Research on Software Reliability Evaluation Model Oriented Final Software Testing Process

Zhongwei Chen[1, a], Zhihao Yu[1, b], Xiaoxia Li[2, c], Liang Xiang[1, d]

[1] Beijing Special Vehicles Institute, Beijing, 100072, China

[2] Beijing Pingguoyuan Middle School, Beijing, 100043, China

[a]bigczw@163.com, [b]13501049415@139.com, [c]zw_chen@21cn.com, [d]fasky@sina.com

Keywords: Software testing and evaluation; Software reliability evaluation model; Imperfect fault-removing

Abstract. In the traditional software reliability evaluation models, there are assumptions that the detected faults can be eliminated immediately and the new fault will not be introduced, which does not fit for the real final software testing process. This essay modifies the above assumptions, treats the detection and removing process of software as two separate process modeling, takes into account the influence of staging fault-removing and fault introduction rate, and then presents a software reliability model which is suitable for the real software testing and evaluation process. According to the actual situation, the appropriate the appropriate parameters are selected for the correlation function in the model. Two failure data sets are used to compare and validate the improved model, as a result, the improved model and the actual data have a good degree of consistency, fitting and forecasting effect is better.

Introduction

Software reliability model is the core of software reliability evaluation and prediction. It is a tool to assess software reliability based on the failure data collected from software reliability testing, and it is of great significance to guarantee the quality of software. The model uses the failure history information, and predicts the actual behavior of the software in the actual work, so as to determine whether the software is reliable and whether it can be put to use.

Non-Homogeneous Poisson Process (NHPP) model is the most common type of software reliability growth models. Because of this kind of model fitting effect, simple structure and application, thus it becomes the most important and the most widely used software reliability model. G-O model is the most widely used as the classical model of the NHPP model. This kind of model is mainly focused on the precise modeling during the fault detection process, but the research of fault-removing process is not enough. In order to simplify the complexity of modeling, they generally assume that the fault can be eliminated immediately once it is found, and elimination process does not introduce new fault. Or they consider the fault removal rate and fault introduction rate as constant. These assumptions make the reliability model easier, but also reduce the accuracy of the model and the reliability of the result. The software testing and evaluation process has the characteristics of uniform and imperfect fault-removing which obviously do not meet the above assumptions. If the relevant data of the removing process is obtained, we can connect the real testing process, consider the influence of imperfect fault-removing process and then introduce relevant parameters. Then, the more accurate reliability model can be built. Based on the above reasons, this paper analyzes the characteristics of software testing and evaluation process and puts forward a software reliability model based on the consideration of the asynchronous of detection process and removing process, and imperfect removing. The test results of two data sets prove that the model has more fitting and prediction effect than the other models.

Software reliability growth G-O model

The G-O model is a continuous time NHPP model proposed by L Okumoto and K GoelA in 1979

[1]. It assumes that the fault-removing process is perfect and no new fault will be introduced in this process. In the model M (t) is a bounded monotone function, and M (t) can be used to predict the number of the observed faults when time t arrives. When t approaches infinity, M (t) approaches the upper bound, which is the expected value of the total faults that may eventually be detected.

The basic assumptions of G-O model:

1) Software testing operation mode is the same as the actual operation section;

2) The number of the detected faults at any time is independent of each other;

3) The severity of each fault and the likelihood of being detected is approximately the same;

4) The detected faults can be immediately removed;

5) The cumulative number of the detected faults {N(t), t≥0} at time t is an independent increment process. N(t) obeys Poisson process with a desirability function of m(t) , so that the expected fault number found in (t, t+Δt) is proportional to the expected remaining fault number at t time;

6) The expected function m (t) of the cumulative fault number is a bounded monotonic increasing function, and meets the following condition:

m(0)=0,

$\lim_{t\to\infty} m(t) = c$,

Where "c" is the total number of the detected faults, c>0.

Software reliability evaluation model for final software testing (SREMFFT)

Characteristics of the final software testing process.

1) Final software testing process and development process are independent. The testing staff is only responsible for detecting faults, while the developer is responsible for fault-removing;

2) Software evaluation process has obvious periodicity. When one fault is found, it will not be removed immediately, but it will be removed with all found faults after the completion of a round of testing;

3) The fault be found cannot be truly removed by the removing process and the new fault will be introduced during the removing process.

Model assumptions. Based on the above characteristics, the basic assumptions of the reliability model for final software testing process can be established:

1) Software is running in a test environment similar to the expected operating environment.

2) The fault detection process is a non-homogeneous Poisson process.

3) Software failure is caused by the remaining faults in the software.

4) The ratio of finding faults in each round of test is proportionate to the number of remaining faults in software. This ratio is called fault detection rate, and it is a changeable function over time, expressed with b(t), 0 <b(t)≤1.

5) Each time the software failure is caused by a fault.

6) When fault is found in the i testing round, removing process may not be able to truly eliminate the fault, probability of fault-removing is called fault-removing rate as c (i), r≥i≥1, "r" is a total of testing rounds; fault-removing process also belongs to the NHPP process, and occurs after the detection process.

7) fault-removing process after the test may introduce new fault, and the probability of introducing a new fault is proportional to the number of faults detected. This ratio is called fault introduction rate, which is a function with the change of test rounds, expressed with q (i), q(i)≥0，r ≥i≥1.

(8) After the i round of testing, the number of cumulative software fault elimination x (t) is proportional to m (t), with the ratio of "c (i) -q (i)".

9) The difference between the expected number of software faults a(t) and the number of faults in the initial time a is proportional to x (t), with the ratio of η (t), 0 < η (t)<1.

10) The expected function m (t) of the cumulative fault number is a bounded monotonic

increasing function, and meets the following condition:

m(0)=0,

$\lim_{t\to\infty} m(t) = c$,

Model building. According to the assumption (1) and (4), we can get the following equation:

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

According to the assumption (1), (6), (7) and (8), we can get the following equation:

During the process of the i testing round,

$$\frac{dx(t)}{dt} = (c(i) - q(i))\frac{dm(t)}{dt} \quad , \ r \geqslant i \geqslant 1 \tag{2}$$

Initial conditions are m(0)=0 and x(0)=0. The solution can be available by the assumption (7) and (9) :

During the process of the i round of testing,

$$a(t) = a + q(i)x(t) \quad , \ r \geqslant i \geqslant 1 \tag{3}$$

Simultaneous equations (1) and (2), and the solution can be available

$$x(t) = e^{-\int_0^t b(\tau)(c(i)-q(i))d\tau} \int_0^t a(\tau)b(\tau)e^{\int_0^\tau b(v)(c(i)-q(i))dv} d\tau \tag{4}$$

If we plug equation 4 into equation 1, combined with equation 3, we can obtain:

$$m(t) = \int_0^t b(t)\left[a - e^{-\int_0^t b(\tau)(c(i)-q(i))d\tau} \int_0^t a(\theta)b(\theta)e^{\int_0^\theta b(v)(c(i)-q(i))dv} d\theta\right]d\tau \tag{5}$$

The reliability function after the n-th software failure is

$R_{n+1}(x|T_n=s) = Pr\{X_{n+1}>x|T_n=s\}$

$=\exp\{-[m(x+s)-m(s)]\}$ (6)

Calculation of model function. In the above model, the difference of fault detection rate, fault-removing rate and fault introduction rate directly determine the different types of model, which lead to the difference of the accuracy and the predictive ability of the model.

On the one hand, if the nature of the fault is considered only, the remaining fault detection rate of software becomes lower and lower, and the function of the fault detection rate should be a decreasing function. On the other hand, with the development of the software testing process, testing personnel have a better and better understanding of the software, and have a growing ability to find software fault. This is a learning process. And the test personnel often use a variety of testing methods and various testing tools to improve the fault detection rate. Therefore, changes of software fault detection rate of software evaluation stage should synthesize the influence of the two aspects, and the solution is similar to an increasing function like S shape, in the form of:

$$b(t) = \frac{b^2 t}{1 + bt} \quad , \ 0 < b \leqslant 1 \tag{7}$$

As testing goes on, the category structure of the hidden faults in the software will change, and the efficiency of fault-removing will change accordingly. After the first round of testing, the simple and easy faults are removed and the proportion of the easily removed faults hidden in software will gradually decrease. But the faults which are difficult to find or cannot be removed will gradually accumulate. As a result, the efficiency of fault-removing will be gradually reduced. As testing goes on, if the software detected are found more lately, they are more difficult to be located and modified. Therefore the software removing efficiency is a function of testing rounds, and showed a decreasing trend. So we choose the negative exponential decreasing function to define the fault removing rate.

$c(i)=ce^{-v(i)}$, $v(i) \geqslant 0$ (8)

In equation 8, c is the fault-removing rate of the first round, and v is the changeable parameter of the fault-removing rate. Because removing staff are generally software R & D personnel and they

are very familiar with the developed software, the fault-removing ability remained unchanged in the testing phase and so v is a constant.

The fault introduction rate has a similar change trend to fault-removing rate, which can be expressed as:

$$q(i)=qe^{-v(i)},\ v\geqslant 0 \tag{9}$$

If equation 7, 8, 9 are plugged into equation 5, the specific form of the cumulative number of faults found during the test can be obtained, then the specific forms of the reliability of R_{n+1}（$x|T_n=s$）can be obtained.

Case analysis

In this paper, we use Mean Square Error (MSE) to measure the fitting ability of the new model.

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}[m(t_i)-y_i]^2 \tag{10}$$

In the type 10, m (t_i) indicates the number of expected faults detected by time t_i by using new model; y_i indicates the number of real faults detected by time t_i. The model coincides with the current data better, and the MSE value should be smaller.

In this paper, the two group public data sets Ohba [3] and Wood[3] are used to verify in reality. The verification process is divided into two steps: 1) The model framework is used to establish the reliability model for two groups of open faults data sets; 2) This model in the paper is compared with other classical models.

Table 1 is the results of the comparison between the two data sets.

Table.1. Comparison of fitting results on the two data sets

SN	Model	MSE on the Ohba data set	MSE on the Wood data set
1	G-O Model(GO)	139.82	11.62
2	Yamada Weibull Model(YW)	260.04	16.59
3	Ohba Imperfect Removing Model(OID)	139.82	11.62
4	SREMFFT	86.05	9.41

Data from Table 1 can be seen that the MSE data obtained from the SREMFFT value in the two data sets are minimal. Therefore, compared with other classic models, the fitting ability of the new model is better.

Conclusion

Based on the classical NHPP model, this model in the paper is established with the characteristics of final software testing process. And when it is established, the factor that the removing process, the fault-removing rate and fault introduction rate are not ideal is also considered. So the model is more consistent with the actual situation. And the experimental data prove that there is a good fitting effect and practical significance in this model. But this model has some limitations, such as not taking into account all the fault severity level for different contribution of reliability evaluation, which will become the focus of the next research direction. Due to the lack of software reliability data, this paper uses the other literature published data as the verified original data. And due to the lack of related information of testing rounds in the public data sets, the data fitting work in this paper is limited to a single round of testing. More suitable and reliable data also need to be accumulated so as to be verified and improved in the actual work.

References

[1] Musa J D. Software reliability engineering, 1999.

[2] Minyan Lu . Software Reliability Engineering, 2011.

[3] Huanh c y，kuo s y，lyu m r. An assessment of testing-effort dependent software reliability growth models. IEEE Trans on Reliability, 2007.

[4] Xingyuan Ma, Jianying Guo, Haiqiu Zhao. Improvement of software reliability growth G-O model [J]. Transducer andMicrosystem Technologies, 2011 30(5) 69-71.

[5] Caihua Wu, Xiaowei Wang, Juntao Liu, Jianchao Ma. Software reliability growth G-O model considering multipie real situations[J]. System and Electronics, 2014 36(11) 2332-23361.