A Cautionary Tale. IEEE Software, 1998, 15(5): 54～59