Analysis of Software Release Problems Based on Fault Detection and Correction Processes*

Haiyan Sun[1], Jia He[1], Bingfeng Xie[1], Ji Wu[2]

[1]School of Mathematics and Systems Science, Beihang University, Beijing, China
[2]School of Computer Science and Engineering, Beihang University, Beijing, China
[1]sunhy@buaa.edu.cn, [1]hejia129@126.com, [2]wuji@buaa.edu.cn

Abstract-In this paper, the modeling of fault-correction process from the viewpoint of correction time is first discussed. By proposing a new cost model, further analysis on the optimal release time determination is presented, which is also based on the model incorporating both fault detection and correction processes. The experiment results show it is more suitable to the reality. Finally, we propose a new solution to the assignment of testing resource. The approach is also illustrated with an actual data set from a software development project.

Index Terms - fault detection process, fault correction process, cost model, optimal release time

1. Introduction

Nowadays, software is playing an increasingly important role in our daily life. As a result, software reliability has become an important aspect of software quality. Numerous models for the software testing process have been developed to measure software reliability, and among these models, software reliability growth models (SRGM) are most widely used in practice [1-3]. Software reliability models provide important information for determination in many software development activities.

Testing is an efficient way to remove faults in software products. However, exhaustive testing of all the paths in a program is not practical. Therefore, it is very important to determine when to stop testing, which is usually called the optimal release time problem. If the length of testing is long, we can remove many faults in the system and its reliability increases, while it may increase the testing cost and delay software delivery. In contrast, if the length of testing is short, a software system may still have many faults remained and its lower reliability may cause loss to users. As a result, the maintenance cost during the operation period increases.

The solution of this problem of the optimal release time is to minimize cost model respect to time. Many existing models assumed that the detected faults can be removed instantly and perfectly, that means the time to remove and fix faults is negligible. However, in practice, removing and correcting faults is critical in testing. It is necessary to consider the correction time in the cost model. In this paper, we model the fault correction process and a combination model of fault detection and correction processes could achieve greater accuracy in decision making of the optimal release time.

The paper is organized as follows. In Section 2, the models for fault detection and correction are presented in detail. In Section 3, a new cost model is discussed .In Section 4, an example is shown using these models on a real data set. The determination of the release time with these models is also presented. In Section 5, a new approach to assign resource is discussed. In Section 6, conclusions are drawn.

2. Models on Fault Detection and Correction Processes

A. Fault detection models

A fault detection process is assumed to follow a non-homogeneous Poisson process (NHPP). Once given the intensity function of fault detection $f_d(t)$, the mean value function (MVF) can be derived as

$$m_d(t)=\int_0^t f_d(s)ds \tag{1}$$

The most well-known SRGM based on NHPP is the model proposed by Goel and Okumoto. This model assumes that the failure detection rate per fault in the testing phase is constant. The MVF $m_d(t)$ could be derived by

$$m_d(t) = a\cdot(1-e^{-bt}),\ \ a,b>0 \tag{2}$$

where a is the total number of faults that can be detected during the testing phase and b can be interpreted as the failure detection rate per fault.

B. Fault correction models based on the correction time

A fault can be corrected only after its detection, thus, fault correction is related to its detection. We can introduce a delay $x(t)$, the time between fault detection and correction process. A fault correction process can be model as a delayed fault detection process. The delay can be model as deterministic or random delay. Then, like the fault detection models characterized with a MVF $m_d(t)$, fault correction models also are characterized with a MVF $m_c(t)$ .By the relationship of $m_d(t)$ and $x(t)$, we can obtain the mean value function $m_c(t)$.

Similar to the fault detection process, given the intensity function of fault correction $f_c(t)$, the mean value function

*Grant sponsor: State Key Laboratory of Software Development Environment (SKLSDE-2013ZX-12); Aeronautical Science Foundation of China (20121951021)

(MVF) of fault correction is

$$m_c(t)=\int_0^t f_c(s)ds\,. \quad (3)$$

Here we show some classical delay models. Simply, we use the GO model for illustration.

1) Deterministic correction time

Assume that each detected fault need the same amount of time to be corrected, i.e. $x(t)=x_0$. Then the related fault correction process can be described by the following MVF

$$m_c(t)=\int_0^t f_d(s\text{-}x_0)ds,\ t\ge x_0\ . \quad (4)$$

Specifically, the MVF of the fault correction process as follows

$$m_c(t)=a\cdot(1\text{-}e^{-b(t\text{-}x_0)}),\ t\ge x_0\ . \quad (5)$$

With the limited assumption, Xie and Zhao [4] extended the time lag to be time-depended. As the detected faults become increasingly difficult to remove and correct, the time delay becomes much longer. The fault correction process can be described by the following MVF

$$m_c(t)=\int_0^t f_d(s\text{-}x(s))ds\ . \quad (6)$$

The specific fault correction process MVF of GO models is:

$$m_c(t)=a\cdot(1\text{-}(1+ct)e^{-bt})\,. \quad (7)$$

2) Fault correction models based on random correction time

The deterministic assumptions on the time delay are not usually realistic in practice. Sometimes the fault correction is full of uncertainty factors. Therefore, it would be more reasonable to model the time delay with a random variable. The time delay is known to follow an exponential distribution, i.e. $x(t)\sim\exp(p)$, which is proved in some practical software testing project [3].

Specifically, the fault correction process MVF of GO models follows,

$$m_c(t)=a\cdot(1\text{-}(1+bt)e^{-bt}),\ p=b \quad (8)$$

, while if $p\neq b$, then

$$m_c(t)=a\cdot(1\text{-}p/(p\text{-}b)\cdot e^{-bt}+b/(p\text{-}b)\cdot e^{-pt})\,. \quad (9)$$

3. The Determination of Release Time

The quality of the software usually depends on many factors. The testing methodologies and the time spent on the testing are two important factors. The longer time spent in testing, the more faults can be removed, which leads to the more reliable software; however, the testing cost will also be higher. On the other hand, if the testing time is too short, although the cost could be reduced, the software may take a higher hidden risk. It is much more expensive to fix a fault during operational phase than testing phase. Therefore, it is important to determine the optimal release time.

There are a lot of papers dealing with software release time problem [5-7]. Based on some classical cost models [8], here we propose a new cost model,

$$C=c_1\cdot m(t)+c_2\cdot[m(\infty)\text{-}m(t)]+c_3\cdot t+c_4\cdot(1\text{-}R(x|t)) \quad (10)$$

$$R(x|t)=\exp(\text{-}m(x+t)+m(t)) \quad (11)$$

in which c_1 is the cost of removing a fault during testing period, c_2 is the cost of removing a fault during operation period, c_3 is the cost per unit time of testing and c_4 is the risk cost due to software failure.

By incorporating the fault-correction process $m_c(t)$, which is different from the traditional SRGMs, the cost model can be expressed as

$$C=c_1\cdot m_c(t)+c_2\cdot[m_d(\infty)\text{-}m_c(t)]+c_3\cdot t+c_4\cdot(1\text{-}R(x|t)) \quad (12)$$

, in which $m_c(t)$ is the total number of the corrected fault at the release time t, $m_d(\infty)\text{-}m_c(t)$ represents the number of uncorrected faults. By minimizing the cost model according to the time, an optimal release time T^* can be obtained.

4. Numerical Examples

In this section, we will apply the proposed models for both fault detection and correction processes on an actual data set [9]. The data set is from the testing process on a middle-size software project. The data set include not only fault detection data but also fault correction data and are summarized as the cumulative number of faults per week.

A. *Criteria for comparison*

Mean Squared Errors (MSE)

The goodness-of-fit can be measured by MSE. MSE is the average of the MSE_d for the fault-detection and the MSE_C for fault-correction process. Both MSEs can be calculated through the average squared difference between the estimated expectations and actual data, as in the following equation

$$\begin{aligned} MSE &= (MSE_d+MSE_c)/2 \\ &=\sum_{i=1}^{n}((m_d(t_i)\text{-}d_i)^2+(m_c(t_i)\text{-}c_i)^2)/(2\cdot n) \end{aligned} \quad (13)$$

in which d_i, c_i denote the cumulative number of detected faults and corrected faults until time t_i respectively and t_i is the running times after the beginning of testing.

B. *Performance analysis*

We select three models for both fault detection and correction processes, which are shown in Table I below. By using the Least Squares Estimation method, we can estimate the parameters of selected models. The results of the estimation with the corresponding goodness-of-fit measure for all models are listed in Table II below.

From the table, we can find out that the Model 1 with constant correction time, the MSE value of which is the lowest than the other models, provides the best fit for the actual data set.

Having found Model 1 fitted the actual data better both fault detection and correction processes, in the following section we will apply mean value function of Model 1 to our cost model.

TABLE I Three Models

Model Name	Models
Model 1	$m_d(t) = a \cdot (1\text{-}e^{-bt})$ $m_c(t) = a \cdot (1\text{-}e^{-b(t-x_0)})$
Model 2	$m_d(t) = a \cdot (1\text{-}e^{-bt})$ $m_c(t) = a \cdot (1\text{-}(1+ct)e^{-bt})$
Model 3	$m_d(t) = a \cdot (1\text{-}e^{-bt})$ $m_c(t) = a \cdot (1\text{-}p/(p\text{-}b) \cdot e^{-bt})$ $+ a \cdot b/(p\text{-}b) \cdot e^{-pt})$

TABLE II Performance

Model Name	Estimation	MSE
Model 1	$a = 153.01$ $b = 0.1487$ $x_0 = 1.6390$	55.47
Model 2	$a = 158.30$ $b = 0.1460$ $c = 0.0459$	113.20
Model 3	$a = 156.01$ $b = 0.1501$ $p = 0.90$	69.21

C. *Estimation of parameters in the cost models*

According to the previous experiences and related papers, the cost coefficients have such correlations. The cost of removing a fault after release is much higher than the cost of removing a fault while testing, i.e. $c_2 > c_1$ .The risk cost coefficient c_4 , is the cost due to software failures. It varies according to the applications. The cost may include the loss of customers, and even the human life. Therefore, we assume the values ranging from 3,000 to 5,000 to describe the impact of the risk cost coefficient on the total cost and the optimal release time.

Based on the information above, we assume the following coefficients in the cost model as the base line case:

Case1: c_1=300, c_2=500, c_3=100, c_4=3000 .

In order to find out the impact of the coefficients on the total cost and the optimal release time, we vary some of the coefficients. By given Cases 2-4 with some different coefficients, we compare them with Case 1. The results of the total costs and the corresponding release time are presented in Table III.

TABLE III Optimal Release Times and Minimum Total Costs

Case me	Conditions	T/weeks	C
Case 1	$c_1 = 300$, $c_2 = 500$, $c_3 = 100$, $c_4 = 3000$	31.8	49,770
Case 2	$c_2 = 1000$	37.6	50,338
Case 3	$c_4 = 5000$	33.6	49,961

Note: the coefficients in the column of "Conditions" only list the modified ones.

D. *The impact of the cost coefficients*

Compared the other coefficients c_2, c_4 , c_3 is relatively stable and negligible from the perspective of magnitude. We vary the values of c_2, c_4 and keep the value of the other parameters unchanged.

In Case 2, we increase the cost coefficient c_2 , which represents the cost of removing a fault during operation period, from 500 to 1,000. Compared the results with Case 1, we find that the cost coefficient c_2 has remarkable impact on the optimal release time. The optimal software release time T^* of Case 2 is 37.6 weeks, while that of Case 1 is 31.8 weeks. That means, if people need to pay much more to fix faults during operation period than testing period, the developers will spend more time on testing. We can have such conclusion: increasing the cost coefficient c_2 will lead to a longer testing time.

In Case 3, we increase the risk cost coefficient c_4 from 3,000 to 5,000 and compare the results of the optimal release time. We can find out that the optimal release time T^* of Case 3 is 33.6 weeks, while that of Case 1 is 31.8 weeks. The risk cost coefficient c_4 has significant effect on the optimal release time. Compared the difference between Case 2 and Case 3, the total cost of Case 2 has more significant increase than Case 3. In fact, there are some safe-critical software systems which have a high reliability requirement. To these software systems, software failure may result in inestimable loss. In order to gain a high reliability, the developers usually take a lot of time and do testing again and again. According to the above discussion, we could make the conclusion: increasing the risk cost coefficient c_4 will also lead to a longer testing time.

On the other hand, different from the traditional analysis with a single fault-detection model, we use the combined fault-detection and fault-correction models discussed above. The traditional model leads to a different result. With an earlier optimal release time $T^* = 29.4$ weeks, the difference with Case 1 shows the effects of correction time.

5. A Discussion on the Resource Assignment Problem

Based on the above discussion, we may consider how to assign the resource in turn. We only discuss the cost model with no risk cost or correction time for simplicity. So the cost model is given as

$$C = c_1 \cdot m(t) + c_2 \cdot [m(\infty)\text{-}m(t)] + c_3 \cdot t \ . \qquad (14)$$

As we analysis the simplified cost model based on GO model, the cost model could be expressed as

$$C = c_1 \cdot a + a \cdot (c_2 - c_1) \cdot e^{-bt} + c_3 \cdot t \,. \quad (15)$$

Software reliability $R(t)$ is an increasing time-dependent function, while the cost model C is a decreasing then increasing function. We now want to study how to determine the cost coefficients c_1, c_2, c_3 or their relationship to achieve the cost minimization after the software reliability reaches a certain value.

We can use the ratio of $R(t_2)$ to $R(t_1)$ to describe the situation of software reliability. The ratio can also be written as the following form:

$$\begin{aligned} R(t_2)/R(t_1) &= (R(t_2) - R(t_1) + R(t_1))/R(t_1) \\ &= 1 + (R(t_2) - R(t_1))/R(t_1), \ t_2 > t_1 \end{aligned} \,. \quad (16)$$

In order to minimize the cost model with respect to time, we obtain the derivative is:

$$dC/dt = a \cdot b \cdot (c_1 - c_2) \cdot e^{-bt} + c_3 \,. \quad (17)$$

As mentioned above, the cost coefficient c_3 is relatively stable and negligible respect to the other two cost coefficients c_1, c_2. Here we assume the ratio of c_3 and c_1 is less than 20%.

After multiplying the reciprocal of c_1 on both sides of (17), then (17) becomes:

$$dC/(c_1 \cdot dt) = a \cdot b \cdot (1 - c_2/c_1) \cdot e^{-bt} + c_3/c_1 \,. \quad (18)$$

According to (16), the ratio reaches a certain value means that the ratio of $(R(t_2) - R(t_1))$ and $R(t_1)$ reaches a certain value (denoted by l_0). By solving the following inequality, we can gain the critical point of the time t_0 to satisfy the inequality.

$$(R(t_2) - R(t_1))/R(t_1) \le l_0 \quad (19)$$

By choosing the optimal ratio of c_2 and c_1, the total cost at the time t_0 can be among the minimum. Here we select some values of l_0 and compare the corresponding results. The results are presented in Table IV.

Table IV Optimal Ratios

l_0	t_0	$R(t_0)$	c_2/c_1
1%	25.2	0.698	1.320~1.480
0.5%	30.2	0.837	1.647~1.971
0.2%	36.6	0.930	2.594~3.391

From Table IV, we can see the lower the value of l_0, the longer the time t_0 and the larger the optimal ratio of c_2 to c_1. The lower the value of l_0, which means the higher demand of software reliability, thus developers need to pay more attention on doing testing, i.e. the longer the time spent on doing testing. As we know, with the higher reliability, it is hard to detect faults and the cost of finding and fixing faults after the release will increase, therefore, along with the higher reliability, the optimal ratio of c_2 and c_1 becomes larger.

On the other hand, the more resource inputs after release don't mean the total cost better. How to reach the minimum cost around the time t_0 is more important. It is meaningful to assign the resource appropriately. With the method discussed above, we can obtain the relationship of the cost doing testing and fixing faults after release once software reliability meets the demand. It provides the information about the resource inputs after release for the developers.

6. Conclusions

In this paper, the cost model based on fault detection and correction processes considers more information about correction time, the optimal release time obtained is more appropriate. We also propose an approach to the problem of resource allocation, which provides a new idea in this problem.

Acknowledgment

The authors thank several referees for their critical reviews, and their comments.

References

[1] M. M.R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill: New York, 1996.

[2] M. M.Xie, *Software Reliability Modeling*, World Science: Singapore, 1991.

[3] M. J. Musa, K.Okumoto, A.Iannino, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill: New York, 1987.

[4] H. M.Xie, M.Zhao, "*The Schneidwind software reliability model revisited*" Proceeding of the 3rd International Symposium on Software Reliability Engineering, 1992, pp.184-192.

[5] Y. M.Kimura, T.Toyota, S.Yamada, "Economic analysis of software release problems with warranty cost and reliability requirement", *Reliability Engineering and System Safety*, vol.66, pp. 49-55.1999.

[6] Y. YS.Dai, M.Xie, KL.Poh, B.Yang,"Optimal testing-resoure allocation with genetic algorithm for modular software systems", *Journal of Systems and Software*,vol.66, pp. 47-55,2003.

[7] Y. M.Xie, GY.Hong, "Software release time determination based on unbounded NHPP model", *Computers and Industrial Engineering*, vol.37, pp. 165-168, 1999.

[8] Y. H.Pham, X.M.Zhang, "A Software Cost Model with Warranty and Risk Costs", *IEEE Transcations on Computers*, vol.48, pp. 71-75, 1999.

[9] Y. M.Xie, Q.Hu, Y.Wu, S.Ng, "A study of the modelling and analysis of software fault-detection and fault-correction processes", *Journal of Quality and Software Reliability Engineering International*, vol.23, pp.459-470, 2007