故障察觉率可变的软件可靠性增长模型①

吴开贵，张 烨，尹鲁燕，陈 镭，李明辉

(重庆大学 计算机学院，重庆 400044)

摘 要：证明了基于 G-O 模型的 NHPP 类型的软件可靠性增长模型不需要考虑不完美排错和排错过程中新错误的引入，并在该基础上提出了一种新的软件可靠性增长模型。该模型在软件排错过程中不但考虑了软件开发员对系统熟悉程度的上升，而且考虑了系统现存错误数的不断减少，是一种故障检测率随时间变化的软件可靠性增长模型。并利用现有的公开发表的数据对该模型进行测试，发现其达到了比 G-O 模型的等其他模型更好的拟合效果。

关键词：非齐次泊松过程；故障检测率；软件可靠性增长模型；非完美排错；故障移除效率

Software Reliability Growth Model with Variable Fault Detection Rate

WU Kai-Gui, ZHANG Ye, YIN Lu-Yan, CHEN Lei, LI Ming-Hui

(Chongqing University, Chongqing 400044, China)

Abstract: It is proved that SRGM based on NHPP type of G-O model doesn't need to consider imperfect debugging and new mistakes during the debugging process. A new type of SRGM comes out which does not only consider the software developer's familiarity with software system, but also consider the diminishing mistakes developing system. This SRGM has considered the fault detection rate changing with the time changing. Moreover, the test result by using the public reported data shows that the goodness of fit is better than that of other models of G-O model.

Key words: non-homogeneous poisson(NHPP); fault detection rate; software reliability growth model(SRGM); imperfect debugging; fault removal efficiency

随着计算机软件规模的扩大，功能的日益复杂，使得软件的可靠性成为衡量软件质量的一个重要的标准。为此，在过去的几十年中产生了多种基于非齐次泊松分布（Non-Homogeneous Poisson, NHPP）软件可靠性增长模型来评估软件的可靠性，如最早出现的 J-M 模型，在其基础之上由 Goel 和 Okumoto 提出的 G-O 模型[1]，该模型首次将软件的不完美排错考虑到模型之中。之后又由 Yamada 改进，提出的 s-Shaped 模型[2,3]，该模型考虑了软件的 S 型的软件可靠性增长模型。G-O 模型由于其结构简单，估计精确，因此，很多模型都在其基础之上，对其某些假设进行该进，修正该模型，使新建立的模型达到更好的拟合效果。

在很多基于 G-O 的软件可靠性增长模型中，很多新的模型考虑了软件的非完美排错，该情况一般认为有两个方面：首先是在故障的排除过程中会引入新的错误[1-4]，其次是在开发过程中故障的移除效率[5-7]，即在开发过程中故障被完美排除的概率。

本文分析了以上两种情况，并证明在基于 G-O 模型建立的新模型不需要着重考虑以上两个方面，已达到在不失精确性的情况下简化模型的结构。并提出一种新的考虑故障检测率随时间变化的软件可靠性增长模型。最后，通过实验，证明了新模型有着更为精确的拟合性能。

1 关于给完美排错的证明

1.1 符号解释：

t：时间。

N(t)：0-t 时间内错误的累计发现数。

m(t)：0-t 时间内错误的期望发现数。

① 基金项目:国家自然科学基金(90818028)

收稿时间:2011-01-10;收到修改稿时间:2011-02-24

b：故障发觉率初值，即：每个故障被检测到的概率。

b(t)：随时间变化的故障发觉率。

λ(t):失效强度函数，即：单位时间内的失效数。

a：故障总数初始值。

R(x|t)：可靠度，软件在 t 到 t+x 时间内不发生故障的概率，t 为最近一次发生故障的时间。

1.2 非完美排错的证明：

排错的不完美性主要体现在故障的排除过程中会引入新的错误以及开发过程中故障的移除效率这两个方面，下面对是否需要考虑这两个方面进行证明：

a)新错误的引入：

实际上，G-O 模型认为符号 a 代表的是软件最终发现的故障数，所以该模型实际上已考虑了错误的引入。其次，即便是认为该模型符号 a 代表故障总数初始值，文献[4]中也已经进行了证明，在此简要说明：

由于故障的排除以及新错误的引入都是一个随机的过程，所以假设在排除故障的过程中新引入错误的概率为β，由此，G-O 模型由：

$$\frac{dm(t)}{dt}=b(a-pm(t))$$

变为：$m'(t)=\frac{a}{1-\beta}(1-e^{-(1-\beta)bt})$

令：$a'=\frac{a}{1-\beta}$

$b'=(1-\beta)\phi$

那么：

$$m'(t)=a'(1-e^{-b't})$$

可以发现 m(t)与 m’(t)在形式上是完全相同的，所以，在样本不断完善的过程中，不论是否考虑错误的引入，模型的参数也在不断精确。因此，该条件在模型的精确性上并不起决定性的影响。

b)故障移除效率：

假设，故障的移除效率为 p，那么 G-O 模型变为:

$$\frac{dm(t)}{dt}=b(a-p*m(t))$$

求解该常微分方程，可得：

$$m(t)=\frac{a}{p}(1-e^{-pbt})$$

令 $a'=\frac{a}{p}$

$b'=pt$

那么：

$$m''(t)=a'(1-e^{-b't})$$

可以发现 m(t)与 m’’(t)在形式上也是完全相同的，所以结论同上，该条件在模型的精确性上并不起决定性的影响。

2 错误察觉率变化的软件可靠性模型

由以上的证明我们可以看出，在基于 G-O 模型：$m(t)=a(1-e^{-bt})$ 的建模过程中，是否会引入新的错误与故障移除效率会随着样本的变化而变化，这两点对模型的精确性影响不大。那么参数 b，即软件故障发觉率，则对软件可靠性增长模型建模的精确性起到了很大的作用。

在软件的开发过程中，软件开发人员逐渐的熟悉系统，对故障的察觉率产生积极地影响；故障数随着调试时间在不断地减少，随着软件开发时间的增加，软件中的故障察觉难度也不断的增加。所以，软件的故障察觉率应该是由以上两个方面共同决定的。因此，我们形成了如下的几点假设：

1) 软件的失效符合非齐次泊松过程；

2) 测试人员的学习能力[8-10]是一个与测试时间相关的非递减函数

3) 系统中遗留错误的察觉率是一个与时间相关的非递增函数；

4) 软件中故障的排除是相互独立的；

5) 软件的失效与软件中现存的未排除的故障数成正比。

根据以上的几点假设，我们可以构建出模型如下：

$$\frac{dm(t)}{dt}=b(a-pm(t)) \quad (1)$$

由于 b 是随时间变化的函数，故障的差距率主要与人类的学习能力和系统中遗留的错误数察觉率有关，且 b∈[0,1]，所以：

$$b_1(t)=1-(1-b)e^{-k_2t} \quad (2)$$

$$b_2(t)=be^{-k_2t} \quad (3)$$

其中 $b_1(t)$是考虑了人类学习能力上升的故障察觉率，$b_2(t)$ 是考虑了系统中剩余故障越来越难以发现的故

障察觉率，所以，综合考虑这两点因素，软件的故障察觉率 b(t)为：

$$b(t)=b_1(t)*b_2(t)=e^{-k_2t}-(1-b)e^{-(k_1+k_2)t} \quad (4)$$

$$m(0)=0 \quad (5)$$

可解得：

$$m(t)=a(1-e^{\frac{(\exp(-k_2t)-1)}{k_2}-\frac{(1-b)(\exp(-kt)-1)}{k}}) \quad (6)$$

由于模型符合非齐次泊松过程，所以到时间 t 时：

$$\mathrm{P}\{\mathrm{N}(t)=n\}=\frac{m(t)^n}{n!}e^{-m(t)} \quad (7)$$

再根据泊松分布，则：

$$\mathrm{R}(x\mid t)=\mathrm{P}\{\mathrm{N}(t+x)-\mathrm{N}(t)=0\}=e^{-m(t+x)-m(t)} \quad (8)$$

3 实验分析

对软件可靠性增长模型性能的测试一般使用误差平方和(Sum of Squared Errors, SSE)与回归曲线方程相关指数（R-square）来分析。其中，SSE 越小越好，而 R-square 越接近于 1 越好。

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

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

本文使用文献[11]中的提供的数据进行试验，并与 G-O 模型，Delayed s-shaped 模型，Inflection s-shaped 模型，Bbell-SRGM 模型[8]这些经典和新的模型进行对比，得到如下的结果：

表 1 实验数据

Test period(week)	CPU hours	Defects found
1	519	16
2	968	24
3	1430	27
4	1893	33
5	2490	41
6	3058	49
7	3625	54
8	4422	58
9	5218	69
10	5823	75
11	6539	81
12	7083	86
13	7487	90
14	7846	93
15	8205	96
16	8564	98
17	8923	99
18	9282	100
19	9641	100
20	10000	100

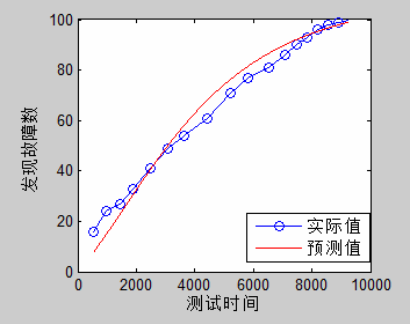

图 1 本文提出的模型对数据的拟合

由图 1 我们能够看出，模型的曲线呈现出 S 形。这表明，在测试阶段的初期，软件中的错误数还比较多，所以人的学习能力在这个阶段主导着故障察觉率的，在软件测试后期，软件的故障越来越难被发现，所以，软件中由错误数控制的故障察觉率慢慢的转变为故障察觉率的主导地位。这个情况也更符合实际。

本文采用以上数据进行预测并与现有的一些模型进行比较，可得表 2：

表 2

	SSE	R-square
G-O 模型	155.2	1.0707
Delayed s-Shaped 模型	824.1	1.2510
Inflection s-shaped 模型	144.6	1.0419
Bbell-SRGM 模型	147.3	0.9929
本文提出的模型	140.5	1.0546

从上表中可以对 SSE 和 R-square 参数中可以看出，在这组数据中，本文提出的模型达到了较好的拟合特性，对软件故障数的预测准确性更高。值得一提的是，Bbell-SRGM 模型虽然也综合考虑了人类的学习能力与系统中遗留的错误数，但由于函数选取不同，导致了 Bbell-SRGM 模型在求解的过程中遇到了积分

困难的问题，通过将定积分转化为求和求解。本文采用的函数在积分中可以直接求得，不需要转化为求和，因此在精确性上达到了更优。

4 结论

利用软件可靠性增长模型可以判断出软件的最优发布时间和测试的开销力度[12]。更精确的软件可靠性增长模型无疑对软件的投资与利润带来了准确的预测。本文提出的模型证明了基于 G-O 模型改进的模型不需要考虑非完美排错过程，又提出了一种随时间变化的故障检测率模型，该故障检测率不但与软件中剩余的错误数有关，而且还与人类的学习能力有关，综合考虑这两方面的因素，使模型的建立更加符合实际情况。经实验表明，该模型在实验数据上达到了很好的拟合。

参考文献

1 Goel AL, Okumoto K. Time-dependent error-detection rate model for software and other performance measures. IEEE Trans.s on Reliability, 1979,28(3):206−211.

2 Yamada S, Ohba M. S-Shaped Software Reliability Growth Models and their Applications. IEEE Trans. on Reliability, 1984,33(4):289−292.

3 Yamada S, Ohba M. S-shaped Reliability Growth Modeling for Software Error Detection. IEEE Trans. on Reliability. 1983,32(5):475−484.

4 Ohba M, Chou XM. Does Imperfect Debugging Affect Software Reliability Growth?Proc. 11th International Conference on Software Engineering, 1989: 237−44.

5 Zhang XM, Teng XL, Pham H. Considering Fault Removal Efficiency in Software Reliability Assessment. IEEE Trans. on Systems Man and Cybernetics Part A-Systems and Humans, 2003,33(1):114−120.

6 Wu YP, Hu QP, Xie M, Ng SH. Modeling and Analysis of Software Fault Detection and Correction Process by Considering Time Dependency. IEEE Trans. on Reliability, 2007,56(4):629−642.

7 Huang CY, Lin CT. Software Reliability Analysis by Considering Fault Dependency and Debugging Time Lag. IEEE Trans. on Reliability, 2006,55(3):436−450.

8 Hsu CJ, Huang CY, Chang JR. Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor. Applied Mathematical Modelling, 2011,35(1):506−521.

9 Pham H, Nordmann L, Zhang ZM. A General Imperfect-Software-Debugging Model with S-shaped Fault-Detection Rate, 1999,48(2):169−175.

10 Liu HW, Yang XZ, Qu F, Dong J. A Software Reliability Growth Model with Bell-Shaped Fault Detection Rate Function. Chinese Journal of Computers, 2005,28(5):908−913.

11 Wood A. Predicting software reliability. IEEE Computer. 1996,29(11):69-77.

12 Huang CY, Kuo SY, Lyu MR. An Assessment of Testing-Effort Dependent Software Reliability Growth Models, 2007,56(2):198−211.

(上接第 130 页)

运用最广泛的 BP 神经网络进行讨论，使用其他神经网络，以及如何在神经网络中吸取模糊和灰色理论的优点，使该方法既有方法的先进性，又有现实的可行性，仍将是一个值得深入研究的问题。另外车祸库如何合理有效的提取、存储车祸特征信息也使一个值得深入研究的问题。

参考文献

1 易白.新技术有效地减少车祸.知识就是力量,2009,(10):58−59.

2 刘唐志,杨贤俊.基于车辆行驶记录仪法的道路危险路段排查.黑龙江交通科技,2009,32(5):155.

3 郑培.机动车驾驶员驾驶疲劳测评方法的研究[博士学位论文].北京:中国农业大学,2001.

4 郑培,宋正河,周一鸣.基于 PERCLOS 的机动车驾驶员驾驶疲劳的识别算法.中国农业大学学报,2002,7(2):104−109.

5 马艳丽,王要武,裴玉龙.疲劳与驾驶时间关系的实验心理学研究.西南交通大学学报,2009,44(4):535−540.

6 施彦,韩力群,廉小亲.神经网络设计方法与实例分析.北京:北京邮电大学出版社,2009.23−25.

7 黄胜伟,董曼玲.自适应步长 BP 神经网络在水质评价中的应用.水利学报,2002,(12):119−12.