TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.5, May 2014, pp. 3634 ~ 3641
DOI: http://dx.doi.org/10.11591/telkomnika.v12i5.5116

Research on Software Fault Distribution for Web Application

Wanjiang Han*[1], Lixin Jiang[2], Sun Yi[1], Li Ye[3], Han Xiao[3], Weijian Li[3], Liu Chi[3]
[1]School of Software Engineering, Beijing University of Posts and Telecommunication, Beijing, China
[2]Department of Emergency Response, China Earthquake Networks Center, Beijing, China
[3]International School, Beijing University of Post and Telecommunication Beijing, China
*Corresponding author, e-mail: hanwanjiang@bupt.edu.cn

Abstract

This paper studies multiple software fault distribution model, then extends the ideas of software fault estimation based on the analysis of a large number of project fault data. It presents the estimation model of software faults distribution for Web Application, which refers to the G-O model and Rayleigh model. This paper fits the fault prediction model and shows the steps to determine the relevant parameters. Firstly, estimating the probability of finding a fault according to similar project data. Then estimating the total number of faults. Software fault estimation model has certain directive significance to the prediction and planning projects.Experiments show that the model has great potential to predict software fault.

Keywords: *fault, fault prediction model, test data, Web application*

1. Introduction

Software fault prediction is a very important research topic in software engineering. It is based on the fault record in historical data to predict the faults in the future. It helps software project planning and process management. At the same time, it provides decision support. Software fault prediction technology developed from the last century 70's. In this period, many fault prediction models are appeared. Its purpose is to count possible faults in software. It can help determine whether the system can be launched and be used. Faults in software products can not meet the needs of users in some extent. Every software development team must know how to properly deal with the faults of the software, which is related to the fundamental quality software development.

Software testing is a very important period of the software life cycle, especially in Web Application. From testing we can obtain fault data. Through studying and analyzing these data, we can estimate faults, then we can effectively improve the project planning and even predict the product release time.

In the process of software testing, fault management is a very important task, which are activities to ensure faults to be tracked and managed in the software life cycle. In general, fault tracking and management needs to achieve the following two objectives: one is to ensure that each deficiency was found to be solved, the other is collecting fault data, recognizing and preventing faults in the future. For fault management, many people only think of how to correct the faults. However, effective preventing faults based on fault analysis are easy to be ignored. In fact, in a project development, collecting and analyzing fault data is very important. From the fault data, we can get a lot of related data about software quality. For example, studying the fault trend curves to determine whether or not the end of test process is commonly used and is an effective way. Fault data common statistical charts include fault trends, fault distribution, fault timely processing statistics and etc.

Therefore, in the actual project, the curve of introducing products is very important index. Study the properties of these curves, which has a certain significance for launching projects and products. Until now, no common model for all project has been reported.Through the study of various kinds of fault models, This paper shows a fault estimating model based on Web application software. It plays a certain role for the prediction and planning projects.

Received November 11, 2013; Revised December 13, 2013; Accepted January 2, 2014

2. Related Software Fault Distribution Model

Software fault estimating technology mainly includes the static prediction technology and dynamic prediction technology. The static prediction technology including Akiyama [1] quantitative model, Halstead [2] model, Lipow [3] model, Takahashi [4] model, Akiyama faultdensity model [5], Boehm COQUALMO model [6-8], Bayesian model [9], Capture-Recapture model [10] and etc. The dynamic model is time relation model, this kind of model is empirical research and statistical technology to find software faults with the distribution relationship between some stages based on time. The main dynamic models include the Rayleigh model, exponential model and S curve model. This paper mainly studies the relationship model between software faults and time during the test period. Therefore, this section mainly describes three models.

2.1. Rayleigh Distribution Model

Rayleigh model [11] is a common reliability model. It can forecast the fault distribution during software life cycle. This kind of model is based on the WeiBull statistical distribution. WeiBull distributed reliability analysis is widely used in different field. The probability density function of WeiBull distributed tag end is gradually converge to 0, but never equal to 0. The experts of Trachtenberg [12] and IBM [13] have researched on the software project faults and they found that the distribution accords with Rayleigh distribution model [14]. The probability distribution density function of Rayleigh model is $f(t) = 2K(t / c^2)e^{-(t/c)^2}$, the cumulative distribution function is $F(t) = K(1 - e^{-(t/c)^2})$,where K is the total faults. T stands for time, c is an constant and $c = \sqrt{2}t_m$, where t_m is the time when $f(t)$ reached maximum, and $F(t_m) / K$ approximately equal to 0.4. Therefore, we can estimate the total number of faults at a certain time as well as the specific Rayleigh distribution parameter. Thus we can simplify the counting process. It is easy to control the quality of the enterprise performance goal by using Rayleigh model. It is necessary to take measures to correct it when the process appeared abnormal.

2.2. Exponential Distributed Model

The exponential distributed model is designed for the test period. The exponential distributed model is also called reliability growth model which includes fault counting model and failure interval time model. The fault probability distribution function (PDF) of the exponential distributed model is $f(t) = K(\lambda e^{-\lambda t})$, and the cumulative distribution function (CDF) is $F(t) = K(1 - e^{-\lambda t})$, where t represents time, k represents total number of faults and λ, a shape parameter, represents the probability of finding a defect. The exponential distributed model is the simplest but one of the most important models over the reliable models and it is the fundament of other growth models.

Jeinski-Moranda (J-M) model [15] is one of the earliest failure interval time model. The software failure rate function is $Z(t_i) = \phi[N - (i - 1)]$, where N represents initial number of defectives, Φ represents ratio constant. The presupposition of this model is 1) the possibility of each fault detected during the test period influence the failure are the same. 2) time for repairing the faults can be ignored.3) all faults can be repaired perfectly. Thus each repairmen of faults counts equally to improve the software's failure rate.

Littlewood model [16] and G-O model are exponential models.Littlewood model supposes that different faults influent failure differently and huge failures are avoided in the early phases, the average error scale will be smaller and smaller. Littlewood proposed the Littlewood nonhomogeneous Poisson proess model(LNHPP) [17] later on. Goel and Okumoto proposed a failure amount model in the test period (G-Omodel). They supposed the accumulated faultnumber was $N(t_i)$ in time t_i which is a nonhomogeneous Poisson process model [18].

2.3. Exponential Distributed Model

Yamada [19] and other people proposed a test period not only includes detecting the faults but also includes isolating faults. When there comes a fault, we need to find the reason why faults are disabled. Therefore, there exists a time delay for finding a fault until we report it. The accumulated time delayed faults accord with S curve distribution and this is called delayed S model which is also a reliable increment model. S model meets the requirement of nonhomogeneous Poisson process. Its CDF is $F(t) = K(1-(1+\lambda t)e^{-\lambda t})$, where t represents time; k represents total faultnumber; λ represents the possibility of detecting a faultt. The PDF is $f(t) = K\lambda^2 te^{-\lambda t}$.

In 1984, Oghba [20] proposed another S distributed model called transformed S model. This model considers the detected faults are interdependent and the more faults are detected the more faults will be detected later. The accumulated faultdistributed function (CDF) is $F(t) = K(1-e^{-\lambda t})/(1+\phi e^{-\lambda t})$, where t represents time; k represents total fault number; λ represents the possibility of detecting a defect. Its PDF is $f(t) = K\lambda e^{-\lambda t}K(1+\phi)/(1+\phi e^{-\lambda t})^2$ [21].

3. The Fault Prediction Model for Web applications during the test period

Each technic to predict software faults are not independent. This paper will combine G-O model [15] and Rayleigh model to develop a fault prediction model for Web applications [22].

Fault analysis can be used to understand the trend of development quality and faults analysis provides two important parameters. One is the trend of detected faults, the other is a trend of cumulative faults. Generally, detected fault curve will be on the rise in the early stages of the project, and to the middle and later periods of the project, the curve should decline overall. Finally, it tends to zero. We may ensure the development of the project through the continually observation of these curves. Also, through analyzing and predicting the time when the found fault tends to zero, we can set up the acceptance criterion and release time. Fault prediction model mainly represents the faults still in the software.

To develop the fault prediction model of Web applications, we analyzed a lot Web application's test data. By studying on the fault data, we found the fault data during the test period presented as exponential model similar to G-O model and Rayleigh model. So we simulated these fault data based on G-O model and Rayleigh model. This paper utilized G-O nonhomogeneous Poisson model and Rayleigh model, showed a fault prediction model during the test period of the project [23, [25].

G-O model supposed that:

1) Found fault number comply with Poisson distribution;
2) Found fault number at each unit interval is direct proportion to the remaining fault numbers in the software;
3) All faults are independent and all share equal possibility being detected;
4) All detected faults will be eliminated immediately and no new error will be involved during the eliminating period.

Thus, the accumulated fault function $B(t)$ is end by time t. If "a" represents the detected fault number at the end, we can conclude that:

$$B(0) = 0 \tag{1}$$

$$\lim_{t\to\infty} B(t) = a \tag{2}$$

Therefore, the number of faults in (t, t +Δt) is in direct proportion to remaining fault at time t, the ration constant is represented by " m", so the relationship is as Equation (3).

$$B(t+\Delta t) - B(t) = m \times (a - B(t)\Delta t \tag{3}$$

Suppose $\Delta t \to 0$, then $B'(t) = m \times (a - B(t))$, by using the boundary conditions we can obtain a model mean value function as in (4).

$$B(t) = a \times (1 - e^{-mt}) \quad (4)$$

This model is just like the exponential distributed model $F(t) = K(1 - e^{-\lambda t})$ during the test period, where t represents time; k represents total fault number, λ represents fault detected possibility [26]. Thus, we need to determine the parameter “a” and “m” in (4). After the parameters are confirmed, we can get the fault prediction model for Web applications.

3.1. Determination of the Total Number of Faults

To estimate the total number of faults in the model , we can refer to the Rayleigh model, Fault distribution density function of the Rayleigh model $f(t) = 2a(t/c^2)e^{-(t/c)^2}$, Cumulative distribution function of faults $B(t) = a(1 - e^{-(t/c)^2})$, where ”a” is the total faults, t stands for time, c is a constant, and $c = \sqrt{2}t_m$, t_m is the time when B (t) reaches the maximum. This model assumption is : When there are 39.35% faults, $f(t)$ reaches the maximum. Therefore, through the curve of faults having been found, we can determine the time t_m, then determine the accumulation fault number “b” of t_m moment, Eventually determine the total number of faults is $a \approx b / 39.35\%$.

3.2. Determination of the Probability of Finding a Fault

The probability of finding a fault is the faults per thousand lines of code number (faults/KLOC), can be represented as $m = \alpha \times \beta$, α is the test particle size, $\alpha = 1000 \times (N_k / N_c)$, where N_c is effective software lines of code and N_k is the number of test case. β is case intensity, $\beta = N_d / N_k$, where N_d is the number of software faults found in the testing process. The probability of finding a fault describes the ability of software testing adequacy. There are two methods to determine this parameter, one is using the nonlinear regression software, according to $B(t) = a \times (1 - e^{-mt})$ model ,matching the related model parameters “m”, the other is counting the test particle sizeα of similar project and case intensity β, Finally calculating the detected rate “m”.

3.3. The Fault Prediction Model for Web Application

This section confirms parameters of the model according to fault data of Web application project [27]. Table 1 tells the statistical data of found fault of a Web Application project. Using these data draws the found faults curve, as shown in Figure 1. Figure 2 describes cumulative faults. When Time tends to infinity, fault number can tend to 0.

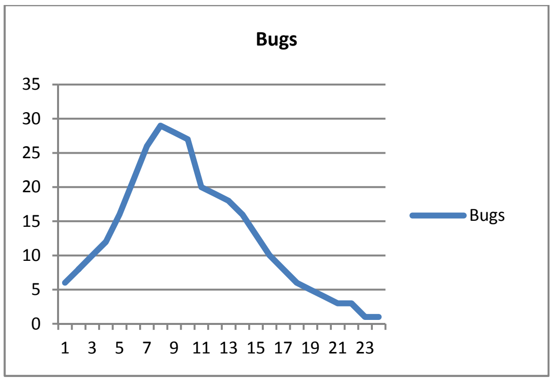

Figure 1. Detected Fault Distribution

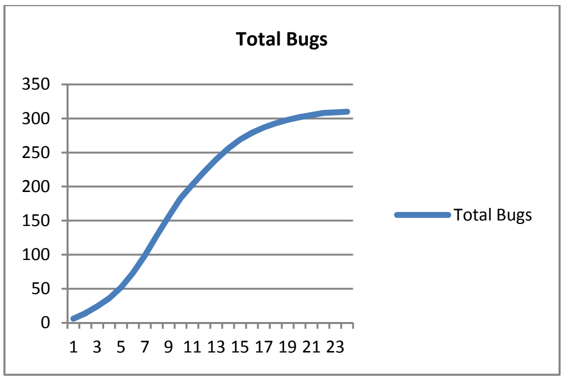

Figure 2. Cumulative Fault Distribution

Table 1. Project's Fault Data

Date	interval	Found faults	Number of faults
2013.6.1	1	6	6
2013.6.2	2	8	14
2013.6.3	3	10	24
2013.6.4	4	12	36
2013.6.5	5	16	52
2013.6.6	6	21	73
2013.6.7	7	26	99
2013.6.8	8	29	128
2013.6.9	9	28	156
2013.6.10	10	27	183
2013.6.11	11	20	203
2013.6.12	12	19	222
2013.6.13	13	18	240
2013.6.14	14	16	256
2013.6.15	15	13	269
2013.6.16	16	10	279
2013.6.17	17	8	287
2013.6.18	18	6	293
2013.6.19	19	5	298
2013.6.20	20	4	302
2013.6.21	21	3	305
2013.6.22	22	3	308
2013.6.23	23	1	309
2013.6.24	24	1	310

For the parameter of "m", we can use two methods to determine, then average it. One is using MATLAB to fit $\mathrm{B(t)} = 320(1 - e^{-mt})$ curve (Seeing Figure 2) by the data in Table 1 , then calculating parameter m=0.0326. Another method is to use the Equation of $\mathrm{m} = \alpha \times \beta$ to calculate the average value. Table 2 is the statistical data of 7 projects which represents particle size and intensity of test cases. So we can get the value of "m " using (5).

$$\mathrm{m} = \frac{1}{7}\sum(\alpha_i \times \beta_i) \tag{5}$$

In (5) α_i = Numbers of test case/Numbers of code line, β_i = Numbers of found defect/ Numbers of test case. Through the data in Table 2, m is approximately 0.0324. Finally get the mean m=0.0325.

Table 2. Statistics Data of Web Application

Project	Code line	Test case	Found faults
1	4125	132	110
2	4428	155	129
3	4724	165	151
4	10381	335	321
5	9982	298	291
6	2115	55	82
7	2392	67	96

So the fault prediction model for Web application can be approximately represented as in (6). B(t):

$$\mathrm{B(t)} = \mathrm{a}(1 - e^{-0.0325t}) \tag{6}$$

Where “a” is the total number of faults in the project, which is different to the different project. We define Quality index “q” as $e^{-0.0325t}$. The smaller Quality index is, the better the product quality is. Moreover, we can estimate how long the system test will last, as in (7):

$$T = -\frac{\ln q}{0.0325} \tag{7}$$

The total number of faults can be estimated by the peak value of found faults. For Figure. 1, peak point of the curve is 29 while “t” equals to 8. Therefore, the accumulation of found faults is 128 when “t” equals to 8, then we can estimate that the total number of faults is 128/0.4≈320. So, for this project , the fault prediction model during system test phase is as in (8).

$$B(t) = 320(1 - e^{-0.0325t}) \tag{8}$$

In fact, fault analysis chart can tell us a lot of valuable information. Such as whether the ratio of human resources on development and test is appropriate, and so on. For abnormal fluctuations,We should pay more attention to them and see what has happened. These fault analysis can help us to improve the development work.

4. In the Case of the Application

This section will verify the model, using $B(t) = a(1 - e^{-0.0325t})$ to predict the project faults during the test phase. Figure 3 shows the test data of first ten days of a project. According to the fault curve we can obtain the peak point of the curve is 13 while “t” equals to 8. Then the accumulation of found faults is 63 when “t” equals to 8. So we can estimate the total number of faults is 63/0.4≈158. Therefore, fault model of this project is $B(t) = 158(1 - e^{-0.0325t})$. Figure 4 shows the actual data of fault distribution. Figure 5 shows the difference between estimated data and actual data. Obviously, the two curves are basically identical.

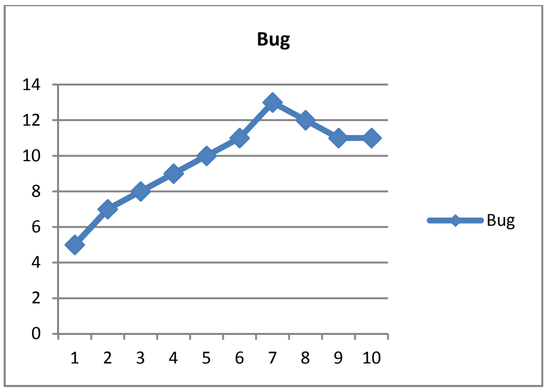

Figure 3. Data of the First Ten Days during Test Period

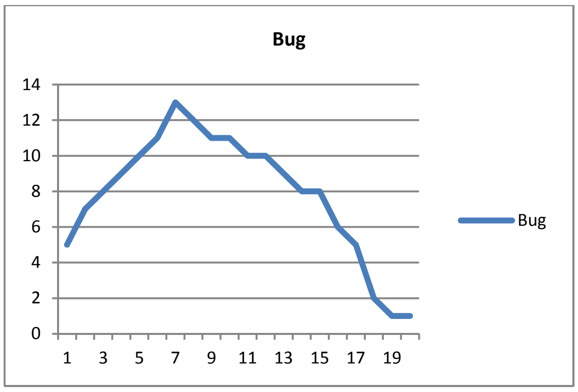

Figure 4. Actual Fault Data after Testing

If the quality index is 0.0023, the test period is approximately 20-day. And the actual test period is 19-day.

We can evaluate the accuracy of estimation by MRE (Magnitude of Relative Error). Namely, MRE=|Actual-Predicted|/Actual. The MMRE (Mean of the Magnitude of Relative Error) is also a useful evaluation tool, Its Equation is as in (9).

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{9}$$

For Figure 5, the MMRE is 9.1%. The calculated MMRE of similar projects is close to 9.9%.

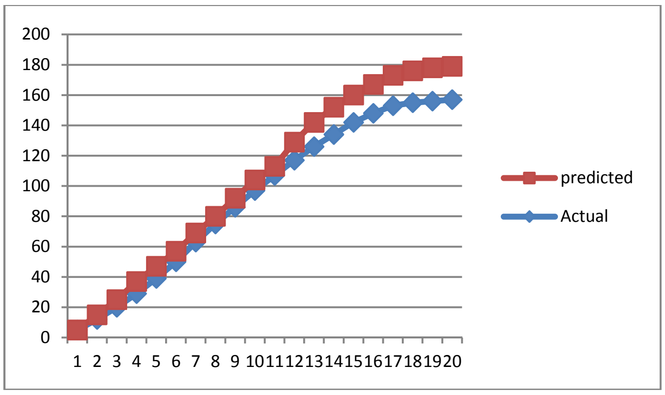

Figure 5. The comparison between the Predicted and Actual Data

5. Conclusion

The research of fault model could be a way to evaluate the product performance. Also, it could be used as a basis to product. It is meaningful to set up research and product model. This paper, through the G-O non-homogeneous Poisson process model and the Rayleigh model, presents the steps of fault prediction model for Web application software. Through this model, we can realize the quantitative management of the testing process, predict test efficiency, and predict the potential number of faults. With the development of software engineering technology, the fault prediction technology is also facing new challenges and it requires constant innovation to adapt to changing needs.

For future work, further investigation can be studied on the slope of the model, and the influence on the prediction results.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61170273).

References

[1] Akiyama F. *An example of software system debugging*. In: Proc. Of the Int'l Federation of Information Proc. Societies Congress. New York: Spring Science and Business Media. 1971: 353-359.

[2] Halstead MH. Elements of Software Science. *New York: Elsevier. North-Holland*. 1977.

[3] Lipow M. Number of faults per line of code. *IEEE Trans. On Software Engineering*.1982; 8(4): 437-439.

[4] Takahashi M, Kamayachi Y. An empirical study of a model for program error prediction. *IEEEE Trans. On Software Engineering*. 1989; 15(1): 82-86.

[5] Malayia Y, Denton J. *Module size distribution and defect density*. In: Proc. Of the 11th Int'd Symp on Software Reliability Engineering. New York: IEEE Computer Society Press. 2000: 62-71.

[6] Chulani S. Bayesian analysis software cost and quality models. Ph.D. Thesis. Los Angeles: *Univcrsity of Southern California*. 1999.

[7] Chulani S. *Results of Delphi for the defect introduction model, sub-model of the cost/quality model extension to COCOMO II.* Technical Report, USC-CSE-97-504. 1997.
[8] Chulani S, Boehm B. *Modeling software defect introduction and removal: COQUALMO (COnstructive QUALity MOdel).* Technical Report. USC-CSE-99-510. 1999.
[9] Chillarege R, Bhandari I, Jarik K, Michael J, Diane S, Bonnie K, Man-Yuen W. Orthogonal defect classification-a concept for in-process measurements. *IEEE Trans. on Software Engineering.* 1992; 18(11): 943-956.
[10] Mills H. On the statistical validation of computer programs. Technical Report, FSC-72-6015, IBM Federal Systems Division. 1972.
[11] Lyu M. Handbook of Software Reliability Engineering. Singapore: McGraw-Hill. 1996.
[12] Trachtenberg M. Discovcring how to ensure software reliability. *RCA Engineer*. 1982; 27(1): 53-57.
[13] Jr Gaffney JE. *On predicting software related perforance of large-scale systems*. Proc. of the Int'l Conf. of the Computerr Measurement Group, CMG XV. San Francisco. 1984.
[14] Putnarn LH, Myers W. Familiar metric management-reliability.1995.
[15] Jelinski Z, Moranda P. Software reliabiliy research. Freiberger W, ed. Statistical Computer Performance Evaluation. New York: Academic Press. 1972: 465-484.
[16] Littlwood B. Stochastic reliability growth: A model for fault removal in computer programs and hardware designs. *IEEE Trans. On Reliability*. 198l; R-30:313-320.
[17] [Abdel-Ghaly AA, Chan PY, Littlewood B. Evaluation of competing software rcliability predictions. *IEEE Trans. on Software. Engineering.* 1986; 12(9): 950-967.
[18] Goel AL, Okumoto K. A time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. on Reliability*. 1979; R-28(3): 206-211.
[19] J Yamada S, Ohba M, Osaki S. S-Shaped reliability growth modeling for software error detection. *IEEE Trans. on Reliability*. 1983; R-32(5): 475-478.
[20] Ohba M. Software reliability analysis models. *IBM Journal of Research and Development*. 1984; 28(4): 428-443.
[21] Liu HW, Yang XZ, Qu F, Zhao JH. Software reliability growth models of nonhomogenous Poisson process. *Journal of Tongji University (Natural Science)*. 2004; 32(8): 1071-1074 (in Chincse with English abstract).
[22] Mendes N, Mosley N, Counsell S. *Early Web size measures and effort prediction for Web costimation*. Proc. Of the 9th Int'l Software Metrics Symp. (METRICS 2003). 2003; 18-29.
[23] Chen HW, Wang J, Dong W. High confidence software engineering technologies. *Acta Electronica Sinica.* 2003; 31(12A): 1933-1938 (in Chinese with English abstract).
[24] GUO SH, Lan YG, Jin MZ. Some issues about trusted components research. *Computer Science*. 2007; 34(5): 243-246 (in Chinese with English abstract).
[25] Boehm B. Value-Based Software engineering-overview and agenda. Technical Report, USC-CSE 2005-504. 2005.
[26] Xin Xia, Shu-xin Zhu. A Survey on Weighted Network Measurement and Modeling. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(1): 181-186.
[27] Jaralikar SM, Aruna M. Case study of a hybrid (wind and solar) power plant. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2011; 9(1): 19-26.