

一种基于离散时间的 NHPP 软件可靠性增长模型

何 焱, 张来顺, 刘 伟, 黎中文

(解放军信息工程大学 电子技术学院, 郑州 450004)

摘 要: 基于非齐次泊松过程的软件可靠性增长模型按时间域可分为连续时间模型和离散时间模型两类。现有的软件可靠性增长模型大多都是针对连续时间构造的,在一定程度上忽视了对离散时间模型的研究。利用概率生成函数构建两种基于离散时间的软件可靠性增长模型——基本模型和扩展模型,具有很大的实用性和必要性。构建的扩展模型以不完美排错情形作为基础,考虑到了由于故障排除而有可能引入新故障的问题,同时还考虑到了在软件排错过程中由于测试团队的熟练程度而引起的软件故障排除率的相对变化情况,这使得提出的模型更加符合实际。最后利用两组公开发表的失效数据集对两种模型与指数模型和 S 型模型进行对比验证,验证结果表明提出的两种模型具有更好的拟合能力和预测效果。

关键词: 软件可靠性增长模型; 非齐次泊松过程(NHPP); 离散时间模型; 参数估计; 不完美调试

中图分类号: TP311 **文献标志码:** A **文章编号:** 1001-3695(2011)07-2569-04

doi:10.3969/j.issn.1001-3695.2011.07.047

Discrete time NHPP based software reliability growth model

HE Yan, ZHANG Lai-shun, LIU Wei, LI Zhong-wen

(Institute of Electronic Technology, PLA Information Engineering University, Zhengzhou 450004, China)

Abstract: Non-homogeneous Poisson process based software reliability growth models are generally classified into two groups by time: continuous time models and discrete time models. Existing software reliability growth models are mostly developed in the continuous time models while there are fewer in the discrete time models. It is necessary to research discrete time SRGM. This paper proposed two discrete time SRGMs using probability generating function based on a NHPP namely, basic model and extended model. In the proposed extended model, if considered imperfect debugging and new fault derived and changed with fault debugging, which was more realistic. The predictive power and goodness-of-fit of the proposed models were examined by two public failure data sets. Compared with other existing models: exponential model and S-shape model, the results demonstrate that the proposed model has better prediction and goodness-of-fit.

Key words: software reliability growth model (SRGM); NHPP (non-homogeneous Poisson process); discrete time models; parameter estimation; imperfect debugging

0 引言

随着软件的结构和规模越来越庞大,人们对软件的安全性和可信度的要求也越来越高。软件可靠性作为可信软件的一个重要指标也因此而得到国内外学者的广泛关注和研究。近 20 年来,针对测试阶段失效数据收集的软件可靠性增长模型在软件工程领域内展开了广泛而深入的研究^[1~9]。实际上软件可靠性增长模型的构建就是寻找一种数学方法来刻画在测试/使用阶段软件累计故障检测数和时间的关系^[1~4]。基于模型的数学描述,国内外学者也提出了许多软件可靠性增长模型,在这些模型中基于非齐次泊松过程(NHPP)类的软件可靠性增长模型以其数学易处理性和广泛适用性而备受欢迎,因此基于 NHPP 类的软件可靠性模型研究也非常广泛。

基于 NHPP 类的软件可靠性增长模型按时间域可分为两类:a)以执行时间(CPU 时间)或日历时间作时间参数的连续时间模型;b)以测试用例或测试场合作时间参数的离散时间

模型。目前,由于数学处理过程较复杂,大量的模型都是基于连续时间的可靠性增长模型,而基于离散时间提出的模型较少。但在实际软件评估工作中,基于离散时间的软件可靠性增长模型的模拟效果并不比连续增长模型差,甚至在很多方面表现得比连续增长模型更优越^[1,3,10]。例如文献[1,10]提出由于离散时间的记录更加容易和准确,离散时间模型比连续时间模型更符合实际情况,评估和预测效果更佳。

在文献[3]中,Yamada 等人于 1985 年提出了两种模型,一种模型是在故障检测过程中每一测试用例的故障期望数随时间呈几何级递减形状;另一模型是期望数和剩余故障数成比例关系的指数模型。但这两种模型都将排错过程认为是完美的,既没考虑到排错过程较检错过程的延迟时间,也没考虑到排错过程中引入错误等。在文献[4]中,Kapur 等人于 1999 年引入不完美排错模型,认为剩余故障的故障排除率因不完美测试而不断减小,可靠性增长曲线是指数型。他们虽然考虑到了不完美调试过程,但认为排错过程和测试时间是独立的,并没有考虑到测试人员随着测试进行,测试经验增加而提高排错效率的

收稿日期:2010-12-09;修回日期:2010-01-21

作者简介:何焱(1984-),男,四川省泸州人,硕士研究生,主要研究方向为软件可靠性评估(heyani16512@yahoo.com.cn);张来顺(1964-),男,教授,硕导,主要研究方向为可信软件;刘伟(1965-),男,副教授,硕导,主要研究方向为软件体系结构;黎中文(1985-),男,硕士,主要研究方向为软件工程。

过程。文献[1]指出在实际的软件排错过程中由于各测试人员等因素会引入新错误,实际上不完美排错率应该是在测试初期最大而随着测试的进行而减小的过程。另外,由于测试人员随着测试熟练程度提高,故障引入率随时间越来越小。

本文提出了两种基于离散时间 NHPP 软件可靠性增长模型:基本离散时间模型和扩展离散时间模型。扩展模型考虑到了不完美排错过程中引入新错误和测试人员随着测试的进行熟练程度增加的情形。基本离散时间模型是扩展离散时间模型的基础,扩展离散时间模型是在基本离散时间模型上考虑实际情况合理扩展的结果。所以说扩展模型的提出是深入研究基本模型的结果,二者密不可分,不能只为了得到更好的模拟结果而忽视了对基本模型的构建和研究,本文并没有只求得出优异的模型而省略了构建基本模型部分章节。文章最后利用两组公开发表的失效数据集对两种提出的模型与指数模型和 S 型模型进行对比验证,验证结果表明本文提出的两种模型具有更好的拟合能力和预测效果。

1 构建离散时间模型

1.1 构建基本离散时间模型

符号声明:

a 表示软件最初所包含的故障数;

$b(n+1)$ 表示故障检测率函数,与测试用例数有关;

b_i 表示初时刻故障检测率;

b_f 表示末时刻故障检测率;

$m(n)$ 表示第 n 次故障检测的平均预期故障数量。

基本离散时间模型包含以下三个假设条件:

a) 错误发现/故障检测服从非齐次泊松过程。

b) 故障每次出现,立即就被发现并找出故障原因,立即排错。

c) 排错过程不会引入新故障。

基于以上假设条件,在第 n 次测试后,在 $n \sim n+1$ 个测试用例使用间累计故障预期检测数与剩余故障数成比例,可得方程:

$$m(n+1) - m(n) = b(n+1)(a - m(n)) \quad (1)$$

由故障检测率和故障检测数间的关系可知:

$$b(n+1) = b_i + (b_f - b_i) \frac{m(n+1)}{a} \quad (2)$$

在式(2)中由于 b_i 和 b_f 值的不同,可以得出以下四种情况:

当 $b_i = b_f = b$ 时,故障检测率为常量;

当 $b_f > b_i$ 时,故障检测率逐渐增大;

当 $b_f < b_i$ 时,故障检测率逐渐减小;

当 $b_i > 0, b_f = 0$ 时,故障检测率逐渐减小,且末时刻故障检测率为 0。

将式(1)代入式(2)中,设 $m(n=0) = 0$ 可得概率生成函数:

$$m(n) = \frac{a[1 - (1 - b_f)^n]}{1 + \frac{b_f - b_i}{b_i}(1 - b_f)^n} \quad (3)$$

在式(3)中,参数 b_f 和 b_i 的值决定了可靠性增长曲线的形状,这其中包括指数型或 S 型曲线。当 $b_i = b_f = b$ 时,式(3)可得:

$$m(n) = a[1 - (1 - b)^n] \quad (4)$$

由以上方程可知,基本离散时间模型可以分为以下四种情况:检测率为常量、严格递增、严格递减、时而增时而减。而基于离散时间的指数模型^[6,8]和 S 型模型^[3~5]均不能实现时而增时而减的情形。此模型对基于离散时间的增长模型来说具有重要意义。

从假设不难看出,此基本模型考虑还不够严密,还不能反映出软件测评过程中一些实际情形。因此,对基本模型进行扩展是很有必要的。

1.2 构建扩展离散时间模型

首先对式(1)中的参数 a 进行扩展。式(1)中 a 不能表示排错过程中引入错误的情形,扩展后用 $a(n)$ 表示软件中的错误数是可变的函数,它的值取决于团队测试水平、测试用例数量、软件大小及复杂性等条件。扩展模型中设计的错误数量包含最初错误数量和引入错误数量。

在原基本离散时间模型基础上增加以下三条符号声明:

$a(n)$ 表示总故障数,包含最初错误数量和引入错误数量;

k 表示每一测试用例所检测故障的故障检测率;

P 表示故障的移除概率。

同时,在原模型假设条件的基础上增加以下条件:

a) 软件所包含的故障内容和测试用例数紧密相关。

b) 排错过程中可能引入新错误。

c) 排错过程并不完美。

d) 随着测试人员的不断熟练,故障检测率逐渐提高。

基于以上基本和扩展的假设条件,在第 n 次测试后,在 $nn+1$ 个测试用例间预期累计故障检测数与剩余故障数成比例,可得方程式(5):

$$m(n+1) - m(n) = b(n+1)(a(n) - m(n)) \quad (5)$$

式(5)中, $a(n)$ 和 $b(n+1)$ 均与测试用例函数有关。如果 $a(n)$ 增大,表示排错过程有错误引入; $b(n+1)$ 是一条 S 型曲线,它能反映出测试中测试人员的学习过程。不仅如此,它还受故障移除概率的影响,二者的方程如下:

$$a(n) = a(1 + kn) \quad (6)$$

$$b(n+1) = \frac{b_f p}{1 + \frac{b_f - b_i}{b_i}(1 - b_f p)^{n+1}} \quad (7)$$

设 $m(n=0) = 0$,将式(6)(7)代入式(5)中,可得扩展模型的预期故障均值函数:

$$m(n) = \frac{a}{1 + \frac{b_f - b_i}{b_i}(1 - b_f p)^n} [(1 - (1 - b_f p)^n) (\frac{b_f p - k}{b_f p}) + kn] \quad (8)$$

由式(8)可知,当故障检测率为常数,即 $b_i = b_f = b$ 时,排错过程中无错误引入,即 $k=0$;如果再考虑完美排错,即 $p=1$,此时式(8)就缩减变形为式(4)。当只考虑排错过程中无错误引入($k=0$)和完美排错($p=1$)时式(8)就缩减变形为式(3)。从这个角度也可以看出,式(8)考虑了更多的影响因素,更全面、更符合实际。

2 模型的参数估计和评价标准

2.1 参数估计

得出扩展模型后,便可对模型中未知的参数进行估计。常

见的参数估计方法有最小二乘法、极大似然估计 (MLE) 法等,这两种方法都可以用来对开发模型的未知参数进行估计。本文采用极大似然估计对模型式(8)中的参数进行估计。

由故障均值函数可得似然函数如下:

$$L(\text{parameters} | (n_i, x_i)) = \prod_{i=1}^f \frac{[m(n_i) - m(n_{i-1})]^{x_i - x_{i-1}}}{(x_i - x_{i-1})!} \times \exp(-(m(n_i) - m(n_{i-1}))) \quad (9)$$

其中: (n_i, x_i) 成对出现, x_i 表示第 n_i 个测试用例的累计故障的数量; n_i 表示检测到 x_i 个故障数量时的测试用例数量。

对式(9)取自然对数可得:

$$\ln L = \sum_{i=1}^f (x_i - x_{i-1}) \ln [m(n_i) - m(n_{i-1})] - \{m(n_i) - m(n_{i-1})\} - \sum_{i=1}^f \ln [(x_i - x_{i-1})!] \quad (10)$$

对式(10)微分便可求出各参数估计值。同时需指出在式(8)中,各参数的约束范围: $a, b_i, b_f > 0, 0 < p \leq 1, k \geq 0$ 。

2.2 模型评价准则

软件可靠性增长模型评价准则可以分别从以下两方面进行评价。

1) 模型的拟合度

模型拟合度评价的标准主要包括误差平方和 (sum of squared error, SSE)、赤池信息量准则 (Akaike information criterion, AIC) 和回归曲线方程相关指数 (R-square) 三个评价标准。

SSE 用来描述预测错误数与实际错误数之间的差别,其公式定义为

$$SSE = \sum_{i=1}^f (\hat{m}(n_i) - x_i)^2 \quad (11)$$

其中: x_i 表示观测错误数; f 表示数据集中失效的数量; $\hat{m}(n_i)$ 表示估算的错误数。SSE 值越小,说明观测值与估计值越接近,模型的拟合度越好。

AIC 定义为

$$AIC = -2\ln(L) + X \quad (12)$$

其中: L 是似然函数; X 是模型中参数的数量。AIC 值越小,说明模型的拟合度越好。

R-square 定义为

$$R\text{-square} = \frac{\sum_{i=1}^f (\hat{m}(n_i) - x_i)^2}{\sum_{i=1}^f (x_i - \bar{x})^2} \quad (13)$$

其中, $\bar{x}$ 表示检测到错误的均值, $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ 。R-square 的值越接近 1,模型的拟合度越好。

2) 模型的预测效果

模型的预测效果是用现在或过去的行为去预测未来行为的能力。预测效果可以用相关故障预测效果 (relative prediction fault, RPF) 来衡量。

RPF 定义为

$$RPF = \frac{\hat{m}(n_f) - x_f}{x_f} \quad (14)$$

其中: x_f 表示在最后一轮测试后被移除的故障数; $\hat{m}(n_f)$ 表示模型估计的故障数。如果 RPF 值为正/负数,那么故障移除的估计值就过高/低;如果值越趋近于 0,那么模型预测结果就越精确。

3 模型的对比验证与评价

3.1 对比模型的选择

为了对扩展的软件可靠性增长模型性能进行验证,本文选择了没有考虑不完美排错和故障引入的两个经典模型与本文提出的两种模型进行对比。

模型 I Goel 和 Okumoto 提出的 G-O 模型。G-O 模型假设错误检测到后被立即排除,且排错过程不引入新的错误。其均值函数为

$$m(t) = a(1 - \exp(-bt)) \quad a > 0, b > 0 \quad (15)$$

其中: a 为软件初始故障总数; b 为故障检测率。该均值函数即为经典的指数模型。

模型 II Yamada 和 Osaki 提出的 S 型曲线模型。此模型假设排错过程不引入新的错误。其均值函数为

$$m(t) = a(1 - (1 + bt) \exp(-bt)) \quad a > 0, b > 0 \quad (16)$$

3.2 模型的对比验证与分析

为了进行对比验证,本文选取了两组公开发表的数据集进行分析,通过这两组数据集分别对各模型进行参数评估和模型验证,最后对对比结果进行分析。

第一组数据 (DS-I) 选自于贝尔实验室,经过 20 周的测试,检测出 100 个错误;第二组数据 (DS-II) 选自于美国雷达系统,经过 35 个月测试,检测出 1 301 个错误^[1,3]。

应用极大似然估计的方法,分别在 DS-I、DS-II 中对基本模型和扩展模型进行参数评估。由模型 I 即式(3)经极大似然估计后参数 a, b_i, b_f 的评估结果如表 1 所示。

表 1 基本模型在数据集中的参数估计

模型名称	数据集	a	b_i	b_f
基本模型	DS-I	108	0.071 7	0.158 1
	DS-II	1 435	0.081 7	0.134 2

由模型 II 即式(8)经极大似然估计后参数 a, b_i, b_f, p, k 的评估结果如表 2 所示。

表 2 扩展模型在数据集中的参数估计

模型名称	数据集	a	b_i	b_f	p	k
扩展模型	DS-I	104	0.068 5	0.139 6	0.993 5	0
	DS-II	1 356	0.079 4	0.126 8	0.991 8	0.002 8

由表 1、2 可以看出 $b_f > b_i$,这是因测试团队技能提高、故障排除效率改善而引起的; $p < 1$ 说明扩展模型考虑到了排错过程并不完美,可能引入新错误,或并没能把错误排除掉。

1) 模型的拟合度评估

根据得出的参数估计值,可以代入基本和扩展模型中计算出预期故障数,从而可以分别计算出模型拟合度评价标准 SSE、AIC、R-square,具体结果见表 3、4。

表 3 DS-I 中各模型的拟合度

模型名称	a	b	SSE	AIC	R-square
模型 I	119	0.079 8	232	92	0.948 2
模型 II	116	0.079 0	216	86	0.952 4
基本模型	见表 1		180	78	0.974 8
扩展模型	见表 2		172	73	0.986 8

由表 3、4 不难看出,基本模型和扩展模型的 SSE 和 AIC 都比模型 I 和模型 II 小,基本模型和扩展模型的 R-square 都比模型 I 和模型 II 大,由此可说明基本模型和扩展模型的拟合度比模型 I 和模型 II 好。

表 4 DS-II 中各模型的拟合度

模型名称	<i>a</i>	<i>b</i>	SSE	AIC	<i>R</i> -square
模型 I	1 560	0.082 2	7 415	520	0.966 9
模型 II	1 515	0.081 4	7 460	518	0.969 7
基本模型	见表 1		6 924	428	0.985 6
扩展模型	见表 2		6 482	316	0.997 6

由此,可以得出四个模型分别在 DS-I 和 DS-II 拟合度曲线图,如图 1 和 2 所示。

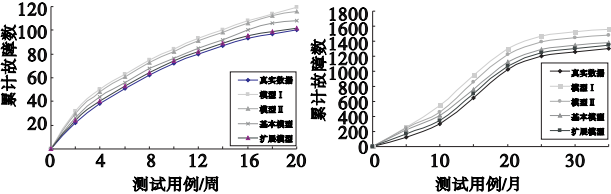


图1 各模型在DS-I中拟合度曲线

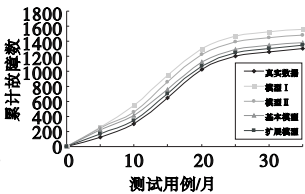


图2 各模型在DS-II中拟合度曲线

由图 1 和图 2 可知,各模型在 DS-I 中的拟合曲线呈现指数类型,在 DS-II 中表现为 S 型曲线。在各模型中扩展模型的拟合度曲线与实际曲线最接近,也可说明扩展模型最能反映真实数据拟合曲线。

2)模型的预测效果评估

本文把 DS-I 和 DS-II 中的数据截断按比例分成不同的部分,针对每一部分计算出 RPF 值,绘制出预测效果图,从而可以进行模型的预测效果评估。本文将数据截断分成六部分,每部分比例和 RPF 值如图 3、4 所示。

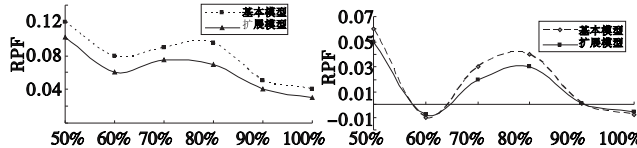


图3 在DS-I中模型的预测效果

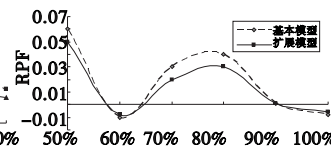


图4 在DS-II中模型的预测效果

由图 3 可知,在基本模型和扩展模型中都高估了模型中的故障数;由图 4 可知,在 58% ~ 64% 时和 90% ~ 100% 时两模型均低估了模型中的故障数。同时还可看出,扩展模型的预测效果比基本模型的预测效果优异,达到了预期构建模型的目标。

4 结束语

本文构建了一种基于离散时间的软件可靠性增长模型和一种扩展的离散时间增长模型。在扩展的模型中考虑了不完美排错和排错过程中引入错误的情况;同时考虑到随着测试的进行,测试人员测试熟练程度的提高错误检测率会相应提高。经两组公开的数据集测试验证,提出的两种模型效果突出;同时验证了扩展模型无论在拟合度和预测效果上相对于基本模型都有明显的提高。在接下来的工作中,可以对离散模型测试用例自动生成相关问题进行研究。

参考文献:

[1] CAI Kai-yuan, CAO Ping, DONG Zhao, *et al.* Mathematical modeling of software reliability testing with imperfect debugging[J]. *Computers & Mathematics with Applications*, 2010, 59 (10): 3245-3285.

[2] 谢景燕,安全霞,朱纪洪.考虑不完美排错情况的 NHPP 类软件可靠性增长模型[J]. *软件学报*, 2010, 21(5): 942-949.

[3] YAMADA S, OSAKI S. Discrete software reliability growth models[J]. *Applied Stochastic Models and Data Analysis*, 1985, 1(1): 65-77.

[4] KAPUR K, SINGH O, SHATNAWI O. A discrete non-homogeneous poisson process model for software reliability growth with imperfect debugging and fault generation[J]. *International Journal of Performance Engineering*, 2006, 2(4): 351-368.

[5] GOEL L, OKUMOTO K. Time dependent error detection rate model for software reliability and other performance measure [J]. *IEEE Trans on Reliability*, 1979, 28(3): 206-211.

[6] 侯春燕,崔刚,刘宏伟.等.基于构件的 NHPP 类软件可靠性增长模型的研究[J]. *计算机科学*, 2009, 36(4): 195-199.

[7] PHAM H, ZHANG X M. NHPP software reliability and cost models with testing coverage [J]. *European Journal of Operational Research*, 2003, 145(3): 443-454.

[8] WU Y P, HU Q P, XIE M, *et al.* Modeling and analysis of software fault detection and correction process by considering time dependency [J]. *IEEE Trans on Reliability*, 2007, 56(4): 629-642.

[9] LO J H. Effect of the delay time in fixing a fault on software error models[C]//*Proc of the 31st Annual International Computer Software and Applications Conference*. Washington DC: IEEE Computer Society, 2007: 711-716.

[10] SHATNAWI O. Modelling software fault dependency using function [J]. *Al Manarah Journal for Research and Studies*, 2007, 15(6): 261-300.

(上接第 2533 页)开发项目的优化资源分配顺序提供了更有效的解决思路。

参考文献:

[1] 陈冬宇,邱苑华,杨敏,等.基于设计结构矩阵理论的产品开发过程建模及仿真优化[J]. *计算机集成制造系统*, 2008, 14(4): 661-666.

[2] KOLISCH R. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation[J]. *European Journal of Operational Research*, 1996, 90(2): 320-333.

[3] HARTMANN S, BRISKORN D. A survey of variants and extensions of the resource-constrained project scheduling problem[J]. *European Journal of Operational Research*, 2010, 207(1): 1-14.

[4] BROWNING T, YASSINE A. Resource-constrained multi-project

scheduling: priority rule performance revisited [J]. *International Journal of Production Economics*, 2010, 126(2): 212-228.

[5] CHEN Jia-qiong, ASKIN R G. Project selection, scheduling and resource allocation with time dependent returns[J]. *European Journal of Operational Research*, 2009, 193(1): 23-34.

[6] PETEGHEM V, VANHOUCKE M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem [J]. *European Journal of Operational Research*, 2010, 201(2): 409-418.

[7] BROWNING T R, EPPINGER S D. Modeling impacts of process architecture on cost and schedule risk in product development [J]. *IEEE Trans on Engineering Management*, 2002, 49(4): 428-443.