文章编号:1001-506X(2012)11-2396-05

基于指数环境因子函数的软件可靠性增长模型

高 峰,闫雪丽,袁赣南
(哈尔滨工程大学自动化学院,黑龙江 哈尔滨 150001)

摘 要:绝大部分软件可靠性增长模型采取的软件可靠性标准都是测试可靠性,实际上,软件的测试剖面很难真实地反映运行剖面,软件运行环境和测试环境不可能一致。考虑测试环境与运行环境差别,根据经验数据拟合环境因子,提出一种随时间变化的指数环境因子函数;并建立基于指数环境因子的非齐次泊松过程类软件可靠性增长模型;利用一组公开发表的数据集对模型进行评估。结果表明,本文提出的环境因子函数计算复杂度低,基于该函数的软件可靠性模型具有更好的拟合和预测能力。

关键词:软件可靠性;非齐次泊松过程;测试环境;运行环境;环境因子函数
中图分类号:TP 311 **文献标志码**:A **DOI**:10.3969/j.issn.1001-506X.2012.11.36

Software reliability growth model with exponential environmental-factor function

GAO Feng, YAN Xue-li, YUAN Gan-nan
(*College of Automation, Harbin Engineering University, Harbin 150001, China*)

Abstract: The test and running environment can not be consistent. Considering the difference between test and running environment, a time-varying exponential environment factor function is proposed. The Non-Homogeneous Poisson Process software reliability growth model based on environment factor is established and evaluated using a group of public data. The results show that the proposed function has low computational complexity, the software reliability model based on this function has better fitting results and prediction ability.

Keywords: software reliability; non-homogeneous Poisson process (NHPP); software testing environment; software operational environment; environment-factor function

0 引 言

软件可靠性模型是对软件可靠性进行评估、管理和预测的有效工具,是软件可靠性工程的基础[1]。绝大部分软件可靠性增长模型在建模时都假设软件的测试和运行环境相同,所采取的软件可靠性标准都是测试可靠性,只有在少数文献中提到了运行可靠性标准[2]。实际上,软件的测试剖面很难真实地反映运行剖面,使得软件运行环境和测试环境不可能一致,那么测试阶段和运行阶段的故障检测率也不可能相同。因此,有学者提出环境因子的概念,利用它降低因测试环境与运行环境的差别给软件可靠性估计精度造成的误差[3-4]。这种方法是用一个变量表示环境因子,把软件系统在运行环境下的故障检测率和测试环境下的故障检测率通过环境因子联系起来。考虑环境因子的软件可靠性增长模型对于有高可靠性要求的国防系统软件,特别是嵌入式软件的可靠性评估和预测有非常好的效果。文献[3]将环境因子定义为两种环境下平均故障检测率的比值,该比值是常数。文献[4]提出一种与环境因子有关的软件可靠性模型,假设环境因子是一个随机变量,并假设环境因子不随时间变化,软件系统在运行环境下的故障检测率为测试环境下的故障检测率除以环境因子。文献[5]认为环境因子是随时间变化的,使用 Logistic 曲线拟合环境因子:$k(t)=N/1+A\times e^{-\partial t}$,其中 N,A,∂ 为待定系数,但是,Logistic 曲线并不直观,数学处理非常复杂,并且拟合误差较大。而后,又有学者[6-7]考虑测试过程中每一测试阶段的差别,应用多移动点技术进行可靠性建模,这是一种广义考虑环境差别的概念,但基于多移动点技术的软件可靠性模型参数评估困难且误差较大。

收稿日期:2011-11-21;**修回日期**:2012-05-17。
基金项目:国家自然科学基金(61001154)资助课题
作者简介:高峰(1982-),女,讲师,博士研究生,主要研究方向为软件测试及软件可靠性、组合导航。E-mail:gaofeng19@hrbeu.edu.cn

本文考虑测试环境与运行环境差别，根据经验数据拟合环境因子，提出一种随时间变化的指数环境因子函数，通过环境因子和测试阶段的故障检测率得到运行阶段的故障检测率，并建立基于该环境因子的非齐次泊松过程（non-homogeneous Poisson process，NHPP）类软件可靠性模型，利用一组公开发表的失效数据集对本文模型进行评估。实验结果表明，本文提出的环境因子函数形式简单，计算复杂度低，并且基于该环境因子函数提出的软件可靠性模型具有更好的拟合能力和预测能力。

1　NHPP 类软件可靠性增长模型

本文符号说明：

a：软件的初始故障数；

b：故障检测率的初始值；

$a(t)$：软件故障总数函数，等于 $x(t)$ 与到时刻 t 为止，软件中尚未被检测到的故障数之和；

$b(t)$：与时间相关的故障检测率函数，单位时间内每个故障被发现的概率；

$N(t)$：$t\geqslant 0$，在$[0,t]$时间段内发现的累计软件故障数；

$m(t)$：到 t 时刻，所能发现的故障数的期望值，$m(t)=E[N(t)]$；

$\lambda(t)$：在 t 时刻，单位时间内软件的失效数，称为失效密度函数；

$R(x|t)$：软件可靠度函数，表示从时刻 t 开始到 $t+x$ 时间段内软件的可靠性。

定义 1　若计数过程$\{N(t),t\geqslant 0\}$满足下列条件：

(1) $\{N(t),t\geqslant 0\}$具有不相关增量；

(2) $P\{N(t+h)-N(t)\geqslant 2\}=O(h)$；

(3) $P\{N(t+h)-N(t)\geqslant 1\}=\lambda(t)+O(h)$；

(4) $N(0)=0$。

则称$\{N(t),t\geqslant 0\}$为强度函数为 $\lambda(t)$ 的 NHPP 过程[5]。

一个基于 NHPP 过程的软件可靠性增长模型的一般形式为

$$P\{N(t)=n\}=\frac{m(t)^n}{n!}\mathrm{e}^{-m(t)}$$

失效密度函数为

$$\lambda(t)=\frac{\mathrm{d}m(t)}{\mathrm{d}t}$$

设 t 为最后一次失效发生的时刻，则被测软件在时间段$(t,t+s)$内的可靠度为

$$R(x\mid t)=\exp\left[-m(t+x)-m(t)\right]$$

G-O 模型假设：

(1) 软件失效遵循 NHPP 过程；

(2) 到时刻 t 的累计错误数 $N(t)$ 服从均值函数 $m(t)$ 的泊松过程，任意时间间隔 t 到 $t+\Delta t$ 内期望的错误发生数与 t 时刻剩余的错误数成比例；

(3) 软件的失效强度与软件中尚未发现的故障总数成正比；

(4) 软件运行方式与可靠性预测方式相同；

(5) 每个错误发生的概率相同，各个错误的严重等级相同；

(6) 错误被检测时失效是独立的。

边界条件 $m(0)=0$，$m(\infty)=a$，假设在$(t,t+\Delta t)$中失效出现的个数与 $a-m(t)$，即剩余失效数（在 t 时刻）成正比，即

$$m(t+\Delta t)-m(t)=b\cdot(a-m(t))\cdot\Delta t \tag{1}$$

令 $\Delta t\to 0$，得

$$m'(t)=ab-bm'(t) \tag{2}$$

即

$$m(t)=a(1-\mathrm{e}^{-bt}),\ a>0,\ b>0 \tag{3}$$

许多 NHPP 类软件可靠性增长模型都是根据不同情况，修正以上假设条件，利用前述的思想推导出来的。

2　随时间变化的环境因子

环境因子是降低测试和运行环境的差别的数学工具。定义环境因子[5]为

$$k(t)=\frac{b_{\text{test}}(t)}{b_{\text{field}}(t)} \tag{4}$$

式中，$b_{\text{test}}(t)$和 $b_{\text{field}}(t)$分别代表测试和运行阶段的故障检测率。

由于实测数据是离散的，那么随时间变化的离散的平均的环境因子可以表示为

$$\overline{k}(ti)=\frac{\overline{b}_{\text{test}}(ti)}{\overline{b}_{\text{field}}(ti)} \tag{5}$$

式中，$i=1,2,3,\cdots,n$，$\overline{b}_{\text{test}}(ti)$和$\overline{b}_{\text{field}}(ti)$分别代表测试和运行阶段随时间变化的平均故障检测率。

软件系统在运行阶段的失效率可以通过下式计算求得：

$$\lambda(t)=b_{\text{field}}(t)\times(\hat{a}-m(t)) \tag{6}$$

式中，$\hat{a}$表示用 P-N-Z 模型拟合整个测试阶段和运行阶段的失效数据而得到的估计数；$m(t)$表示到时间 t 累积的期望失效数。

用 $N(t_i)$表示到时间 t_i 的实测的累积失效数，离散的运行阶段的平均失效率可以表示为

$$\overline{\lambda}(t_i)=\frac{N(t_i)}{t_i} \tag{7}$$

那么随时间变化的平均故障检测率可表示为

$$\overline{b}_{\text{field}}(t_i)=\frac{N(t_i)}{t_i\times(\hat{a}-N(t_i))} \tag{8}$$

3 基于指数环境因子函数的考虑测试与运行差别的可靠性模型

3.1 假设条件

(1) 无论是测试阶段或者运行阶段,软件失效都分别遵循一个 NHPP 过程;

(2) 到时刻 t 的累计错误数 $N(t)$ 服从均值函数 $m(t)$ 的泊松过程,任意时间间隔 t 到 $t+\Delta t$ 内期望的错误发生数与 t 时刻剩余的错误数成比例;

(3) 考虑软件测试环境与运行环境的差别,软件运行阶段的故障检测率是通过测试阶段故障检测率和随时间变化的环境因子函数转化而得到的;

(4) 软件中每个错误是相互独立的,每个错误导致系统发生失效的可能性也相同。

3.2 模型的评价和拟合标准

分析中使用误差平方和 SSE[5] 与回归曲线方程相关指数 *R-square*[8] 度量曲线拟合效果。

SSE 用来描述累计错误数的实际值与预测值之间的距离,其定义为

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

式中,n 表示失效数据集中失效样本的数量;y_i 表示到 t_i 时刻观测到的错误数;$\hat{m}(t_i)$ 表示在 t_i 时刻,模型估算得到的错误数。SSE 的值越小,说明曲线拟合效果越好。

R-square 定义为

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

式中,$\bar{y}$ 表示观测到错误的平均值,$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$。*R-square* 的值越接近于 1,表示曲线拟合得越好。

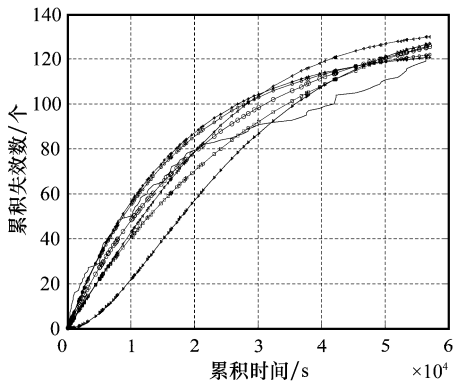
3.3 选择测试阶段最优拟合模型

在文献[9]中公开发表了一组数据:136 个故障(Fault)及其出现的累积时间。从第 122 个失效到第 123 个失效之间的时间间隔很明显;从第 123 个失效以后,失效时间间隔很大,远远超过在第 122 个失效以前的失效时间间隔,这预示着软件系统的可靠性已经增长,系统变得稳定。所以,在软件可靠性建模方法研究中,选择前 122 个失效数据作为测试阶段的数据,用于估计模型参数,计算拟合误差,并选择测试阶段的最优拟合模型;余下的数据作为运行阶段的数据,用来评估模型的预测能力。

在软件测试阶段,在几个经典的 NHPP 类软件可靠性增长模型中选择最优拟合模型,模型名称、模型表达式、估计的参数值,SSE 及 *R-square* 的计算结果如表 1 所示,测试阶段各个模型的拟合曲线如图 1 所示。

表 1 各个模型的参数估计值及拟合能力

Model Name	Parameters	*SSE*	*R-square*
G-O Model[10]	$\hat{a}=125$ $\hat{b}=0.000\ 06$	7 614.3	0.963 9
Delayed S-shaped[7]	$\hat{a}=140$ $\hat{b}=0.000\ 07$	51 672	0.842 3
Inflection S-shaped SRGM[11]	$\hat{a}=135.5$ $\hat{b}=0.000\ 07$ $\hat{\beta}=1.2$	16 192	0.938 7
Yamada Exponential[12]	$\hat{a}=130$ $\hat{\alpha}=10.5$ $\hat{\beta}=5.4\times10^{-6}$	6 571.2	0.968 6
P-N-Z Model[13]	$\hat{a}=121$ $\hat{b}=0.000\ 05$ $\hat{\alpha}=2.5\times10^{-6}$ $\hat{\beta}=0.002$	6 203.1	0.971 2
Z-T-P Model[7]	$\hat{a}=135$ $\hat{b}=0.001$ $\hat{\alpha}=0.01$ $\hat{\beta}=0.012$ $\hat{c}=3.5\times10^{-5}$	11 297	0.949 8



—: The Real Data; →: G-O Model; →: Delayed S-shaped;
→: Inflection S-shaped; →: Yamada Exponetial;
→: P-N-Z Model; →: Z-T-P Model。

图 1 测试阶段各个模型的拟合曲线

图 1 仿真结果显示,P-N-Z 模型在测试阶段,尤其是在前 20 000 s,拟合效果非常好;虽然在 350 000 s 以后所有模型的期望累积失效数都大于实际值,出现了悲观的估计,但从表 1 可以看出整个测试阶段 P-N-Z 模型的 SSE 最小,*R-square* 最接近 1,说明 P-N-Z 模型的误差最小、拟合能力最好。所以在测试阶段的拟合模型选择 P-N-Z 模型。

3.4 定义环境因子函数

为了定义环境因子函数,做如下讨论:

分析 1 当故障检测率降低到一定程度时候,软件具有一定的可靠性才会发布,所以测试阶段结束时故障检测率 $b_{\text{test}}(t)$ 是递减函数或趋于一个常数;运行阶段的故障检测率由式(8)决定,它与使用环境、操作人员习惯、软件系统运行时间长度、功能模块使用频率等因素有关,$b_{\text{field}}(t)$ 变化是不确定的,可能是增函数也可能是减函数或者是常数。

不管是什么情况，环境因子都应该是随时间变化的函数，其趋势是随时间递增的或者递减的。

分析 2　在已存在的拟合效果比较好的模型中，有些模型的预测结果比较乐观，比如 G-O 模型，估计的累积失效数比实际累积失效数少，使计算的 $b_{\text{field}}(t_i)$ 为负数；有些模型的预测结果比较悲观，如 LV 模型，使计算的 $b_{\text{field}}(t_i)$ 为正数。这两种情况使得环境因子 $k(t)$ 值有可能为正值有可能为负值。

根据以上分析，定义环境因子的理论曲线表达式为

$$k_{\text{new}}(t) = A \times \mathrm{e}^{-Bt} + C \tag{9}$$

其中，A，B，C 为待定系数，其代表意义如下：

(1) 用 e^{-Bt} 拟合环境因子的递增或递减趋势：当参数 $B>0$ 时，表示环境因子随时间递减；当 $B<0$ 时，表示环境因子随时间递增，参数 B 的大小表示环境因子变化的速率；

(2) 用 A 表示环境因子变化的比例；

(3) 根据分析(2)，添加 C 为调整系数。当 $A>0$ 时，$A\times\mathrm{e}^{-Bt}>0$；当 $A<0$ 时，$A\times\mathrm{e}^{-Bt}<0$，调整系数 C，可以实现使离散的环境因子值有正数也有负数。

由失效数据得到 $b_{\text{test}}(ti)$，$b_{\text{field}}(ti)$ 值，估算环境因子的参数：

(1) 当 $t=0$，软件系统进入运行阶段，$k_{\text{new}}(0)=A+C$；

(2) 当 $t\to t_{\text{final}}$ 时，到达运行的最后阶段，$k_{\text{new}}(t_{\text{final}})=A\times\mathrm{e}^{-Bt_{\text{final}}}+C$。

根据上面两个条件，设置 A，B，C 参数的初始值，用最小二乘法计算 A，B，C。将其代入式(9)，得到新的环境因子的函数形式为

$$k_{\text{new}}(t) = A \times \mathrm{e}^{-Bt} + C = 3.7 \times \mathrm{e}^{-0.000\,036t} + 0.01 \tag{10}$$

即系数 $A=3.7$，$B=0.000\,036$，$C=0.01$；拟合误差为 $SSE=0.009\,8$。

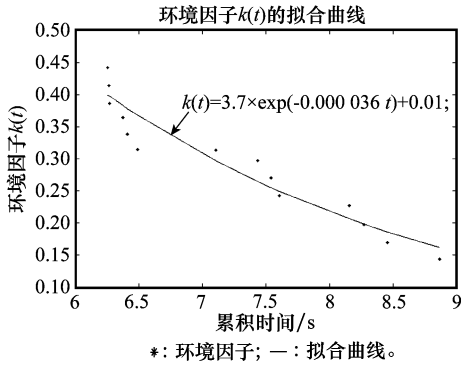


图 2　指数环境因子函数拟合曲线

3.5　运行阶段的软件可靠性模型

测试阶段，P-N-Z 模型的故障检测率 $b(t)$ 为

$$b(t) = \frac{b}{1+\beta \mathrm{e}^{-bt}} = \frac{0.000\,05}{1+0.002\times \mathrm{e}^{-0.000\,05t}} \tag{11}$$

在测试阶段，当故障检测率降低到一定程度时候，软件具有一定的可靠性才会发布。随着测试时间的增长，t 逐渐增大，$b_{\text{test}}(t)\to 0.000\,05$，取 $b_{\text{test}}(t)=0.000\,05$。

运行阶段的故障检测率为

$$b_{\text{field}}(t) = \frac{b_{\text{test}}(t)}{A\times \mathrm{e}^{-Bt}+C} = \frac{0.000\,05}{3.7\times \mathrm{e}^{-0.000\,036t}+0.01} \tag{12}$$

运行阶段的故障 $a_{\text{field}}(t)$ 为

$$a_{\text{field}}(t) = (\hat{a} - m(T))(1+\alpha t) \tag{13}$$

式中，T 为软件发布的时间；$m(T)$ 为软件发布时期望的累积失效数。

那么运行阶段的软件可靠性增长模型为

$$m_{\text{field}}(t) = a_{\text{field}}(1-\mathrm{e}^{-B^*(t)}) \tag{14}$$

式中，$B^*(t)=\int_0^T b_{\text{field}}(t)\mathrm{d}t$。

将式(12)和式(13)代入式(14)可得到指数环境因子函数形式的软件可靠性模型：

$$m_{\text{field}}(t) = (\hat{a} - m(T))(1+\alpha t)\left(1-\left(\frac{A+C\mathrm{e}^{Bt}}{A+C}\right)^{-\frac{b}{BC}}\right) \tag{15}$$

若令 $M=(\hat{a}-m(T))$，$X=\dfrac{A}{A+C}$，$Z=\dfrac{b}{B+C}$，那么式(15)可以转化为

$$m_{\text{new}}(t) = M(1+\alpha t)(1-(X+XC\mathrm{e}^{Bt})^{-Z}),\ t\geqslant T \tag{16}$$

因此，从测试到运行阶段本文模型的均值函数如下：

$$m(t) = \begin{cases} \dfrac{a[1-\mathrm{e}^{-bt}]\times[1-(\alpha/b)]+a\alpha t}{1+\beta \mathrm{e}^{-bt}}, & 0\leqslant t<T \\ (\hat{a}-m(T))(1+\alpha t)\left(1-\left(\dfrac{A+C\mathrm{e}^{Bt}}{A+C}\right)^{-\frac{b}{BC}}\right), & t\geqslant T \end{cases} \tag{17}$$

4　模型的评估比较

使用软件系统的运行阶段失效数据检验模型的拟合能力和预测能力。为描述方便，用 Proposed Model 表示本文模型，L-F Model 表示文献[6]中的模型，这两个模型的测试阶段拟合模型都是 P-N-Z 模型。

各个模型对运行阶段的拟合曲线见图 3，预测能力评估曲线见图 4。在运行阶段各个模型拟合能力和预测能力数据见表 2。

表 2　各个模型预测能力比较

Model Name	*SSE*	*R-square*
G-O Model	705.693 4	0.448 3
Delayed S-shaped	314.746 8	0.553 8
Inflection S-shaped SRGM	321.281 6	0.408 1
Yamada Exponential	333.337 6	0.445 1
P-N-Z Model	236.385 4	0.658 0
Z-T-P Model	229 200	0.003 3
L-F Model	137.171 3	0.605 2
本文模型(Proposed Model)	57.330 1	0.863 5

由图 3 仿真结果，本文模型在整个运行阶段的预测能力显而易见，特别是在第 80 000 和 85 000 之间模型几乎与

实测数据重合；相对来说，其他模型只能预测实测累积失效数的大体趋势；观察 L-F Model，虽然其预测结果比不使用环境因子的模型效果好很多，但与实测数据相距很大，相对本文模型，还是有些逊色。图 4 预测能力评估曲线显示，本文模型的 RE 曲线几乎在 0.02 以内，并始终是最接近于 0 的，在震荡中逐渐趋近于 0；相对来说，其他模型的 RE 曲线变化很大，没有收敛的趋势，如 G-O Model，Yamada Exponential Model，或者一直比较大，如 Delayed S-Shaped Model；观察 L-F Model，其 RE 曲线几乎都在 0.02 之外，并没有收敛到 0 的趋势。表 2 中的数据显示，在整个运行阶段，本文模型的 $SSE=60.951\ 5$，$R\text{-}square=0.846\ 5$，是所列模型中预测误差最小，预测趋势最好的模型。

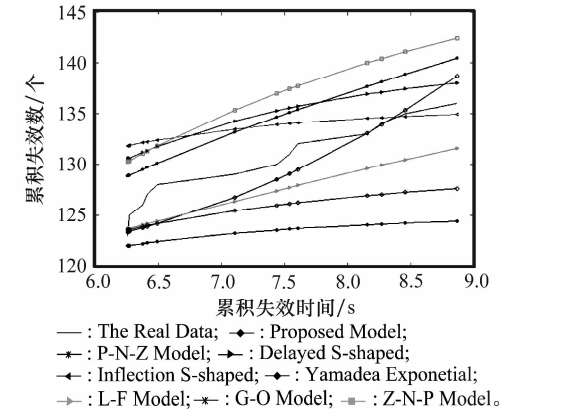


图 3 运行阶段各个模型的拟合曲线

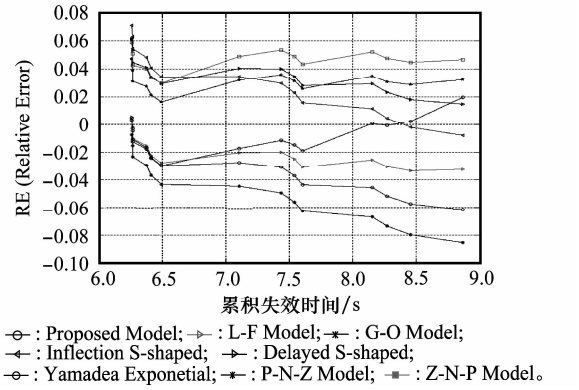


图 4 运行阶段各个模型的预测能力评估曲线

5 结 论

软件的测试剖面并不等于运行剖面，软件运行环境和测试环境不可能一致。考虑测试环境与运行环境差别，提出一种随时间变化的指数环境因子函数，通过环境因子函数和测试阶段的故障检测率转化得到运行阶段的故障检测率，并建立了运行阶段软件可靠性增长模型。通过一组公开发表的数据对本文模型进行评估，实验结果表明，本文提出的指数环境因子函数，表示形式简单，计算复杂度低；并且基于该环境因子函数提出的软件可靠性模型与其他模型相比具有较好的拟合能力和预测能力。

参考文献：

[1] Musa J D. *Software reliability engineering*[M]. New York: McGraw Hill, 1999: 2 - 3.

[2] 赵靖，刘宏伟，崔刚，等. 考虑测试环境与运行差别的软件可靠性增长模型[J]. 计算机研究与发展，2006，43(5)：881 - 887.(Zhao J，Liu H W，Cui G，et al. A software reliability growth model considering testing environment and actual operation environment[J]. *Journal of Computer Research and Development*，2006，43(5)：881 - 887.)

[3] 楼俊刚，江建慧，帅春燕，等. 软件可靠性模型研究进展[J]. 计算机科学，2010，37(9)：13 - 19.(Lou J G，Jiang J H，Shuai C Y，et al. Software reliability models: a survey[J]. *Computer Science*，2010，37(9)：13 - 19.)

[4] Teng X，Pham H. A software cost model for quantifying the gain with considerations of random field environments[J]. *IEEE Trans. on Computers*，2004，53(3)：380 - 384.

[5] Zhao J，Liu H W，Cui G，et al. Software reliability growth model with change-point and environmental function[J]. *The Journal of Software and Systems*，2006，79(11)：1578 - 1587.

[6] Huang C Y，Hang Tsui-Ying. Software reliability analysis and assessment using queueing models with multiple change-points[J]. *Computer and Mathematics with Application*，2010，60(7)：2015 - 2030.

[7] Brendan M. The difficulties of building generic reliability models for software[J]. *Empirical Software Engineering*，2012，17(1 - 2)：18 - 22.

[8] Huang C Y，Huang W C. Software reliability analysis and measurement using finite and infinite server queuing models[J]. *IEEE Trans. on Realiabilty*，2008，57(1)：192 - 203.

[9] Zhang X，Teng X L，Pham H. Considering fault removal efficiency in software reliability assessment[J]. *IEEE Trans. on Systems，Man，and Cybernetics-Part A: Systems and Humans*，2003，33(1)：114 - 119.

[10] Dragan J. The L-p-norm estimation of the parameters for the Jelinski-Moranda model in software reliability[J]. *International Journal of Computer Mathematics*，2012，89(4)：467 - 481.

[11] Ohba M，Yamada S. S-Shaped software reliability growth models[J]. *Proc. of the 4th International Conference on Reliability and Maintainability*，1984：430 - 436.

[12] Yuan Y，Zhu H Y，Liu B. Software reliability modeling with removed errors and compounded-decreased-rate[J]. *Mathematical and Computer Modeling*，2012，55(3 - 4)：697 - 709.

[13] Pham H，NordMann L，Zhang X. A general imperfect software debugging model with S-Shaped fault detection rate[J]. *IEEE Trans. on Reliability*，1999，48(2)：169 - 175.