基于测试级别的软件成本模型的研究

舒燕君[1]　吴智博[1]　刘宏伟[1,2]　杨孝宗[1]
(1. 哈尔滨工业大学计算机科学与技术学院, 黑龙江 哈尔滨　150001; 2. 北京图形研究所, 北京 100029)

摘　要: 很少有软件成本模型会考虑测试的实际情况,因此不能够准确地反映测试过程中成本的实际消耗以及当测试情况需要发生改变时对软件成本需求的变化。绝大多数的软件测试存在着不完全排错和学习过程等现象,这些现象通常反映了实际测试的成本使用情况。本文提出了一个反映实际测试情况的测试级别,在此基础上构造了一个新的软件成本模型,并讨论了基于最低软件成本的最优发布策略。实验证明该成本模型更符合软件测试的实际情况,软件开发者可以利用该模型实现对测试成本科学的管理。

关键词: 软件测试; 软件成本模型; 软件可靠性增长模型; 测试级别
中图分类号: TP311

引　言

软件测试是提高软件质量的主要手段,测试的时间越长,软件的质量越高,软件的可靠性就越高,测试的成本也越高。然而用户要求的交付期限是既定的,测试不可能无限制地进行下去,必须在交付期限之前结束。何时结束软件测试、进行软件发布是一个综合考虑软件可靠性,相关软件成本以及合同交付期的问题[1]。在过去的 20 年中,已经有许多软件成本模型从可靠性和成本的角度提出了软件的最优发布策略。其中,绝大多数的成本模型都是通过利用软件可靠性增长模型(SRGMs)来建立软件开发的成本模型,通过计算成本模型的最小值来确定软件的最优发布策略[2]。

SRGMs 是软件可靠性工程实践中非常成功的工具,它能够得到测试中的很多实际特性,如不完全排错和学习过程[3]。这些特性反映了软件测试的实际情况,许多学者认为对于软件开发者来说掌握这些特性是非常重要的[4-7],但是很少有软件成本模型会考虑到这一点,所以不能够准确的预测软件的最优发布时间以及当调整这些参数时对软件成本的影响。

Xie 等[8]使用测试级别指标来描述实际的测试情况,并使用 SRGMs 中的故障排除效率 p 作为测试级别。在他们的成本模型中测试级别只是对测试时间的成本部分进行了调整,排除故障的成本也会受到测试级别的影响。事实上,测试时间的成本与排除故障的成本是紧密联系的。当测试时间的成本越高时,故障的检测也就越有效,故障检测率也会越高,开发者的学习强度也较高,故障也较易排除[9]。此外,在 Xie 建立的软件成本模型中,假设完全排错是不存在的,当测试级别趋近于 1 时,对于测试时间成本的影响函数趋近于无穷,这种假设是不必要的,因为当测试人员掌握了故障的本质时,故障是能够被完全排除的。

本文通过分析 Zhang 等建立的 Z-T-P 模型[10],讨论了不完全排错和学习过程对于软件测试时间成本和排错成本的影响,构造了一个测试级别函数;在研究测试级别对测试时间成本和故障排除成本的影响以及这两种成本之间的相互关系的基础上提出了一个基于测试级别的软件成本模型,给出了软件的最优发布策略。最后,通过实例对模型进行了解释,并讨论了测试级别发生变化时测试成本的改变。

1　不完全排错和学习过程对软件成本的影响

软件的最优发布策略通常是从软件最小成本出发来确定软件的发布时间,这些成本中包括测试阶段的排错成本、运行阶段的排错成本和测试时间的成本。基于 NHPP 类的 SRGMs 可以建立如下传统的软件成本模型[11]

收稿日期: 2007-04-30
基金项目: 国家自然科学基金(60503015)
第一作者: 女,1981 年生,讲师,博士生
E-mail: syj@ftcl.hit.edu.cn

$$C(T) = C_1 m(T) + C_2[m(\infty) - m(T)] + C_3 T \quad (1)$$

式中，T 是软件的发布时间，$m(T)$ 是到时刻 T 为止的故障累积数的期望值。C_1 是测试阶段排除单个故障的成本，C_2 运行阶段排除单个故障的成本，显然，$C_2 \gg C_1$。C_3 是软件在测试阶段$[0, T]$的单位测试时间成本。当 $T = T^*$ 时，$C(T^*)$ 取 $C(T)$ 的最小值，软件的最优发布时间 T^*。

G-O 模型是一个简单而基本的 NHPP 类的 SRGM，它假设测试情况是理想的，故障累积数的期望值函数表达式为

$$m(t) = a(1 - e^{-bt}) \quad (2)$$

式中，a 是软件中的初始故障，b 是残余故障发现率。使用一组实时控制系统的失效数据[10]进行模型讨论，利用极大似然估计 MLE 可以对模型中的参数进行估计。其中，$\hat{a} = 125$，$\hat{b} = 0.000056$。假设 $C_1 = 60$，$C_2 = 3600$，$C_3 = 700/3600$（测试时间单位是 s）[12]，求式(1)的极小值得到最优发布时间 $T^* = 81873$。实际上，测试过程并没有在 81873 s 停止，因为此时的软件中还残留大量的故障。G-O 模型由于没有考虑故障排除效率和故障引入等测试中的实际情况，对软件中残余的故障数估计过小，导致了最优发布时间的估计也过早。

Z-T-P 模型对失效数据的拟合度和预测能力都要好于其他的模型[10]，其故障累积数的期望值为：

$$m(t) = \frac{a}{p-\beta}\left[1 - \left(\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}}\right)^{(c/b)(p-\beta)}\right] \quad (3)$$

其中，p 是故障排除率，β 是故障引入率，α 是学习过程的强度参数。使用同一组失效数据进行 MLE，$\hat{a} = 135$，$\hat{b} = 0.001$，$\hat{c} = 0.000035$，$\hat{p} = 0.9$，$\hat{\alpha} = 0.01$，$\hat{\beta} = 0.012$[10]。在成本模型的参数不变的情况下，可以计算得出 $T^* = 143220$。与 G-O 模型的结果相比较，Z-T-P 模型估计的最优发布时间显然更加合理。由此可见故障排除效率、故障引入和学习过程存在于实际的测试过程中，它们对测试进程的影响是不可忽略的。

下面讨论这些实际现象对软件成本的影响。假设软件的发布时间固定，p，β，α 分别发生变化时的软件成本变化如表 1～3 所示。

可以看出 p，β，α 变化对软件成本产生影响。当 p 越大，β 越小，α 越小时，软件的成本越低。这是因为故障排除越完全，引入的故障越少，排除故障所需的成本也会降低，而且用于代码复审和回归测试的成本也会降低。实际上，为了提高测试级别，开发者通常需要投入更多的成本等，传统的成本模型因为没有考虑到实际测试中成本的使用情况，使这些额外的成本在表 1～3 中不能够体现。导致软件的预算过低，使得测试不够充分，软件质量也会因此而降低。

表 1 故障排除效率 p 改变时成本的变化

Table 1 Change of cost with fault removal efficiency p

p	$C(T)$
0.60	58818
0.70	43692
0.80	35423
0.90 *	30787
1.00	28115

表 2 故障引入率 β 改变时成本的变化

Table 2 Change of cost with fault introduction rate β

β	$C(T)$
0.002	30450
0.012 *	30787
0.022	31144
0.032	31520
0.042	31919

表 3 学习强度参数 α 改变时成本的变化

Table 3 Change of cost with learning process intensity α

α	$C(T)$
0.01 *	30787
0.02	30789
0.03	30791
0.04	30793
0.05	30795

2 基于测试级别的软件成本模型

2.1 考虑多种情况的测试级别

本文的模型考虑测试中的不完全排错和学习过程，因此测试级别应当是一个能够反映这两种情况的综合指标。如果定义测试级别为 l，通过上一小节中的分析可以得出，当 p 越大，β 越小，α 越小时测试情况越好，测试级别也越高，所以做如下假设

(1) 测试级别 l 是由 p，β 和 α 构成的函数；

(2) 测试级别 l 是 $p - \beta$ 的单调递增函数；

(3) 测试级别 l 是 α 的单调递减函数。

为讨论方便,根据上述假设可以得到测试级别的形式为

$$l = (p - \beta)(1 - \alpha) \tag{4}$$

l 能够反映测试过程中的不完全排错和学习过程的变化情况,由于 $0 < p \leq 1, 0 \leq \beta < 1, 0 \leq \alpha < 1$ 且 $\beta \ll p$, l 的取值范围为 $(0, 1]$,当 $l = 1$ 时,测试级别最高,是测试中最理想的情况。这个测试级别函数是一个考虑不完全排错和学习过程的简单形式,软件开发者可以根据具体情况进行变化。

2.2 基于测试级别的软件成本模型

软件成本模型(1)中与测试阶段有关的成本参数 C_1、C_3 的大小都取决于测试中对测试工具、测试员结构所投入的成本,而这些直接影响着故障的不完全排除和学习过程等情况,也就是测试的级别。因此,l 作为测试情况的综合指标将会根据实际的测试情况对成本参数 C_1、C_3 进行调整,那么 C_1、C_3 应当是 l 的函数,记为 $C_1(l)$、$C_3(l)$,下面对这两个函数的特性进行讨论。

通常,如果较高的测试级别需要较好的测试工具或测试队伍,单位时间的测试成本 $C_3(l)$ 也会随之增加,因此 $C_1(l)$ 关于测试级别 l 递增。与此同时故障能够得到更有效的检测,所需要的排除时间也会因此而缩短,从而单个故障排除所需要的成本 $C_1(l)$ 也会降低,因此 $C_1(l)$ 关于测试级别 l 递减。此外, C_1、C_3 通常是根据以往开发软件系统的经验数据估计得出,但进行估计时,一般考虑测试情况对应于测试级别的常规标准,假设其值为 l_0。当测试过程高于测试级别的常规标准($l > l_0$),单位时间的测试成本 $C_3(l)$ 增长较快,需要增加更多的成本,排除单个故障的成本 $C_1(l)$ 降低也会更加缓慢。反之,当测试过程低于测试级别($l < l_0$)的常规标准,单位时间的测试成本 $C_3(l)$ 降低的也就越慢,测试成本减少变得缓慢,排除单个故障的成本 $C_1(l)$ 的增加也会越快。

假设 $\overline{C}_1$、$\overline{C}_3$ 分别是普遍测试情况下的软件成本模型参数,即 $C_1(l_0) = \overline{C}_1$, $C_3(l_0) = \overline{C}_3$,根据 $C_1(l)$、$C_3(l)$ 随 l 变化趋势的特点,可以构造出以下函数

$$C_1(l) = \overline{C}_1 e^{l_0 - l}, \quad C_3(l) = \overline{C}_3 e^{l - l_0} \tag{5}$$

从等式(5)中可以看出 l 增大时, $C_3(l)$ 随 l 呈指数型增大, $C_1(l)$ 随 l 呈负指数型减小。将上述基于测试级别变换后的参数应用于成本模型(1)中,可以得到一个考虑软件测试实际情况的成本模型(6)。

$$C(T, l) = C_1 e^{l_0 - l} m(T) + C_2 [m(\infty) - m(T)] + C_3 e^{l - l_0} T \tag{6}$$

3 软件最优发布时间的讨论

从基于测试级别的软件成本模型的最小值出发,讨论软件最优发布时间。设

$$y(T) = \frac{\mathrm{d}C(T, l)}{\mathrm{d}T} = (\overline{C}_1 e^{l_0 - l} - C_2) \frac{ac}{1 + \alpha e^{bT}} \left[\left(\frac{(1 + \alpha) e^{-bT}}{1 + \alpha e^{-bT}} \right)^{(c/b)(p - \beta)} \right] + \overline{C}_3 e^{l - l_0} \tag{7}$$

$$k(T) = \frac{\mathrm{d}^2 C(T, l)}{\mathrm{d}T^2} = v(T)[C - u(T)] \tag{8}$$

$$v(T) = (C_2 - \overline{C}_1 e^{l_0 - l}) \frac{ac}{(1 + \alpha e^{-bT})^2} \cdot \left[\left(\frac{(1 + \alpha) e^{-bT}}{1 + \alpha e^{-bT}} \right)^{(c/b)(p - \beta)} \right] \tag{9}$$

$$C = c(p - \beta), \quad u(T) = \alpha b e^{-bT} \tag{10}$$

从以上等式中可以看出 $v(T) > 0$, $u(+\infty) = 0 < C$, $y(\infty) = C_3 e^{l - l_0} > 0$。

定理 1:给定 C_1、C_2、C_3, $y(T)$, $k(T)$ 分别是对 $C(T, l)$ 关于 T 的一次和二次导数,通过取 $C(T, l)$ 取最小值得到软件的最优发布时间 T^*,软件的最优发布时间可以描述为

(1)如果 $u(0) \leq C$,当 $y(0) \geq 0$,则 $T^* = 0$;

(2)如果 $u(0) \leq C$,当 $y(+\infty) \leq 0$,则 $T^* = +\infty$;

(3)如果 $u(0) \leq C$,当 $y(0) < 0$, $y(+\infty) > 0$ 则 $T^* = y^{-1}(0)$;

(4)如果 $u(0) > C$,则 $\exists T^0$, $u(T^0) = C$,当 $y(T^0) \geq 0$,则 $T^* = 0$;

(5)如果 $u(0) > C$,则 $\exists T^0$, $u(T^0) = C$,当 $y(0) < 0$, $\exists T_a \in [T_0, +\infty)$, $y(T_a) = 0$ 则 $T^* = T_a$;

(6)如果 $u(0) > C$,则 $\exists T^0$, $u(T^0) = C$,当 $y(0) \geq 0$, $\exists T_c \in [T_0, +\infty)$, $y(T_c) = 0$,则 $T^* = T_c$。

根据高等数学中的有关定理易证定理 1。

4 实例应用

4.1 不同测试级别下的最优发布情况

根据爱尔兰软件测试公司 LTS 的研究,不完全排错的概率一般在 15 %左右[4]。由于测试人员的经验在不同的测试过程中变化很大,不考虑学习过

程的强度在测试级别的常规标准中。因此,常规标准的测试级别是 0.85。本节将使用两组具有不同测试级别的软件系统对新的软件成本模型进行验证。

(1) 实时控制系统[10]

根据这组失效数据,Z-T-P 模型参数的 MLE 为 $\hat{a}=135$, $\hat{b}=0.001$, $\hat{c}=0.000035$, $\hat{p}=0.9$, $\hat{\alpha}=0.01$, $\hat{\beta}=0.012$ 那么测试级别 $l=0.87912$。$\overline{C}_1=60$, $C_2=3600$, $\overline{C}_3=700/3600$,将这些参数带入等式(7),(10)中可以得到

$$u(0)<C,\ y(0)<0,\ y(+\infty)>0$$

根据定理 1, $T^*=142420$ s,比传统模型得出的最优发布时间为 143220 要少 800 s,由于在实际的实时控制系统测试境况要好于成本预算时估计的普遍的测试情况,测试级别也要高于常规标准,因此可以缩短实际的测试周期。事实上根据 Z-T-P 模型的预测, $m(142420)=150.2086$, $m(143220)=150.2533$,在 142420 后测试的 800 s 时间内发生一次失效的概率只有 0.0437,测试结束时间实际上可以提前,以节省测试的成本。

(2) Tandem 公司计算机系统[10]

根据这组失效数据,Z-T-P 模型参数的 MLE 为 $\hat{a}=103.6$ $\hat{b}=0.095$ $\hat{c}=0.0001363$ $\hat{p}=0.63$, $\hat{\alpha}=0.00039$, $\hat{\beta}=0.00054$,那么此次测试的级别 $l=0.62921$。根据 AT&T 研究者收集的经验数据可以假设 $\overline{C}_1=60$, $C_2=600$, $\overline{C}_3=1.9$[9],将这些参数带入等式(7),(10)中可以得到

$$u(0)<C,\ y(0)<0,\ y(+\infty)>0$$

根据定理 1 可以得到 $T^*=14469$,比传统模型得出的最优发布时间为 12100 要多 2369 h 的测试时间,这是因为在实际的 Tandem 公司计算机系统测试情况很差,测试级别低于常规标准的测试级别,实际的测试周期应当增长。按照失效数据显示,每周的 CPU 执行 359 h,因此实际测试时间比预先估计得要多 6.6 周。根据此结论,在测试的过程中应当投入更多的测试成本来以采用新的测试工具和技术,雇佣更多、更有经验的测试人员等,改善测试情况,提高测试级别。

4.2 测试级别发生变化时的成本的改变

通常当测试进行到 60 %以上的时间,SRGM 的变化将趋于稳定,此时软件开发者将密切监控测试情况和 SRGM 的变化情况直至软件测试结束[13]。在这个阶段,软件开发者为了在测试结束前达到既定的可靠性指标,需要根据测试的实际情况和 SRGM 对软件可靠性的估计,对测试情况进行密切监控。描述考虑测试工期和可靠性指标的软件成本模型为

$$\text{Minimize } C(T,l)=\overline{C}_1 e^{l_0-l}m(T)+C_2[m(\infty)-m(T)]+\overline{C}_3 e^{l-l_0}T$$

$$\text{Subject to } R(x/T)=e^{-[m(T+x)-m(T)]}\geq R_0 \tag{11}$$

其中, $R(x/T)$ 是可靠性函数,代表系统在 $(T, T+x)$ 时间内不发生失效的概率。假设 $R(x/T_R)\geq R_0$,根据定理 1,基于此模型(11)的软件最优发行时间是 $\max\{T^*, T_R\}$[13]。假设上述实时控制系统软件在 140000 s 结束测试,需要达到 $R(180/T)\geq 0.99$ 的目标。

Zhang 等使用前 122 次的失效数据进行 Z-T-P 模型的参数估计,接近整个测试过程的 60 %。假设软件开发者在排除第 122 个故障(时间为 57042 s),根据当前软件测试情况计算 $R(180/140000)=e^{-[m(140180)-m(140000)]}=0.9891$。为了在测试结束前达到可靠性指标,软件开发者将提高测试级别。经过计算,当故障排除效率 p 提高 2 %,故障引入率 β 和学习强度 α 降低 2 %时,刚好可以达到可靠性标准, $R/(180/140000)=0.09$。测试情况改进后的测试级别为 $l'=0.89736$,测试级别需要从原来的 0.87912 提高至 0.89736,软件可以在 140000 s 停止测试时达到可靠性指标。下面对软件成本的变化进行计算,假设测试在 $T_1=57042$ s 后进行了测试级别的调整,那么测试进行到 $T=140000$ 停止时,需要增加的成本的表达式为

$$\Delta C=\overline{C}_1 e^{l_0-l'}[m'(T)-m(T_1)]+\overline{C}_3 e^{l'-l_0}(T-T_1)-\{\overline{C}_1 e^{l_0-l}[m(T)-m(T_1)]+\overline{C}_3 e^{l-l_0}(T-T_1)\} \tag{12}$$

将新的参数带入式(12),得到 $\Delta C=417$。因此,软件开发者至少要增加 417 个单位的成本才能在测试期限到来时,达到软件交付的可靠性指标。

5 结论

本文分析了不完全排错和学习过程对软件成本影响,提出了一个描述实际测试情况的测试级别函数。在传统的软件成本模型基础上,建立了基于测试级别的软件成本模型,并应用多组公开发表的数据对这个模型的不同情况进行讨论。实验证明,与

传统的成本模型相比,基于测试级别的软件成本模型更符合软件测试的实际情况。软件开发者可以利用该模型对不同测试级别的测试情况进行模拟,通过调整测试人员的数量、不同经验的测试人员或不同的测试工具或来实现所需要的故障排除效率,故障引入率和学习过程强度,以达到对测试成本科学的管理。

参考文献:

[1] LYU M R. Handbook of software reliability engineering [M]. New York: McGraw-Hill and IEEE Computer Society, 1996.

[2] PHAM H. Software reliability and cost models: Perspectives, comparison, and practices[J]. European Journal of Operational Research, 2003, 149(3): 475 - 489.

[3] PHAM H, ZHANG X. NHPP software reliability and cost models with testing coverage[J]. European Journal of Operational Research, 2003, 145(2): 445 - 454.

[4] BOLAND P J, CHUIV N. Optimal times for software release when repair is imperfect[J]. Statistics & Probability Letters, 2007, 3 (4): 1 - 5.

[5] ABU G, CANGUSSU J W. A quantitative learning Model for software testing process[C]//38th international conference on system science, Hawaii, 2005.

[6] PHAM H. A software cost model with imperfect debugging, random life cycle and penalty cost[J]. International Journal on Systems Science, 1996, 27(5): 455 - 463.

[7] HUANG C Y, LYU M R. Optimal release time for software systems considering cost, test-effort, and test efficiency[J]. IEEE Transaction on Reliability, 2005, 54 (4): 583 - 591.

[8] XIE M, YANG Bao. A study of the effect of imperfect debugging on software development cost [J]. IEEE Transaction on Software Engineeriog, 2003, 29 (5): 471 - 473.

[9] EHRLISH W, PRASANNA B, STAMPFEL J. Determiningthe cost of a stop-test decision [J]. IEEE Software, 1993, 33(3): 33 - 42.

[10] ZHANG Xuemei, TENG Xiaolin, PHAM H. Considering fault removal efficiency in software reliability assement[J]. IEEE Transaction on system, man, and cybernetics-part a: systems and human, January 2003, 33 (1): 114 - 120.

[11] YAMADA S, OHTERA H, NARIHSA H. Software reliability growth model with testing effort [J] IEEE Transaction on reliability, 1986, 35(1): 19 - 23.

[12] PHAM H, ZHANG Xuemei. A software cost model with warranty and risk costs[J]. IEEE Transaction on computers, 1999, 48(1): 71 - 75.

[13] HUANG C, LIN C. Software reliability analysis by considering fault dependency and debugging time lag [J]. IEEE Transaction on Reliability 2006, 55 (3): 436 - 450.

Study of a software cost model based on testing level

SHU YanJun　WU ZhiBo　LIU HongWei　YANG XiaoZong

(1. School of Computer Science and Technology, Harbin Institute of Technology, Harbin Heilongjiang, 150001;
2. Beijing Graphic and Image Institute, Beijing 100029, China)

Abstract: Since few software cost models take into account the actual cost of software testing, they can not reflect actual consumption of cost and the needed cost when the testing circumstance changes. Imperfect debugging and learning process exist in most software testing processes which usually reflect the actual cost in software testing process. In this paper, a testing level function which reflects actual testing circumstance is proposed, and a new software cost model is developed based on the testing level function. The optimal release polices to minimize the expected total software cost are discussed. Numerical experiments are presented and the results show that software cost model based on the testing level reflect the consumption of the testing resources more realistically. By using this model, software developers can manage testing cost scientifically.

Key words: software testing; software cost model; software reliability growth model; testing level