Simulation of Software Reliability Growth Model Based on Fault Severity and Imperfect Debugging

Xuejie Sun(✉) and Jiwei Li

School of Information Science and Engineering, Ocean University of China, Songling Road No. 238, Qingdao, China
sunxuejie@stu.ouc.edu.cn

Abstract. The existing software reliability growth model (SRGMs) usually assumes that the detected faults can be eliminated well when considering different types of software faults, to simplify the problem. Therefore, given these existing defects, we propose a new non-homogeneous Poisson process (NHPP) SRGM based on considering different fault severity. According to the complexity of the fault, we define the software fault as three levels: Level I is a simple fault, Level II is a general fault, and Level III is a severe fault. In the process of fault detection, the model comprehensively considers the tester's ability to find problems and the number of remaining issues. In the process of debugging, the problems of imperfection and new fault introduction are considered. Two kinds of real data sets, fault classification and non-classification, were selected and we made simulation for the proposed model and other traditional SRGMs on the PyCharm platform. The experimental results show that the software reliability model considering fault severity has excellent performance of fault fitting and prediction on both types of data sets.

Keywords: Fault severity · Non-homogeneous Poisson process · Software reliability growth models

1 Introduction

With the comprehensive application of computer software technology in various systems such as daily life and safety-critical applications, software quality is an essential guarantee for software survival, and software reliability is an important index to measure software quality [1–3]. In the process of software development, it is necessary to consider the time of software release, the number of failures after software release, and the severity of the failures. Therefore, as an important means of quantitative evaluation and prediction of software reliability, in recent years, many software reliability growth models (SRGMs) based on time-domain have been proposed and successfully applied to the development process of various types of security-critical software. Among all SRGMs, the non-homogeneous Poisson process (NHPP) class SRGM is recognized as

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 370, pp. 140–152, 2021.
https://doi.org/10.1007/978-3-030-72795-6_12

the most effective and widely used model because of its excellent characteristics such as easy to understand and easy to use.

In 1979, Goel and Okumoto [4] first used NHPP to describe the SRGM, known as the G-O model. The model assumes that the failure detection rate function is constant. Many models are based on the G-O model to improve some assumptions and modify the G-O model to make the newly established model achieve a better fitting effect. Yamada [5] proposed the Delay S-Shaped model, which believed that the failure detection rate was a non-decreasing function changing with time. Meanwhile, Ohba [6, 7] considers that there is more than one fault in the software. Kapur et al. [8] introduced the concept of Fault Severity Factor (FSF). They proposed a SRGM with two types of fault. The first type is the model proposed by Goel and Okumoto. The second type introduced the logistic rate during the removal process. Later many materials indicate that there is more than one level of software failure in the software system. However, in the models that have been proposed, it is usually assumed that the faults detected are immediately eliminated to simplify the calculations. In other words, they assume that the troubleshooting process is perfect. This assumption ignores the fact that all detected faults cannot be eliminated due to resource constraints and the introduction of new faults in the troubleshooting process. Therefore, the models can not be well applied to a practical application environment. The fitting ability and prediction ability of the model needs to be improved.

In this paper, we show how to classify software failures into three categories: severe, general, and simple. According to the severity of the software fault, we proposed a new SRGM to quantify the software level. In the process of fault detection, for simple failure, in different time intervals, the probability of testers finding problems is only related to the number of residual failures. Therefore, unlike Kapur et al., we assume that the fault detection rate of this fault level is a decreasing function over time. In the other two more complex fault levels, we will consider the ability of testers to detect problems and the number of remaining issues. Similarly, during the troubleshooting process, we introduced the non-removable software failure rate into consideration of the possibility of imperfect debugging. We entered the failure lead-in rate parameter into account of the option of adding new errors. Based on the above analysis, we carried out experiments on two types of failure data and compared the experimental results with the existing SRGMs. From the results, we can see that the new model has a better fitting and prediction effect.

The rest of the paper is arranged as follows. In the second part, three models of different severity fault of software reliability are given. This section also describes the proposed new model. The third part presents the evaluation criteria of the model. Part four discusses estimating model parameters using the Least Square Method (LSE) [4, 25] and applying these models to two failure data sets. Finally, in the fifth part, the experimental results are compared with other classical SRGMs.

2 The Fault Levels Model

We improved the assumptions of the earliest NHPP class model and obtained the following assumptions [9–11].

1. The software test runs in the same way as the actual running profile.
2. The different types of software faults are mutually independent.
3. Assume that $m(t)$ is the mean value function (MVF) of the expected number of problems detected in time $(0, t)$. The cumulative errors to time t follow the Poisson process where the MVF is $m(t)$. We can get the predicted function $m(t)$ of the increasing error number is a bounded non-subtractive function that satisfies the requirement that $m(0) = 0$.
4. The expected number of errors at any time interval $(t, t + \Delta t)$ is proportional to the number of errors remaining at time t. The ratio is the failure detection rate $b(t)$. The function of the failure detection rate over time for different problem levels is assumed as follows:

    1) For Level I, in different time intervals, the probability of testers finding minor problems is only related to the number of residual failures, and the failure detection rate function is $b_1(t)$.
    2) For Level II, the probability of the tester finding non-minor problems is related not only to the number of residual failures, but also to the tester's learning ability. The failure detection rate function is $b_2(t)$.
    3) For Level III, similarly, we can assume the fault detection rate function of Level III is the same as that of Level II, which is $b_3(t)$.

5. We assume that the original fault content in the software is N. N_1, N_2 and N_3 represent the initial fault number of simple, general, and severe levels respectively. Eliminating errors isn't all perfect for problems of different grades. Thus, we introduce the failure introduction rate a.

    1) For Level I and Level II, when the fault level is low, the developer does not introduce new errors in the debugging process, the failure introduction rate are a_1 and a_2, and $a_1 = a_2 = 0$.
    2) For Level III, at this point, developers may introduce new problems when solving the problem, so assume that the failure introduction rate is a_3.

6. In practice, due to limited test resources, the skill and experience of the tester, and different severity of fault, it is not possible to eliminate all detected faults in the test phase. Therefore, we introduce the non-removable failure rate c.

    1) For Level I and Level II, when the fault is relatively simple, we assume that the fault can be completely removed, so the non-removable failure rate is $c_1 = c_2 = 0$.
    2) For Level III, there are some software glitches that the software development team can not eliminate, so we assume that the non-removable failure rate is c_3.

Based on the above assumptions, we can construct the model as follows, where, the $m(t)$ of Level I, II, III are respectively expressed as $m_1(t)$, $m_2(t)$, $m_3(t)$.

2.1 Level I SGRM

According to the above assumptions, and from [12, 13], the model of the simple problem can be expressed as

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b_1(t) \times [N - m_1(t)] \tag{1}$$

Since the failure detection rate of minor problems by testers is only related to the number of remaining faults, with the continuous correction of the failures, the number of residual failures in the software become less and less, and the probability of detection becomes lower and lower. Therefore, it is assumed that the function of the failure detection rate over time of the tester for minor problems satisfies the following equation.

$$b_1(t) = \frac{b_1}{1+t}(0 \le b_1 \le 1) \tag{2}$$

In the formula, b_1 denotes a fault detection rate of simple faults found by the tester at the initial time.

Substitute (2) into (1), and solving (1) under the condition $m_1(0) = 0$, we can get the MVF of Level I as follows

$$m_1(t) = N[1 - (1+t)^{-b_1}] \tag{3}$$

2.2 Level II SGRM

Similarly, according to the above assumptions, the model of the middle problem can be formulated as

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b_2(t) \times [N - m_2(t)] \tag{4}$$

The logistic Testing-Effort Function (TEF) [14–18] can well describe extensive test work, we use the ratio of test coverage growth rate and uncovered code to express the failure detection rate of the Level II fault.

The logistic TEF formula for the period $(0, t]$ is

$$W(t) = \frac{W_{\max}}{1 + A\exp(-\alpha t)} \tag{5}$$

Where A is a constant, α is the consumption rate of testing effort, and $W_{\max}$ is the total testing effort that can be consumed finally. The current TEF rate at test time t can be shown as

$$W'(t) = \frac{W_{\max} A\alpha \exp(-\alpha t)}{[1 + A\exp(-\alpha t)]^2} \tag{6}$$

For simplicity of calculation, we assume $W_{\max} = 1$. The function of the failure detection rate over time can be represented as

$$b_2(t) = \frac{W'(t)}{1 - W(t)} = \frac{\alpha}{1 + A\exp(-\alpha t)} \tag{7}$$

With the tester's continuous understanding of the software under test, the tester can write better test cases, and the failure detection rate will increase; at the same time, as the failure is constantly corrected, the remaining failure in the software is less and less, and the probability of detection is lower and lower. Therefore, the fault detection rate at this time is affected by these two aspects.

Substitute (7) into (4), and solving (4) under the boundary condition $m_2(0) = 0$, we can obtain the MVF of Level II as follow

$$m_2(t) = \frac{N[\exp(\alpha t) - 1]}{A + \exp(\alpha t)} \tag{8}$$

2.3 Level III SGRM

According to the above assumptions, the model of the complex problem can be shown as

$$\frac{dm_3(t)}{dt} = b_3(t) \times [N(1 + a_3 t) - m_3(t)] - c_3 m_3(t) \tag{9}$$

According to assumptions 2, the parameters of $b_2(t)$ and $b_3(t)$ should be different. Therefore, the function of the failure detection rate over time of Level III can be formulated as

$$b_3(t) = \frac{\alpha_1}{1 + A_1\exp(-\alpha_1 t)} \tag{10}$$

Substitute (10) into (9), and solving (9) with the condition $m_3(0) = 0$, the MVF of Level III is

$$\begin{aligned} m_3(t) =& \frac{N\alpha_1}{[\exp(\alpha_1 t) + A_1](\alpha_1 + c_3)^2\exp(c_3 t)} \times \\ & [(1 + a_3 t)(\alpha_1 + c_3)\exp(\alpha_1 t + c_3 t) - a_3\exp(\alpha_1 t + c_3 t) + a_3 - \alpha_1 - c_3] \end{aligned} \tag{11}$$

Therefore, we can assume different parameters in front of the formulas of varying Level and get the new SGRM as follow

$$m(t) = \sum_{i=1}^{k=3} p_i m_i(t) \tag{12}$$

We call it a fault levels model. In Eq. (12), p_1, p_2 and p_3 need to satisfy $\sum_{i=1}^{k=3} p_i = 1$. And the initial fault number satisfies $p_i N = N_i (i \in 1, 2, 3)$.

3 Model Comparison Criteria

We analyze models based on the ability to fit software failures and the ability to predict future software behavior based on observed failure data sets. The four standards of model comparison are:

3.1 The Fitting Effect Criterion

To quantitatively compare the effects of model fitting data, we use the Sum of Squared Errors (SSE), the Mean Square of Fitting Errors (MSE), and the R-square (R) [19–22].

MSE. The MSE formula is shown below

$$MSE = \frac{\sum_{i=1}^{k} (m(t_i) - m_i)^2}{k} \tag{13}$$

The smaller the value of MSE, the lower the fitting error, and the better the performance.

SSE. The calculation formula of SSE is as follows

$$SSE = \sum_{i=1}^{k} (m(t_i) - m_i)^2 \tag{14}$$

Similarly, the smaller the value of SSE is, the lower the fitting error is, that is, the better the performance is.

R. The formula for R is

$$R = 1 - \sum_{i=1}^{k} (m(t_i) - m_i)^2 \Big/ \sum_{i=1}^{k} (m_i - m_{ave})^2 \tag{15}$$

Unlike the above, the closer the value of R is to one, the better the fitting effect will be.

3.2 The Predictive Goodness Criterion

The ability of a model to predict failure behavior based on the current number of failures is called predictive validity. Musa [8, 23, 24] proposed a method that could be used to calculate the Relative Error (RE) of the data set to represent the predictive validity.

$$RE = \frac{\hat{m}(t_q) - q}{q} \tag{16}$$

First, assuming that q faults are found at the end of the test time t_q, we use the failure data before $t_e (t_e \leq t_q)$ to predict the parameters of $m(t)$. By substituting the values of these prediction parameters into MVF, we can obtain the number of failures $m(t_q)$ over time t_q. The second step is to compare the predicted value with the actual amount q. Third, repeat the process for different t_e values. The validity of the prediction can be verified by drawing the relative errors of different t_q values. The closer the number is to zero, the better the prediction. Where the positive error represents an overestimate; A negative error indicates underestimation.

4 Model Simulation and Result Analysis

In this section, the proposed model has been tested on two real data sets to evaluate its validity. At the same time, the model with better performance on each data set and the classical models are used as the comparison model. In this paper, we use PyCharm software as a simulation platform. The Least Square Estimation (LSE) method is used to estimate the model parameters [4, 25, 26], and the estimation results generated by LSE are unbiased.

4.1 Data Set I

The data is from Misra, which is the failure data of software developed in the contract between IBM's Federal Systems Division and NASA's Johnson [27, 28]. The software was tested for 38 weeks, during which 2456.4 computer hours were used, and 231 faults were removed. It can be seen that faults are classified when failure data is recorded. The proposed model has been compared with the model proposed by Kapur et al. [29, 30], who also used the data set for experiments.

Analysis of Fitting Results. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 1. It is observed that the proposed model has the smallest value of SSE and MSE when compared with the SDE model. The two models have the same value of R. From the weight coefficient values of the proposed model, we found that Level I and Level II faults account for a significant proportion of the DS-I, while Level III accounts for a small percentage. The SDE model also reflects this phenomenon. The fitting results of the two models are close to the original data set, which further proves the validity of the model. Compared with the SDE model, the total number of faults fitted by the proposed model is closer to the total number of faults in the original data set. Figure 1 describes the comparison between the fitting value $m(t_i)$ of each failure data in the DS-I by the two models and the actual observed failure data m_i. It can be seen from Fig. 1 that the fitting results of the two models basically coincide with the real data. Combined with Table 1 and Fig. 1, the proposed model performs better in DS-I fitting.

Analysis of Prediction Results. We train with the failure data of the first 22 weeks, and compare the predicted value with the real cost to get the RE curves in Fig. 2. It can find that the RE value of the proposed model is the closest to zero as a whole when compared with the SDE model. It means that the proposed model predicts more accurately.

Table 1. Fitting parameters for DS-I.

Models under comparisons	Parameter estimation	MSE	SSE	R
Stochastic differential equation-based (SDE) model	$b_1 = .059, b_2 = .104, b_3 = .378, \beta = 66.593$ $p_1 = .64, p_2 = .342, p_3 = .018, a = 420$ $\sigma_1 = .048, \sigma_2 = .185, \sigma_3 = .599$	7.22	274.35	.999
Proposed SRGM (fault level (FL) model)	$p_1 = .711, p_2 = .268, p_3 = .021$ $N = 382, A = 12.01, A_1 = -8.14$ $\alpha = 1.98, \alpha_1 = 2.91, a_3 = .006, c_3 = 2.89$	6.32	240.15	.999

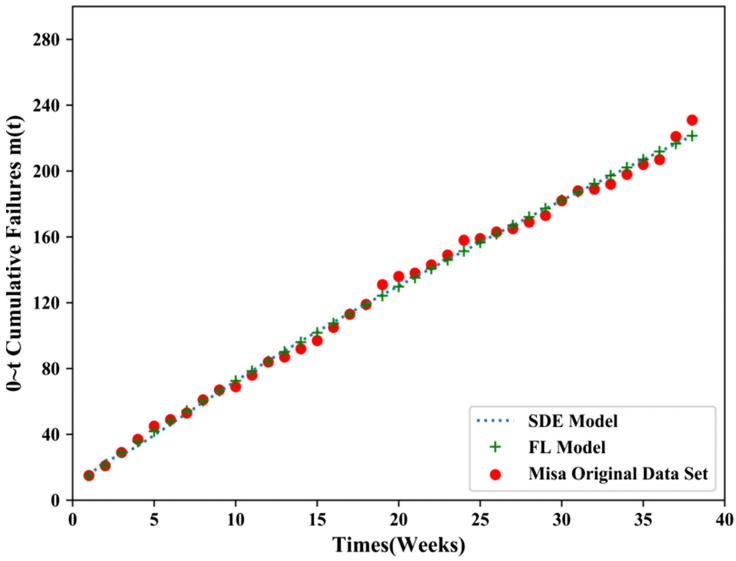

Fig. 1. Goodness of fit curves for DS-I.

4.2 Data Set II

The Ohba data set is mentioned in a paper written by Ohba for a database software system that contains approximately 1317,000 lines of code [12, 31, 32]. The software was tested for 19 weeks, during which 47.65 computer hours were used, and 328 faults were removed. Different from the DS-I, DS-II does not classify the failures when recording the failure data. Therefore, we choose three models that are tested with DS-II to compare with the proposed and prove that the proposed model has an excellent performance in both classified and unclassified data sets.

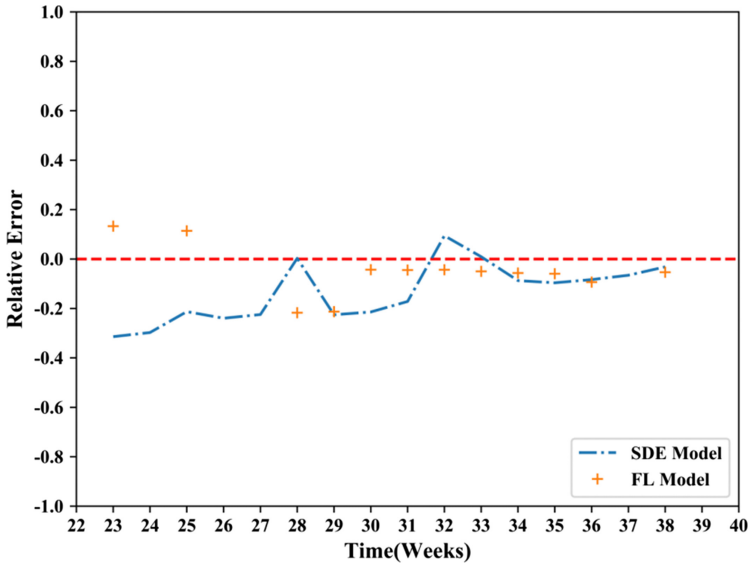

Fig. 2. RE curves on DS-I.

Analysis of Fitting Results. Table 2 lists the estimates of different model parameters on DS-II, including the G-O model, and the traditional Yamada Delayed S-Shaped model. We also give the values of SSE, MSE, and R in Table 2. It is observed that the proposed model has the smallest value of SSE and MSE, and the value of R for the proposed model is the closest to one when compared with other SRGMs.

Table 2. Fitting parameters for DS-II.

Models under comparisons	Parameter estimation	MSE	SSE	R
G-O model	$N = 760.53, r = .03$	139.82	2656.48	.986
Delay S-shaped model	$N = 374.05, r = .21$	168.67	3204.79	.984
Improved G-O model	$N = 451.32, b = 13.03, A = .04$	89.81	1706.43	.991
Proposed SRGM (fault level (FL) model)	$p_1 = .073, p_2 = .755, p_3 = .172$ $N = 506, A = 19.28, A_1 = 61.88$ $\alpha = .301, \alpha_1 = 1.64, a_3 = -0.095, c_3 = .47$	32.47	616.95	.997

Different from the DS-I, we found that Level II and Level III faults account for a significant proportion of the DS-II, while Level I accounts for a small percentage. This is because different software development environments and application scenarios have different fault level distribution.

To clearly show the fitting effect diagram, we chose the two models with the best fitting effect for comparison. The two models with better fitting effect are the proposed model, and the Improved G-O model. Figure 3 describes the contrast between the appropriate value ($m(t_i)$) of each failure data in the DS-II by the above two models and the actual observed failure data (m_i). On average, the proposed model performs better in data set fitting.

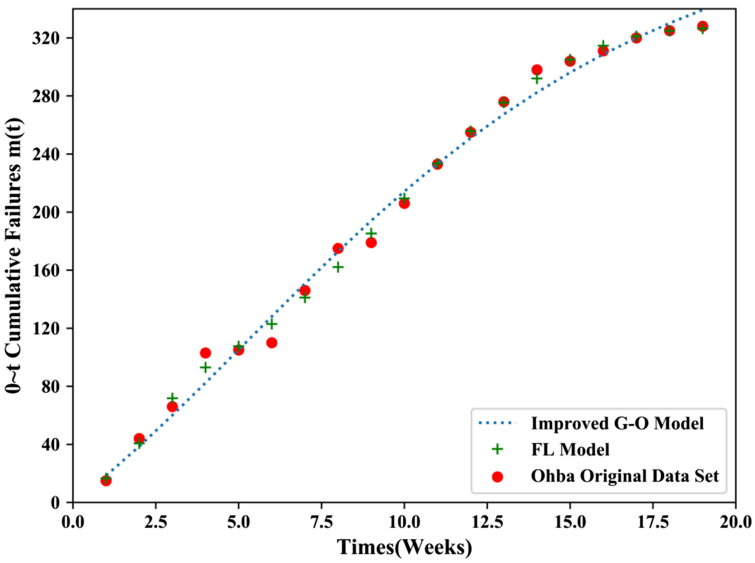

Fig. 3. Goodness of fit curves for DS-II.

Analysis of Prediction Results. In DS-II, we train with the failure data of the first 12 weeks, and compare the predicted value with the real value to get the RE curves in Fig. 4. For the convenience of observation, the RE curves of two models with a better fitting effect on the DS-II are depicted in Fig. 4. It is worth noting the curve of the Improved G-O model deviates from zero by a large margin. The RE value of the proposed model is the closest to zero as a whole, and the speed of the curve approaching zero is the fastest after eighteen weeks, which indicates that the proposed model has excellent predicted results on the DS-II.

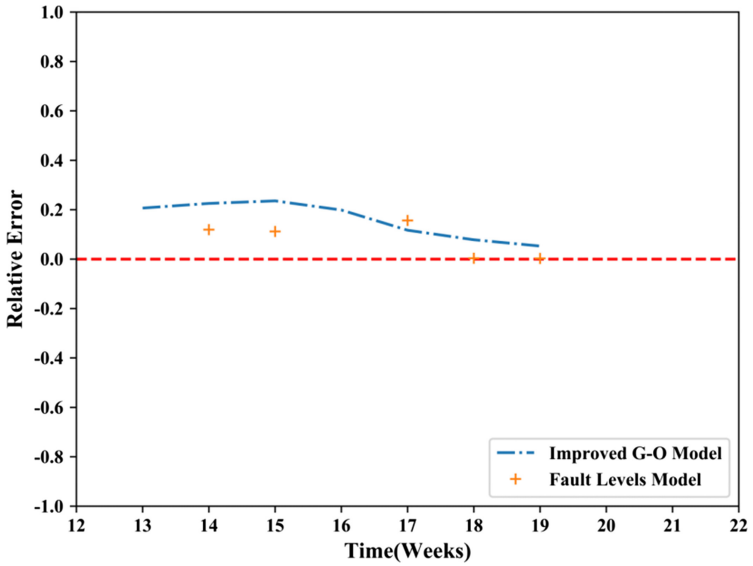

Fig. 4. RE curves on DS-II.

5 Conclusion

This paper provides a new SRGM based on three different types of fault severity. The model not only considers the existence of more than one type of software failure but also considers the possibility of imperfect debugging in the real world, introducing failure introduction rate and non-removable software failure rate. This makes the establishment of the model more in line with the actual situation and the calculation is simple, which is convenient for transplantation and application. The simulation results on two different types of data sets show that, compared with the previous methods, the fault severity classification can effectively improve the fitting effect and prediction effect of the traditional software reliability model, which plays an essential role in the theoretical research and engineering application of the software reliability model.

References

1. Mengmeng, Z., Hoang, P.: A software reliability model incorporating martingale process with gamma-distributed environmental factors. Ann. Oper. Res., 1–22 (2018)
2. Jaiswal, A., Malhotra, R.: Software reliability prediction using machine learning techniques. Int. J. Syst. Assur. Eng. Manag. **9**(1), 230–244 (2018)
3. Chatterjee, S., Shukla, A.: A unified approach of testing coverage based software reliability growth modelling with fault detection probability, imperfect debugging, and change point. J. Softw. Evol. Process **31**(3), e2150 (2019)
4. Aihua, G., Chunyang, Z., Qingxin, H.: Improvement of G-O model of software reliability growth. J. Inner Mongolia Univ. Nat. Sci. Edn. **45**(2), 84–87 (2014)
5. Kumar, R., Kumar, S., Tiwari, S.K.: A study of software reliability on big data open source software. Int. J. Syst. Assur. Eng. Manage. **10**(2), 1–9 (2019)

6. Mengmeng, Z., Hoang, P.: A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal. Comput. Lang. Syst. Struct. **53**, 27–42 (2017)
7. Hwang, S., Pham, H.: Quasi-renewal time-delay fault-removal consideration in software reliability modeling. IEEE Trans. Syst. Man Cybern. **39**(1), 200–209 (2009)
8. Garmabaki, A.H., Aggarwal, A.G., Kapur, P.K.: Multi up-gradation software reliability growth model with faults of different severity. In: Industrial Engineering and Engineering Management, pp. 1539–1543 (2011)
9. Goseva-Popstojanova, K., Trivedi, K.S.: Failure correlation in software reliability models. IEEE Trans. Reliab. **49**(1), 37–48 (2000)
10. Singh, V.B., Sharma, M., Pham, H.: Entropy based software reliability analysis of multi-version open source software. IEEE Trans. Softw. Eng. **44**(12), 1207–1223 (2018)
11. Yaghoobi, T.: Parameter optimization of software reliability models using improved differential evolution algorithm. Math. Comput. Simul. **17**, 46–62 (2020)
12. Ohba, M.: Software reliability analysis models. IBM J. Res. Dev. **28**(4), 428–443 (1984)
13. Nagaraju, V., Fiondella, L., Zeephongsekul, P., Jayasinghe, C.L., Wandji, T.: Performance optimized expectation conditional maximization algorithms for nonhomogeneous poisson process software reliability models. IEEE Trans. Reliab. **66**(3), 722–734 (2017)
14. Vizarreta, P.: Assessing the maturity of SDN controllers with software reliability growth models. IEEE Trans. Netw. Serv. Manage. **15**(3), 1090–1104 (2018)
15. Chatterjee, S., Singh, J.B., Roy, A.: NHPP-Based software reliability growth modeling and optimal release policy for N-Version programming system with increasing fault detection rate under imperfect debugging. Proc. Natl. Acad. Sci. India Sect. A Phys. Sci. **90**, 11–26 (2020)
16. Peng, R., Ma, X., Zhai, Q.: Software reliability growth model considering first-step and second-step fault dependency. J. Shanghai Jiaotong Univ. (Sci.) **24**(4), 477–479 (2019)
17. Li, Q., Pham, H.: A generalized software reliability growth model with consideration of the uncertainty of operating environments. IEEE Access **7**, 84253–84267 (2019)
18. Utkin, L.V., Coolen, F.P.A.: A robust weighted SVR-based software reliability growth model. Reliab. Eng. Syst. Saf. **176**(8), 93–101 (2018)
19. Briones, B.A.: Wiley encyclopedia of electrical and electronics engineering. Charleston Adv. **21**(2), 51–54 (2019)
20. Dai, Y., Xie, M., Poh, K.: Modeling and analysis of correlated software failures of multiple types. IEEE Trans. Reliab. **54**(1), 100–106 (2005)
21. Rani, P., Mahapatra, G.S.: A novel approach of NPSO on dynamic weighted NHPP model for software reliability analysis with additional fault introduction parameter. Heliyon **5**(7) (2019)
22. Lin, C., Huang, C.: Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models. J. Syst. Softw. **81**(6), 1025–1038 (2008)
23. Li, Q., Pham, H.: A testing-coverage software reliability model considering fault removal efficiency and error generation. PLOS ONE **12**(7), e0181524 (2017)
24. Huang, C., Kuo, S., Lyu, M.R.: An assessment of testing-effort dependent software reliability growth models. IEEE Trans. Reliab. **56**(2), 198–211 (2007)
25. Jing, Z., Hongwei, L., Gang, C., Xiaozong, Y.: A software reliability growth model considering differences between testing and operation. J. Comput. Res. Dev. **43**(3), 503 (2006)
26. Misra, P.N.: Software reliability analysis. IBM Syst. J. **22**(3), 262–270 (1983)
27. Hui, Z., Liu, X.: Research on software reliability growth model based on gaussian new distribution. Procedia Comput. Sci. **166**, 73–77 (2020)
28. Kapur, P.K., Anand, S., Yamada, S., Yadavalli, V.S.: Stochastic differential equation-based flexible software reliability growth model. Math. Probl. Eng. **2009**, 1–15 (2009)
29. Xie, J., Jinxia, A.N., Zhu, J.: NHPP software reliability growth model considering imperfect debugging. J. Softw. **21**(5), 942–949 (2010)

30. Chatterjee, S., Chaudhuri, B., Bhar, C.: Optimal release time determination using FMOCCP involving randomized cost budget for FSDE-based software reliability growth model. Int. J. Reliab. Qual. Saf. Eng. **27**(1), 257–279 (2020)
31. Yamada, S., Ohba, M., Osaki, S.: S-shaped reliability growth modeling for software error detection. IEEE Trans. Reliab. **32**(5), 475–484 (1983)
32. Laura, P.: Software fault tolerance techniques and implementation. Artech (2001)