基于NHPP类软件可靠性通用模型研究与应用

费 琪 刘春裕
(江苏自动化研究所 连云港 222061)

摘 要 对现有NHPP类软件可靠性模型进行分析总结,指明了已有NHPP类软件可靠性模型存在的不足及缺陷。综合考虑缺陷探测率、软件运行覆盖率、排除错误时的错误引入率等软件故障数的影响因素,提出了一种通用的NHPP类软件可靠性模型,最后对通用模型中的泛函数取特殊值后,求得期望故障数及软件可靠度,并对其进行分析,证明了所提模型的有效性。

关键词 通用的NHPP类软件可靠性模型,非完美排错,缺陷探测率,测试覆盖率
中图法分类号 N945.12 **文献标识码** A

General NHPP Software Reliability Research and Application

FEI Qi LIU Chun-yu
(Jiangsu Institute of Automation, Lianyungang 222061, China)

Abstract This paper analyzed the existing non homogeneous Poisson process (NHPP) software reliability model, pointd out the problems and defects of existing NHPP software reliability model. Considering the defect detection rate, coverage, software error introduction rate, the paper proposed a general NHPP software reliability model. Finally, the general model was applied to a set of published failure data, proofing the effectiveness of model.

Keywords General NHPP software reliability model, Imperfect debugging, Defect detection rate, Test coverage

随着软件被广泛地应用于社会各个领域,软件尤其是安全关键软件的失效很可能会导致生命与财产的灾难性损失。软件可靠性已成为软件发布时用户最为关心的质量指标之一。软件可靠性模型是评估软件可靠性的主要工具。模型利用软件测试时采集到的故障数据来预测软件实际运行时的可靠度,从而决定是否将该软件投入使用。在众多软件可靠性模型中,非齐次泊松过程(NHPP)类模型由于拟合效果好、结构简单,因此成为最重要应用最广泛的软件可靠性模型。

1 NHPP类软件可靠性模型分析

G-O模型是最典型的NHPP模型。1979年,A. L. Goel和K. Okumoto提出了关于连续时间的NHPP模型,即NHPP类G-O模型。之后,以此为基础,又有许多人提出了不少类似模型,他们的工作对软件可靠性模型的建立和应用,产生了重大的影响[1]。但是,经分析现有的NHPP模型,在设计上会有各种考虑不周或缺陷。

G-O模型框架:

$$\frac{dm(t)}{dt}=b[a-m(t)] \tag{1}$$

其中,$m(t)$为期望故障数;b为故障检测率;a为软件预期故障总数。

G-O模型成功地将期望故障数、故障检测率、软件预期故障数相关联,开创了NHPP类可靠性模型先河,但有如下缺点:测试人员在探测缺陷的过程中,随着时间的推移,故障越难被发现,故障检测率不可能为常数;在发现故障、解决故障过程中,必然会引起新的故障,故软件的故障总数不可能为常数。

文献[2,3]成功分析了人类在测试软件过程中随着时间的推移发现故障能力的特点,并引入了软件覆盖率,通过这两个特性来考虑故障检测率,模型框架如下:

$$\frac{dm(t)}{dt}=\phi(t)[a-m(t)] \tag{2}$$

$$\phi(t)=\alpha(t)\beta(t) \tag{3}$$

其中,$m(t)$为期望故障数;a为软件预期故障总数;$\phi(t)$为故障检测率;$\alpha(t)$为单位时间测试工作量,$\beta(t)$为单位时间软件运行覆盖率。

文献[2,3]在G-O模型的基础上,成功分析了故障检测率,唯一缺陷是仍未对软件预期故障总量进行分析。

文献[4,5]在考虑故障检测率的同时,成功考虑了软件预期故障总数为非常数的情况,模型框架如下:

$$\frac{dm(t)}{dt}=\phi(t)[a(t)-m(t)] \tag{4}$$

$$a(t)=N+\beta m(t) \tag{5}$$

其中,$m(t)$为期望故障数;$a(t)$为软件预期故障总数;$\phi(t)$为故障检测率;N为初始软件故障总数;β为常量。

文献[4,5]在考虑故障检测率的同时,成功假设了软件在t时刻的软件预期故障数与期望发现故障数成正比,但是在提出模型(4)时有假设"发现的错误彼此是独立的",故发现的故障越多并不意味着预期故障数会增多。

基于以上分析,本文提出了一种通用的NHPP类软件可

费 琪(1985—),男,硕士,工程师,主要研究方向为软件可靠性、软件形式化规约,E-mail:feiqixia@163.com;**刘春裕**(1981—),男,高级工程师,主要研究方向为软件工程化。

靠性模型，模型综合考虑了故障检测率、预期故障总数及故障排除率，从而更好地对现有的 NHPP 类软件可靠性模型进行通用。

2 通用的 NHPP 类软件可靠性模型

为了构建描述 NHPP 类软件可靠性通用模型，需做如下假设：

1)软件运行剖面与可靠性测试剖面相同；

2)在任何时间序列构成的时间区间中检测到的错误数是相互独立的；

3)在 t 时刻检测到的累积错误数 $[N(t),t\geqslant 0]$是一个独立增量过程，$N(t)$ 服从期望函数为 $m(t)$ 的 Poisson 分布；

4)在时间间隔 $(t,t+\Delta t)$ 内，单位时间所发现的故障数的期望值与软件残留的故障数成比例，比例为故障检测率 $\phi(t)$，$0\leqslant\phi(t)\leqslant 1$；

5)在时间间隔 $(t,t+\Delta t)$ 内，单位时间解决的错误数与单位时间内期望发现的错误数成正比，比例为错误排除率 $\varphi(t)$，$0\leqslant\varphi(t)\leqslant 1$；

6)排除错误为非完美排错，时间 t 时刻预期的错误总量 $a(t)$与预期初始错误数 β 之差与排除的错误数成正比，比例为错误引入率 $\chi(t)$，$0\leqslant\chi(t)\leqslant 1$。

依据假设 4)有：

$$\frac{dm(t)}{dt}=\phi(t)(a(t)-\tau(t)) \tag{6}$$

其中，$\tau(t)$为软件运行至 t 时刻已解决的错误数。

依据假设 5)有：

$$\frac{d\tau(t)}{dt}=\varphi(t)\frac{dm(t)}{dt} \tag{7}$$

依据假设 6)有：

$$a(t)=\beta+\chi(t)\tau(t) \tag{8}$$

其中：

$$m(0)=0,\tau(0)=0,a(0)=\beta \tag{9}$$

联立方程式(6)－式(9)可求得通用模型的期望故障数为：

$$m(t)=\int_0^t\phi(\xi)[\beta+\chi(\xi)\exp(\int_0^\xi\varphi(\zeta)\phi(\zeta)\chi(\zeta)d\zeta)\int_0^\xi(\exp(-\int_0^\xi\varphi(\zeta)\phi(\zeta)\chi(\zeta)d\zeta))\varphi(\xi)\phi(\xi)(\beta-\tau(\xi))d\xi-\tau(\xi)]d\xi \tag{10}$$

由于到时刻 t 为止的累计故障数 $N(t)$ 服从均值为 $m(t)$ 的非齐次泊松分布，所以有：

$$P\{N(t)=n\}=\frac{(m(t))^n}{n!}\exp(-m(t)),n=0,1,2\cdots \tag{11}$$

根据非齐次泊松分布的性质，可靠度函数为：

$$R(x|t)=P\{N(t+x)-N(t)=0\}=\exp(m(t)-m(t+x)) \tag{12}$$

3 通用模型的特殊函数取值

3.1 故障检测率的考虑

在软件人员测试过程中，随着测试的进行，软件的问题将越来越难以被发现，用 $\delta(t)$ 代表错误探测率，则 $\delta(t)$ 满足：

$$\lim_{t\to\infty}\delta(t)=0 \tag{13}$$

随着时间趋于无穷大，软件的覆盖率将无限趋近于 100%，用 $\lambda(t)$ 表示软件覆盖率，则 $\lambda(t)$ 满足：

$$\lim_{t\to\infty}\lambda(t)=1 \tag{14}$$

基于式 (13)及式 (14)，给出满足条件的错误探测率及软件覆盖率如下：

$$\delta(t)=\frac{k}{1+t},\lambda(t)=1-\frac{1}{1+\mu t} \tag{15}$$

其中，k 为初始时刻的缺陷探测率，μ 为尺度参数。故障检测率为缺陷探测率与软件运行覆盖率的乘积，用下式来表示：

$$\phi(t)=\delta(t)\lambda(t)=\frac{k}{1+t}(1-\frac{1}{1+\mu t}) \tag{16}$$

3.2 软件预期故障数的考虑

关于不完美排错的假设有很多种形式，比较常见且重要的为如下 2 种形式：

1)软件预期故障数与测试时间成正比[6,7-9,10]

$$a(t)=\beta(1+\kappa t) \tag{17}$$

注：式中 β 为软件初始预期故障数，κ 为缺陷引入率。

2)软件预期故障数与探测到的缺陷数成正比[4,5]

$$a(t)=\beta+\kappa m(t) \tag{18}$$

由式(17)知，当 $t\to\infty$ 时，$a(t)\to\infty$，即一个软件的故障数是无穷无尽的，针对一部分软件可能是如此，但是大多数软件的故障数为有限的，式(18)直接假设预期故障数与探测到的缺陷数成正比，同检测到的故障数相互独立矛盾，故不可取。

本文假设预期故障数与排除的故障数成正比，因为排除的故障数越多，引入的新错误数也越多，假设错误排除率及错误引入率为常数，表达式如下：

$$\varphi(t)=\varphi,\chi(t)=\chi \tag{19}$$

将式(19)代入至式(7)及式(8)可得：

$$a(t)=\beta+\chi\varphi m(t) \tag{20}$$

由式(20)可知，虽然 $a(t)$ 仍然与 $m(t)$ 成正比，但是它是基于软件预期故障数与排除的故障数而考虑的。

3.3 软件可靠度分析

将式(16)、式(19)、式(20)代入至式(10)可得软件期望故障数为：

$$m(t)=-\frac{\beta}{(\chi-1)\varphi}+\frac{(1+\mu t)^{\frac{k\varphi(1-\chi)}{\mu-1}}(1+t)^{\frac{\mu k\varphi(\chi-1)}{\mu-1}}\beta}{(\chi-1)\varphi} \tag{21}$$

由式(21)可知，当 $t\to\infty$时，$\mu t\to\infty$，式(21)变为：

$$m(t)=\frac{\beta}{(1-\chi)\varphi}-\frac{\beta}{(1-\chi)\varphi(1+t)^{k\varphi(1-\chi)}} \tag{22}$$

由式(22)可知，伴随着时间的增长，软件期望发现的故障数为时间的增函数，且当 $t\to\infty$ 时，期望发现的故障数将变为常量：

$$m(t)=\frac{\beta}{(1-\chi)\varphi} \tag{23}$$

由非齐次泊松分布软件可靠度函数(12)可知，当 $t\to\infty$ 时，$R(x|t)\to 1$，即随着时间的增长，故障的排除，软件的可靠度将不断提高。

4 最大似然估计及模型评价标准

为了更好地预估期望故障函数中的参数，采用最小误差平方和[12-17](Sum of Squared Errors，SSE)来计算：

$$SSE=\sum_{i=1}^{n}(y_i-m(t_i))^2 \tag{24}$$

其中，n 为测试阶段数，t_i 为第 i 阶段软件累计运行时间，y_i 为第 i 阶段累计发现错误数。

为获得最小误差平方和，分别对 β,χ,φ,k,μ 求偏导，导数为零，得如下方程组：

$$\begin{cases}\dfrac{\partial SSE}{\partial \beta}=\sum\limits_{i=1}^{n}2(y_i-m(t_i))\dfrac{\partial m(t_i)}{\partial \beta}=0\\ \dfrac{\partial SSE}{\partial \chi}=\sum\limits_{i=1}^{n}2(y_i-m(t_i))\dfrac{\partial m(t_i)}{\partial \chi}=0\\ \dfrac{\partial SSE}{\partial \varphi}=\sum\limits_{i=1}^{n}2(y_i-m(t_i))\dfrac{\partial m(t_i)}{\partial \varphi}=0\\ \dfrac{\partial SSE}{\partial k}=\sum\limits_{i=1}^{n}2(y_i-m(t_i))\dfrac{\partial m(t_i)}{\partial k}=0\\ \dfrac{\partial SSE}{\partial \mu}=\sum\limits_{i=1}^{n}2(y_i-m(t_i))\dfrac{\partial m(t_i)}{\partial \mu}=0\end{cases} \tag{25}$$

在约束条件 $0\leqslant\beta<\infty,0\leqslant\chi\leqslant1,0\leqslant\varphi\leqslant1,0\leqslant k\leqslant1,0\leqslant\mu<\infty$ 下，通过迭代可求得 β,χ,φ,k,μ 的最小二乘估计。

可通过 MSE 来度量模型的拟合效果，MSE 方程如下：

$$MSE=\frac{1}{n}\sum_{i=1}^{n}[m(t_i)-y_i]^2 \tag{26}$$

由式(26)可知，MSE 越小，拟合效果越好。

5 实验分析

用 Matlab 对文献[11] 中的数据进行拟合分析，求得的参数为：

$$\beta=108.346,\chi=0.0053,\varphi=0.4116,k=0.457,$$
$$\mu=105.71 \tag{27}$$

拟合图如图 1 所示。

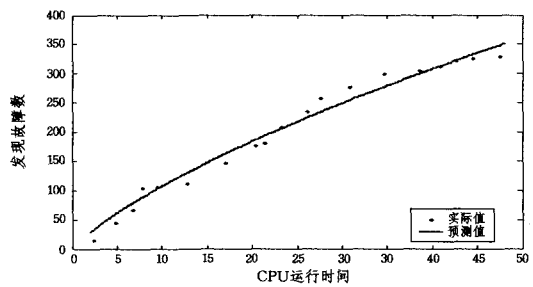

图 1

将式(27)代入式(26)可求得 MSE，与现有模型拟合效果比较如表 1 所列。

表 1

模型名称	MSE
Delayed S-Shaped Model (DS)[2]	168.67
Log-Logistic Coverage Model(LLCM)[3]	194.07
Ohba Imperfect Debugging Model(OID)[4]	139.82
本文提出的模型	105.43

从上表可以看出，本文提出的模型达到了较好的拟合效果，对软件故障数的预测准确性更高，后续研究中可以更好地通过分析缺陷探测率、软件运行覆盖率、排除错误时错误引入率等特性，代入本文提出的模型中，从而达到更好的拟合效果。

结束语 本文通过对现有的 NHPP 类软件可靠性模型进行分析，提出 NHPP 类软件可靠性通用模型，对其中的通用函数进物特殊化，并对提出的模型进行仿真实验及比较，证明了本文提出模型的优良性，在后续研究中可以更好地将分析缺陷探测率、软件运行覆盖率、排错误时错误引入率等特性，代入本文提出的模型中，从而达到更好的拟合效果。

参 考 文 献

[1] 孙志安，裴晓黎，等. 软件可靠性工程[M]. 北京：北京航空航天大学出版社，2009

[2] Pharn H, Zhang Xue-mei. NHPP software reliability and cost models with testing coverage[J]. European Journal of Operational Research, 2003, 145(2): 445-454

[3] Pham H. An imperfect-debugging fault-detection dependent-parameter software[J]. International Journal of Automation and Computing, 2007, 4(4): 325-328

[4] Huang Chin-yu, Lin Chu-ti. Software reliability analysis by considering fault dependency and debugging time lag[J]. IEEE Transactions on Reliability, 2006, 55(3): 436-450

[5] Ahmad N, Khan M G M, Rafi L S. A study of testing-effort dependent inflection S-shaped software Reliability growth models with imperfect debugging[J]. International Journal of Quality & Reliability Management, 2010, 27(1): 89-110

[6] 谢景燕，安金霞，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

[7] Pham H, Zhang Xue-mei. NHPP software reliability and cost models with testing coverage[J]. European Journal of Operational Research, 2003, 145(2): 445-454

[8] Pham H. An imperfect-debugging fault-detection dependent-parameter software[J]. International Journal of Automation and Computing, 2007, 4(4): 325-328

[9] Pham H, Nordmann L, Zhang X. A general imperfect-software-debugging model with S-shaped fault-detection rate[J]. IEEE Transactions on reliability, 1999, 48(2): 169-175

[10] Yamada S, Tokuno K, Osaki S. Imperfect debugging models with fault introduction rate for software reliability assessment[J]. International Journal of Systems Science, 1992, 23(12): 2241-2252

[11] Ohba M. Software reliability analysis models[J]. IBM Journal of Research and Development, 1984, 28(4): 428-443

[12] 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

[13] Huang C Y, Lin C T. Software reliability analysis by considering fault dependency and debugging time lag[J]. IEEE Trans. on Reliability, 2006, 55(3): 436-450

[14] 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 Tram. on Reliability, 2007, 56(4): 629-642

[15] Zhang X M, Teng X L, Pham H. Considering fault removal efficiency in software reliability assessment[J]. IEEE Tram. on Systems, Man, and Cybernetics: Systems and Humans, 2003, 33(1): 1 14-119

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

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