Prediction Model Based Failure Time Data for Software Reliability

Peng Lin[1,2](✉), Xu Tian[1], Xiaojuan Wang[1], Xu Cao[1], Jiejing Cao[1], Jianli Li[1], and Yan Gong[1]

[1] Software Testing and Evaluation Centre, China Electronic Equipment of System Engineering Institute, Beijing 100141, China
paul-lim@163.com

[2] Academy of Military Sciences PLA China, Beijing 100091, China

Abstract. Since all the defects cannot be detected within a finite software testing process (STP), the failure data should be wisely used to estimate the potential defects for software reliability. Therefore, a standard graphical methodology (GM) model is proposed for software reliability, in which failure data of time domain is utilized to predict the potential defects. First, non homogeneous and compound Poisson process is involved to model the failure time during STP. Then, GM model is utilized to predict the potential defects. Further, the software reliability is estimated based on GM model. Finally, compared with the traditional models, GM model can reach an improvement of 30% relative gain on average.

Keywords: Defect prediction · Poisson process · Data fitting
Software reliability

1 Introduction

As the development of software industry, the scale of software can be increasing day by day. In a typical system environment of software development, the software is programmed by a development term. Once a software release is generated, it should be assigned to a testing term, which should verify the release meets the design specifications restrictly. A software release can be a small program, or a large integrated system consisted by a group of subsystems. The testing term tests the software by applying necessary operation to detect the potential defects, which can find the failures of the software against to the design specifications. Generally, the testing activity is limited by a pre-determined time and a pre-determined number of testers. However, it is not feasible to discover every potential defects in a finite software testing process (STP). Therefore, the software reliability attracts more and more attentions [1–4].

Software reliability can be defined as the probability that a software operates without any failure within a specified time by specified operating. Software reliability can be obtained from the testing progress during the software development

G. Sun and S. Liu (Eds.): ADHIP 2017, LNICST 219, pp. 265–274, 2018.
https://doi.org/10.1007/978-3-319-73317-3_32

[1]. When one defect is detected and repaired during STP, the potential defects in the code will decrease, which means the defect detection rate is against to the number of detected defects [2] as show in Fig. 1.

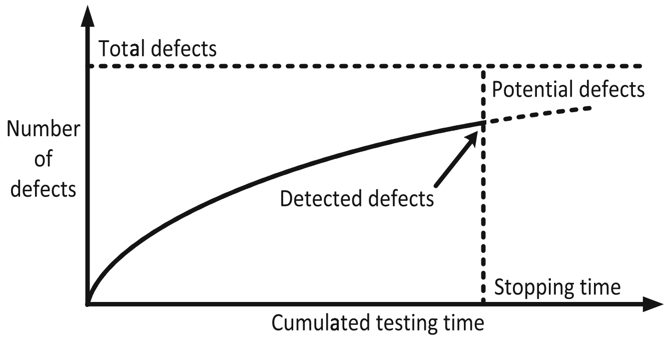

Fig. 1. Defect detection in software testing process.

Based on the detection defects, we can determine whether the program can be released, or more testing time is necessary. The number of failures can be used to estimate the software, which should be reported to our customers before operating the software [3]. The estimation can help us to evaluate the necessary time for software testing.

There are always some constraints limit the software testing time, especially, the constraint of the time-consuming to release a software system under business pressure [4]. Therefore, software reliability modeling is proposed to evaluate both the current and future reliability. In particular, defect prediction can help determine when the software development should be terminated.

Traditional software reliability models can be grouped into three classes.

First, the amount of defects is finite within a finite amount of code [5], by which more code means more defects, so as the defect detection rate too. The amount of detection defects during STP are explicitly made in some models, while others assume that these defects data can be fitted statistically by the software reliability growth model.

Second, a compound Poisson process (CPP) [6, 7] is used to model the clusters of detected defects, where the defects are grouped in a given time interval. CPP not only try to predict the remaining failure rate after a limited testing time, but also can forecast the potential detects after a given STP.

Third, the number of defects is infinite in log Poisson model assumption [8]. Further, Log Poisson models make assumptions that all the detected defects are perfectly repaired. However, it is difficult to evaluate the affection of the model assumptions.

Although infinite assumption is questionable, it is reasonable that the potential defects will exist until be found and repaired [9].

Focus on large-scale software or integrated system within STP, a defect prediction model basing on standard graphical methodology is proposed in this paper, where the defect data of time domain is utilized. Three actual testing data sets of time domain are used to compare GM model with the classical G-O, CPP and M-O models, both the fitting and prediction results of GM model perform better than the others.

2 Poisson Models

2.1 Non Homogeneous Poisson Process

Non homogeneous Poisson process (NHPP) has been proposed as software reliability model [9,10]. The Poisson parameter $\lambda(t)$ (where $t > 0$) in NHPP model is a time dependent function.

Defects of the software cause failures at random times. Assume $N(t)$ is the amount of detected defects within time t, which should be a cumulated testing time (either CPU time or calendar time). Then, let $\lambda(t)$ be the probability of the amount of detected defects within a time interval t. Further, the probability of n failures can be calculated as

$$P\{N(t) = n\} = \frac{\lambda^n(t)}{n!} e^{(-\lambda(t))}. \tag{1}$$

Therefore, $\lambda(t)$ should be the key function in NHPP. There are two kind of NHPP based models, which are G-O model and M-O model [5].

In G-O model [9], let $\lambda(t)$ be the expected amount of total defects, which are detected within time t can be calculated as

$$\lambda(t) = \alpha(1 - e^{(-\beta t)}), \tag{2}$$

where β is the detection rate for each individual defect. The parameter α is the expected amount of total defects in G-O model, and

$$\lambda(t) = \alpha P(t), \tag{3}$$

where $P(t)$ is the cumulative distribution function. Then, $P(0) = 0$, which means that no defect is detected before the software testing process (STP) starts. On the other side, $P(\infty) = 1$, then, $\lambda(\infty) = \alpha$, and α is the amount of total defects, which should be detected after a finite STP.

M-O model [9] assumes that one software may have an infinite number of defects. M-O is a log Poisson model, where the detected number of defects after cumulated testing time t is NHPP. The $\lambda(t)$ in M-O model can be calculated as

$$\lambda(t) = \frac{1}{\theta} \log(\lambda_0 \theta t + 1). \tag{4}$$

Both the operational performance of G-O and M-O depends on the lasting time of STP. Longer STP leads to better performance. Although the time spent in STP delays the product release, which leads to additional costs, the cost of repairing a defect after software release should be more expensive than during STP [7]. Therefore, the defect prediction can be optimal release time to minimize cost by determining STP time.

2.2 Compound Poisson Process

The failures of CPP model are detected and grouped in clusters [11], which are found following a Poisson process. Further, a compounding distribution is used to model the clusters size, which follows a geometric distribution [12]. Further, the amount of detected defects follows a compound Poisson process, and the probability function is given by

$$P\{(N(t) = n)\} = \sum_{k=1}^{m} \frac{(\lambda t)^k}{k!} e^{(-\lambda t)} f^{*k}\{X_1 + X_2 + \cdots + X_k = n\}, \tag{5}$$

where, $f^{*k}\{X_1 + X_2 + \cdots + X_k = n\}$ is the sum of k independent identically distributed random variables X_i, which follows a distribution function $f(X)$. Since the distribution function $f(X)$ models the size of failures cluster, the mean value can be given by

$$E[N(t)] = \lambda t E[X]. \tag{6}$$

CPP model is easier to implement, which can adaptively change to fit different projects. The failure rate of CPP model is constant. However, as the failure rate will update time to time, the CPP model can not predict a long time period. Then, the predicted failure rate of CPP is a constant, while it is dependent on time in NHPP model.

3 Prediction Model

A standard graphical methodology (GM) [13] is proposed for defect prediction. There are three steps to realize defect prediction. First, the data of the cumulated failure times should be ascending ordered. Further, the two necessary parameters of the theoretical exponential distribution should be estimated too. Finally, a defect prediction basing on the theoretical distribution can be obtained.

The cumulated failure times are ascending ordered, then, associate a probability with ascending ordered failure time t_i by

$$p_i = \frac{(i - \frac{1}{2})}{n}, \tag{7}$$

and further associate with the points

$$z_i = (t_i, p_i), i = 1, \cdots, n. \tag{8}$$

The cumulative distribution function [14] for the two-parameter exponential distribution can be

$$F(t) = 1 - e^{-\frac{t-\mu}{\lambda}}, \tag{9}$$

where λ is the respected mean, and μ is the respected shift. Then, the $\lambda(t)$ in GM model can be given by

$$\lambda(t) = \alpha e^{-\frac{t-\mu}{\lambda}}. \tag{10}$$

Further, let $Q(p_i)$ be the theoretical distribution and given by

$$F(Q(p_i)) = p_i. \tag{11}$$

And then,

$$Q(p_i) = F^{-1}(p_i), \tag{12}$$

for the two-parameter exponential distribution,

$$Q(p_i) = -\lambda ln(1 - p_i) + \mu. \tag{13}$$

4 Software Reliability

A software reliability model is defined by NHPP [1,2], where $N(t)$ represents the amount of detected defects by cumulated testing time t during STP. Further, the failure intensity function of the software reliability model is defined by

$$F(t) = \frac{d\lambda(t)}{dt}. \tag{14}$$

The probability of the detected defects $N(t)$ has the value n is given by

$$P\{N(t) = n\} = \prod_i P\{N(t_i) = i\}, i = 1, \cdots, n. \tag{15}$$

Further, the reliability of t_i based on the last failure time t_{i-1} can be obtained as

$$\begin{aligned} & P\{N(t_i)|N(t_{i-1})\} \\ & = P\{N(t_i) > i - 1|N(t_{i-1}) = i - 1\} \\ & = 1 - P\{N(t_i) \leq i|N(t_{i-1}) = i - 1\}. \end{aligned} \tag{16}$$

5 Comparison

5.1 Data Sets

Three are three classical models selected to compare with GM model, including G-O, CPP and M-O model. Three actual testing data sets [3,5,15] of time domain are used to estimate the performance of the compared models.

Each data set is sorted and separated into the fitting part and the estimation part. The fitting part data takes a part of 90% percent in the data set, which is the lower part of time domain. The estimation part is about 10% of the data set, which is the higher part of the time domain. The fitting part data is used to compare fitting performance. The estimation part data is used to compare the prediction performance.

The first data set contains 17 defects with cumulated failure time [15], and then, the fitting part has 15 defects and the estimation part has 2 defects. The second data set contains 30 defects [5], in which the fitting part has 27 defects and the estimation part has 3 defects. There are 136 defects in the third data set [3], where the fitting part has the lower 122 defects of time domain and the estimation part has 14 defects.

Therefore, the first data set is the smallest one in the three actual data sets, the second data set is the middle one, and the third data set is the largest one.

5.2 Performance Estimation

The four referenced models are compared on the three actual data sets, and the result of the first data set is shown in Fig. 2.

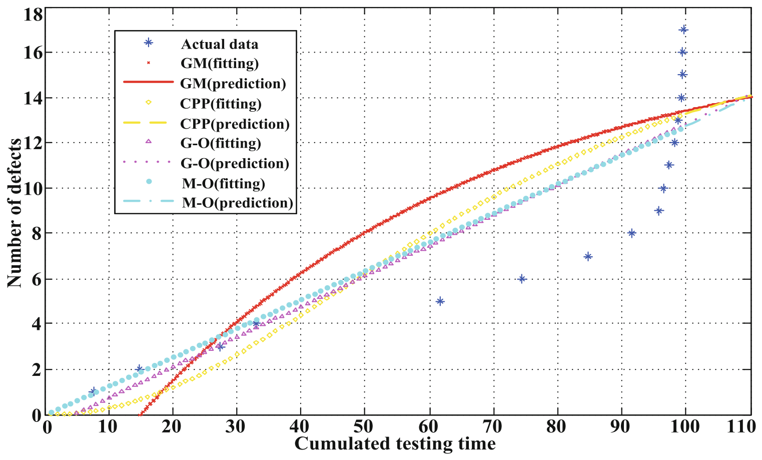

Fig. 2. Performance on the first data set

Further, the second result of the four models is shown in Fig. 3.

At last, the third result of the four models is shown in Fig. 4.

Each fitting curve are separated into two parts. The left part is the fitting result marked with fitting, and the right part is the prediction result marked with prediction.

The mean square error (MSE) is used to compare the performance of the four models, the MSE can be calculated as

$$MSE = \sqrt{\frac{\sum_{i=1}^{n} (e_{t(i)} - p_{t(i)})^2}{n}}, \tag{17}$$

where $e_{t(i)}$ represents the estimated amount of expected defects at time $t(i)$, which can be a fitting or prediction result. $p_{t(i)}$ represents the real amount of detected defects at time $t(i)$, which can be picked up from the actual testing data sets of time domain.

The results of fitting and prediction performance on the three data sets are demonstrated in Table 1.

On the first data set, the prediction MSE of GM model is the smallest. Although GM model performs the best on the prediction MSE, both the fitting and average MSE are bigger than the other models. Therefore, the average MSE of GM model performs worst on the smallest data set.

Further, the average MSE of GM model is 1.397 on the second data set, which is a little bigger than CPP model, but smaller than the G-O and M-O

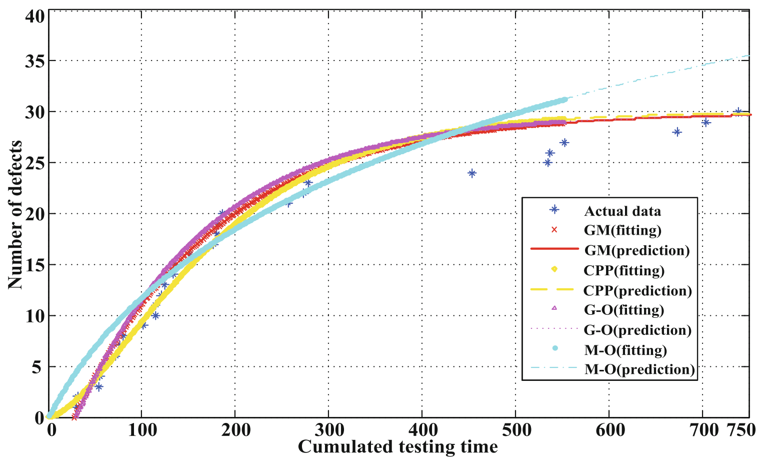

Fig. 3. Performance on the second data set

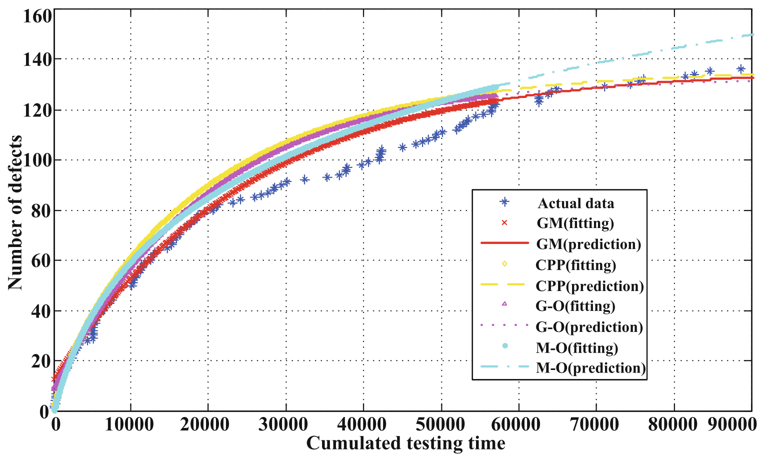

Fig. 4. Performance on the third data set.

model. The third data set is the largest one, the average MSE of GM model is just about 3.950, which is much smaller than the others.

At last, the average MSE of GM model performs the best on the three data sets in the compared models. Since the defect number of the third data set is larger than both the first and second ones, the larger data set could lead to more accuracy. With the number of defects increasing, the performance of GM model should be always better than the other models.

Table 1. Performance results

Data set		G-O	CPP	M-O	GM
1	Fitting MSE	2.210	2.679	2.227	3.211
	Prediction MSE	3.775	3.303	3.841	3.158
	Average MSE	2.993	2.991	3.034	3.185
2	Fitting MSE	2.047	1.668	2.815	1.738
	Prediction MSE	0.934	1.111	5.715	0.957
	Average MSE	1.491	1.389	4.265	1.397
3	Fitting MSE	8.738	10.562	8.532	5.993
	Prediction MSE	2.684	2.881	10.168	1.907
	Average MSE	5.711	6.721	9.350	3.950

5.3 Reliability Estimation

To further estimate the defect prediction, the reliability is compared by the last data of both fitting part and estimation part. The last data of both the fitting part $e_{0.9N}$ and prediction part $e_{1.0N}$ are picked up, and then, the MSE is used to compare the prediction accuracy and given by

$$P_{MSE} = \sqrt{\frac{(\frac{e_{0.9N}}{N} - 0.9)^2 + (\frac{e_{1.0N}}{N} - 1.0)^2}{2}}, \tag{18}$$

where N is the size of the data set.

The reliability results of the four models on the three actual data sets are shown in Table 2.

Table 2. Reliability results

Data set		G-O	CPP	M-O	GM
1	Fitting $P_{0.9N}$	0.746	0.775	0.742	0.786
	Prediction $P_{1.0N}$	0.754	0.781	0.750	0.787
	P_{MSE}	0.205	0.178	0.209	0.171
2	Fitting $P_{0.9N}$	0.968	0.976	1.043	0.965
	Prediction $P_{1.0N}$	0.986	0.995	1.179	0.989
	P_{MSE}	0.049	0.054	0.162	0.047
3	Fitting $P_{0.9N}$	0.922	0.935	0.950	0.908
	Prediction $P_{1.0N}$	0.964	0.983	1.095	0.974
	P_{MSE}	0.030	0.027	0.076	0.019

On the first data set, both $P_{0.9N}$ and $P_{1.0N}$ of GM model are bigger than the other three models, which means bad performance on the smallest data set.

However, both on the second and third data sets $P_{0.9N}$ and $P_{1.0N}$ of GM model are the smallest ones. Therefore, GM model perform better both on the middle and largest data sets than the three compared models.

Further, P_{MSE} is 0.171, 0.047 and 0.019 respectively. Therefore, P_{MSE} of GM model is the smallest in the compared models, which means the best fitting accuracy at the lasting data of time domain.

Since the GM model performs better both in the performance and reliability results, the relative gain G is formulated as

$$G = \sum_{*=G-O}^{CPP,M-O} \frac{P_{MSE}^{*} - P_{MSE}^{GM}}{P_{MSE}^{*}}. \tag{19}$$

The relative gain of GM model on the first data set is 13.15%, 29.90% and 47.10% on the second data set. Then, an average relative gain 30.05% can be reached by GM model, which shows an improvement compared to the classical G-O, CPP and M-O models.

6 Conclusion

A standard GM model is proposed to predict potential defects for software reliability in this paper. By fitting defect data of time domain, both the number of defects and failure rate can be estimated by GM model theoretically. Further, GM model is used to estimate the software reliability during STP, which can help to determine when to terminate the STP and release the software. Finally, three traditional models are compared with GM model on the three actual testing data sets. The performance comparison shows that GM model performs much better than the others, and an average 30% relative gain can be obtained for software reliability estimation.

References

1. Wood, A.: Software reliability growth models: assumptions vs. reality. In: International Symposium on Software Reliability Engineering, pp. 136–141 (1997)
2. Almering, V., von Genuchten, M., Cloudt, G., Sonnemans, P.J.M.: Using software reliability growth models in practice. IEEE Softw. **24**, 82–88 (2007)
3. Musa, J.D., Iannino, A., Okumoto, K.: Software Reliability: Measurement, Prediction, Application. McGraw Hill, New York (1987)
4. Sahinoglu, M., Glover, S.: Economic analysis of a stopping-rule in branch coverage testing. In: International Symposium on Quality Electronic Design, pp. 341–346 (2002)
5. Akuno, A.O., Orawo, L.A., Islam, A.S.: One-sample Bayesian predictive analyses for an exponential non homogeneous Poisson process in software reliability. Open J. Stat. **4**, 402–411 (2014)
6. Sahinoglu, M.: Compound-Poisson software reliability model. IEEE Trans. Softw. Eng. **18**(7), 624–630 (1992)

7. Xianghui, Z., Lin, L., Yafang, H., Lei, Z., Yuangang, Y.: Software reliability measurements based on compound Poisson processes. J. Tsinghua Univ. Sci. Technol. **53**(12), 1743–1749 (2013)
8. Musa, J.D., Okumoto, K.: A logarithmic Poisson execution time model for software reliability measurement. In: International Conference on Software Engineering, pp. 230–238 (1984)
9. Barraza, N.R.: Compound and non homogeneous Poisson software reliability models. In: ASSE 2010–11th Argentine Symposium on Software Engineering, pp. 461–472 (1984)
10. Chang, Y.: An alternative reliability evaluation of non homogeneous Poisson process models for software reliability. Int. J. Qual. Reliab. Manag. **17**(7), 800–811 (2013)
11. Sahinoglu, M., Can, U.: Alternative parameter estimation methods for the compound Poisson software reliability model with clustered failure data. Softw. Test. Verif. Reliab. **7**(1), 35–57 (1997)
12. Barraza, N.R.: Parameter estimation for the compound Poisson software reliability model. Int. J. Softw. Eng. Appl. **7**(1), 137–148 (2013)
13. Chambers, J.M.: Graphical methods for data analysis. Biometrics **40**(2), 493–499 (1983)
14. Aiex, R.M., Resende, M.G., Ribeiro, C.C., et al.: Probability distribution of solution time in GRASP: an experimental investigation. J. Heuristics **8**(3), 343–373 (2002)
15. Lohmor, S., Sagar, B.B.: Overview: software reliability growth models. Int. J. Comput. Sci. Inf. Technol. **5**(4), 5545–5547 (2014)