考虑发布后故障发现概率的软件费用模型①

张　烨，吴开贵

(重庆大学 计算机学院，重庆 400044)

摘　要：在基于 G-O 模型的软件可靠性增长模型中引入以时间为变量故障察觉率，并以此模型建立了一种新的软件费用模型。该费用模型考虑了软件发布后使用者对软件系统剩余故障的发现概率，并不是所有的剩余故障全部被发现这种情况，使模型更符合实际。以软件开发费用最小为最优化条件，讨论发布时间与费用的关系，并在文章最后用示例说明了发布时间随参数变化。

关键词：非齐次泊松过程；故障察觉率；软件可靠性增长模型；软件费用模型；最优发布时间

Software Cost Model of Considering the Fault Detection Probability After Software's Publishment

ZHANG Ye, WU Kai-Gui

(College of Comuter Science, Chongqing University, Chongqing 400044, China)

Abstract: Leading time as a variable of fault detection rate into the G-O Model of Software Reliability Growth Model(SRGM), we build a new Software Cost Model. This Software Cost Model considers the remaining fault detection probability of software system users after the software's publishment. It is a model that considers error is not completely found, so the model is more practical. The article is based on the least costs of the software as the best condition and talks about the relationship of publishing time and costs. At the end of this article, examples are showed to explain the publishing time changing by parameters.

Key words: Non-Homogeneous Poisson(NHPP); fault detection rate; Software Reliability Growth Model(SRGM); Software Cost Model; optimal release times

伴随着信息时代的来临，计算机软件系统的规模也在不断的扩大，软件的可靠性和软件的开发费用已成为一个不可忽视软件质量指标。软件的开发费用是与软件的可靠性，维护费等软件的指标密不可分的。在过去的几十年中，为了提前预测软件的可靠性，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，其中以 Goel 和 Okumoto 提出的 G-O 模型[1]最为经典，该模型首次将软件的不完美排错引入其中。在软件费用的预测上，也有很多文章提出了相关的模型[2,3]。

本文在 G-O 模型的基础之上引入了随时间变化的故障察觉率，使模型更加符合实际情况。并在新建立的软件可靠性增长模型基础之上建立了软件的费用模型，该模型考虑了在软件在发布之后的规定时间内，软件使用者对软件中剩余错误的发现概率[4]，而不是简单的假定软件中未排除的错误全部会被使用者发现这种特殊情况，以此分析了在何种情况下何时发布软件可以使软件的开发费用降低到最低。最后，通过简单的实例，直观地说明软件的最优发布时间随软件中各个参数的变化情况。

1　建立软件可靠性增长模型

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(大型分布式软件系统的行为监控与可信演化 项目编号:90818028)
收稿时间:2011-05-19;收到修改稿时间:2011-06-11

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.

P{N(t)=n}:发现概率函数，软件在时间 t 内发现故障数为 n 的概率.

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 模型建立：

以 G-O 模型为基础模型。考虑软件的失效过程，失效的发现是随机点过程，是以 N(t)表示在(0,t](t>0)内发现的失效数的随机数，且 N(0)=0, 则失效的发现过程{N(t)，t≥0}为非齐次泊松过程。故软件的失效符合非齐次泊松过程。

(1) G-O 模型：

G-O 模型的几点假设：

① 软件的失效符合非齐次泊松过程；

② 软件中故障的排除是相互独立的；

③ 故障察觉率是一个恒定的值；

④ 故障一经发现则立即被排除；

⑤ 软件的失效与软件中现存的未排除的故障数成正比；

⑥ 故障的发现与软件中先存的为排除的故障数成正比。

根据以上几点假设，G-O 模型可建立为如下形式：

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

对以上常微分方程求解，可得：

$$m(t)=a(1-e^{-bt})$$

在 G-O 模型中，需要说明以下两点：

① G-O 模型中的 a 表示软件故障的最终预测值，既包含了新错误的引入，而不是一般情况认为的软件初始故障数；

② b 为恒定的故障发现率，不是时间的变化而变化，表示测试人员对故障的发现概率为一恒定值。

(2) 新建立软件可靠性增长模型：

新模型的几点假设：

① 软件的失效符合非齐次泊松过程；

② 软件中故障的排除是相互独立的；

③ 故障察觉率是一个随时间变化（随时间的增加不断降低）的值；

④ 故障一经发现则立即被排除；

⑤ 软件的失效与软件中现存的未排除的故障数成正比；

⑥故障的发现与软件中先存的为排除的故障数成正比。

根据以上假设，我们可以构建出新的模型：

将测试人员的故障发现率表达为与时间相关的函数[5]：

$$b(t)=be^{-kt}$$

其中，b 是故障察觉率初始值，k 是调整参数，其作用可以调整 b(t)的曲线，使之更符合实际情况。可以发现 b(t)是一个随时间增长而减少的减函数，表示在软件故障的排除过程中，剩余故障数减少，故障越来越难以被发现，故障的察觉率随时间的降低的过程。

将新的变化的故障察觉率带入到常微分方程中，可得新的软件可靠性增长模型：

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

求解该常微分方程，可得：

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

由于模型符合非齐次泊松过程，所以到时间 t 时：

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

再根据泊松分布，则：

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

2 建立软件费用模型

软件费用模型的一般形式如下：

$$\mathrm{E}=\mathrm{E}_1+\mathrm{E}_2+\mathrm{E}_3+\mathrm{E}_4$$

其中各符号表达的含义如下

E：软件开发总费用的期望值；

E1：软件的设计及初期开发费用，在软件设计初期制定，为一定值；

E2：软件测试费用，单位为时间内，软件的测试费用一定；

E3：故障排除费用，与软件的故障发现数相同，与建立的软件可靠性增长模型有关；

E4：软件发布后的维护费用等，该费用也与建立

的软件可靠性怎么故障模型有关。

其中每个符号都为一个简单费用表达式：

① $E_2 = C_2 * t$

其中 C2 为单位时间内的测试费用，t 为时间。

② $E_3 = C_3 * m(t)$

其中 C3 为一个故障的排除费用，m(t)为到 t 时间内错误的期望发现数。

③ 现存的费用模型中，一般认为

$$E_4 = C_4 * (a - m(t))$$

C4 为软件发布后，使用者发现故障，并交回开发商，修复一个故障的费用。C4 在实际情况中远远大于 C2 和 C3。该费用表示，在软件发布之后，软件中剩余的未在测试期间发现的错误将会在发布后全部被使用者发现，并被要求修正。

在实际情况中，由于使用者的不同以及使用者对软件操作剖面的不用，实际上，并不是所有的故障都会被软件的使用者发现，所以，将所有的错误计算在内会将软件的最优发布时间推后，且不符合实际情况。考虑软件的错误发现符合泊松分布，并且软件在规定时间内不发生错误的概率符合下式：

$$R(x|t) = P\{N(t+x) - N(t) = 0\} = e^{-m(t+x)-m(t)}$$

所以我们改进发布后费用为：

$$E = C_4 * (1 - R(x|t))$$

其中，C4 为软件发布后期最高维护费用（该费用表示软件在发布后最坏情况下的维护费用），t 为发布时间，x 为发布后的软件质保时间，可根据实际情况而定。

由此我们建立了一个更符合实际情况的软件费用模型：

$$E = E_1 + E_2 + E_3 + E_4 = E_1 + C_2 * t + C_3 * m(t) + C_4 * (1 - R(x|t))$$

3　实验分析

本文使用文献[6]中的提供的数据，对软件可靠性增长模型进行拟合，并使用拟合后的软件可靠性增长模型计算软件的开发费用。

表 1　软件可靠性增长模型实验数据

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

参考文献[7]中参数，并进行简单修改，能够得出以下四种情况：

(1) 取 C_2 =5，C_3 =30，C_4 =100000，E_1 = 500000

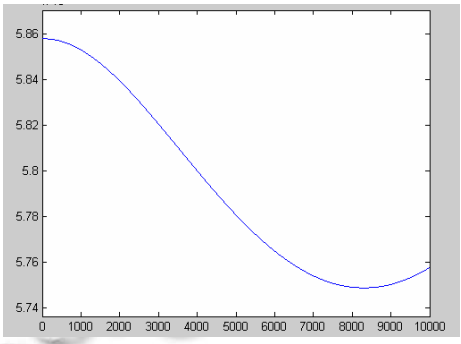

图 1

由图 1 我们能够看出，模型的曲线呈现出先降低后增加的情况。该情况表明，软件在排错的过程中，故障越来越少，软件发布后被用户发现故障的概率越来越低，由于软件在发布后改错费用远远大于发布前改错，所以在前期，费用的消耗随着时间的变化降低，但是当软件排错到达一定时间后，测试期软件的错误发觉时间与错误排除的费用超过了发布后排错的费用，故软件的费用又有所升高。由此可得出软件的最优发布时间。这种情况是最符合实际的情况。在软件的实际制作过程中，多数情况都符合这种。

(2) 取 C2 =5，C3 =30，C4 =500000，E1 = 500000

由上图我们能够看出，软件的费用在最迟发布时

间前处于不断降低的过程中，这种情况是由于软件发布后错误的修正费用过高，导致在软件的排错过程中，测试期软件的错误发觉时间与错误排除的费用始终无法超过软件的后期维护费用，而导致软件的费用在不断的减少。实际情况中该现象不常发生。

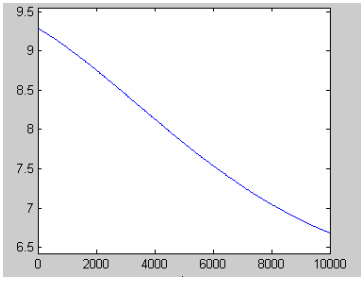

图 2

(3) 取 C_2 =10，C_3 =30，C_4 =100000，E_1 = 500000

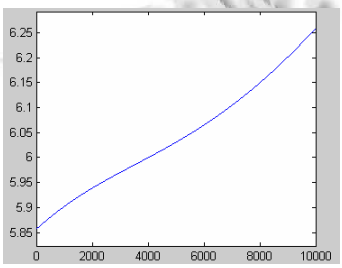

图 3

由上图我们可以看出，软件的费用在不断的升高，这种情况是由于测试期软件的错误发觉时间与错误排除的费用的上升速率要比发布后软件的错误排除费用的下降速率快很多，所以，软件在开发过程中，测试期软件的错误发觉时间与错误排除的费用占主导作用，软件总费用在不断的上升，在这种情况下，我们应该直接将软件发布出去，让用户发现错误，反馈给开发者进行错误的修正。实际情况中该现象不常发生。

(4) 取 C_2 =5，C_3 =60，C_4 =100000，E_1 = 500000

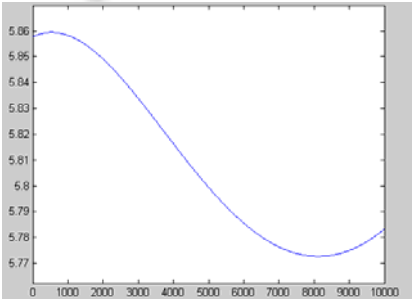

图 4

由上图我们能够看出，软件的费用呈现出先增后减再增的情况。这种情况表明，软件的排错初期，由于软件的测试期错误排除费用稍高，导致在一定时间时，软件剩余的故障数也很多，若在此时发布，则后期故障的修正费用与之前测试期的故障排除费用之和会呈现一个最大值，从而导致了软件此时的费用达到一个峰值。之后的情况与分析 1）相同。该情况也是实际情况中较常发生的一种情况。

4 结论

利用软件费用模型可以较为准确的预知软件的最优发布时间。与以往的模型相比，该新模型并不是简单的将发布后软件中剩余的错误数全部计算，而是考虑符合非其次泊松过程的故障出现概率，通过该概率的计算，可知软件在发布后的维护费低于将所有剩余故障全部计算的费用，使得软件的最优发布时间提前，达到了一种更符合实际情况的最优的软件的开发费用。

参考文献

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

2 Hou RH, Kuo SY, Chang YP. Optimal Release Times for Software systems with scheduled delivery time based on gthe HGDM. IEEE Trans. on Computers, 1997,46(2):216–221.

3 Yun WY, Bai DS. Optimum Software Release Policy with Random Life Cycle. IEEE Trans. on Reliability, 1990, 39(2):167–170.

4 Pham H, Zhang XM. A Software Cost Model with Warranty and Risk Costs. IEEE Trans. on Computer, 1999,48(1):71–75.

5 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.

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

7 Zhang XM, Pham H. A software cost model with error removal times and risk costs. System Science, 1998,29:435–442.