



A logistic software reliability model with Loglog fault detection rate

Md. Asraf ul Haque¹ · Nesar Ahmad¹

Received: 21 January 2024 / Accepted: 14 May 2024
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2024

Abstract

Research on software reliability modeling is essential for improving software quality, reducing costs, and ensuring customer satisfaction in the ever-expanding digital landscape. To achieve the desired level of reliability in the testing phase, software reliability growth models (SRGMs) are used. The goal of this study is to develop a novel SRGM based on a hypothesis obtained from a logistic model. The logistic model is commonly used to describe the rate of change in the size of a population over time. According to this model, a population's per capita growth rate declines as it approaches the carrying capacity, which is the population's upper bound. Similarly, in the software testing process, the fault detection rate per fault in the software grows faster in the beginning and later decreases with testing time until the maximum fault content is discovered. Considering the fault detection rate based on a log–log distribution, the proposed model is constructed in an NHPP analytical framework. The model is compared to six existing reliability models using four different evaluation criteria. The findings are quite encouraging.

Keywords Software reliability · Software testing · SRGM · NHPP model · Fault detection rate

Abbreviations:

SRGM	Software reliability growth model
NHPP	Non-homogeneous Poisson process
MSE	Mean square error
PRR	Predictive ratio risk
R^2	Correlation coefficient or coefficient of determination
AIC	Akaike information criterion
$m(t)$	Cumulative number of faults detected by time t
N	Total no. of faults that exist in the s/w before testing
$b(t)$	Time dependent fault detection rate function
GoF	Goodness of fit
LSE	Least square estimation

1 Introduction

Due to the fast advancement of digital systems and the widespread usage of software to operate them, it is essential to accurately and thoroughly assess software reliability to examine system reliability [1]. The software industry faces a big challenge in producing highly reliable software while complying with a strict timeline and budget [2]. As testing progresses, the faults are found and fixed, and it is assumed that the software system's reliability improves [3]. Testing and debugging the system until it reaches the required degree of reliability is the fundamental goal of software reliability modeling [4]. The mathematical models that depict this behavior of the testing process to predict software reliability are known as SRGMs. These models have broad implications for software development, maintenance, and management, enabling organizations to enhance software quality, reliability, and performance while minimizing risks and maximizing resource efficiency [5, 6]. When a model is used to abstract reality, a few assumptions are necessary [6, 7]. The model's form and the meaning of the model's parameters are determined by these assumptions. Important phases in the construction of these mathematical models include the identification of realistic assumptions and the modeling of those assumptions within the right framework. The NHPP

✉ Md. Asraf ul Haque
md_asraf@zhcet.ac.in

Nesar Ahmad
nesar.ahmad@gmail.com

¹ Department of Computer Engineering, Aligarh Muslim University, Aligarh 202002, India

models are capable of providing an analytical framework for the software fault removal phenomenon [8]. In NHPP, the mean value function $m(t)$, which is a cumulative number of faults identified over time t , is the function that represents the pattern [9]. A lot of research on software reliability has been carried out through SRGM [5–12]. Researchers focused on developing new SRGMs by considering different aspects of testing processes, i.e., testing coverage, testing effort, testing skill, fault types, etc. Their aim is to fulfill the growing need for an accurate model that can guarantee a minimum prediction error. A good software reliability model should be simple, widely applicable across different software failure datasets, and based on realistic assumptions. Some notable models in the domain of software reliability engineering are the Jelinski–Moranda (JM) model [13], Musa–Okumoto (MO) model [14], Goel–Okumoto model [15], Delayed S-shaped model [16], Inflection S-shaped model [17], etc. The JM model [13] which is considered as the first SRGM, is based on the assumption of a constant fault detection rate, making it relatively simple but not always suitable for modern complex software development practices. The MO model [14] is an extension of the JM model. It introduces a time-dependent hazard function, which allows the fault detection rate to decrease as testing progresses. The Goel–Okumoto model [15] is based on the assumption that the software failure rate decreases exponentially over time as defects are discovered and fixed. The Delayed S-shaped model [16] is a variation of the S-shaped logistic growth model. It is used to represent software reliability growth with an initial “delay” or slower improvement in reliability, followed by a more rapid increase as testing continues. The Inflection S-shaped model [17] represents software reliability growth with an initial slow rate of improvement, followed by an inflection point where the rate of improvement increases significantly. As software systems continue to evolve, ongoing research efforts are needed to refine and extend the SRGMs to address emerging challenges. Recent developments in software reliability growth modeling have demonstrated the importance of adapting models to contemporary software development practices. These models are now looking into the various unexplored facets of the software testing process and adding them to the model. Some of them employ machine learning algorithms [18–20], some examine uncertainty factors of the testing or operational environment [21–24], and some mix the various techniques to create hybrid models [25, 26].

In this paper, we present a model that utilizes the logistic growth function and incorporates log–log distribution for predicting software reliability. PF Verhulst introduced the logistic function in the middle of the nineteenth century [27]. By modifying the exponential growth model, he developed it as a population growth model, which is known as the logistic growth model. In this context, the word “logistic” has no special meaning or significance other than the fact that it is

widely accepted. Apart from biology, the logistic model is now widely used in chemistry, geoscience, political science, and statistics. According to the model, each member of a population will have equal access to resources and a similar probability of surviving. The population’s per capita growth will decrease with time due to resource limitations. First, Yamada and Osaki [28], employed this model to estimate software reliability growth. Later, many researchers have used this model in evaluating software reliability under the assumption that each fault has an equal chance of discovery and that the fault detection rate per fault will decrease as testing progresses [29–31]. It allows for a realistic representation of software reliability improvement over time, accommodating various project-specific factors. Some researchers have shown promising results when they consider the fault detection rate function based on log–log distribution [32–34]. The purpose of this paper is to suggest a new SRGM by taking advantage of both the logistic model and the log–log fault detection rate function. A log–log distribution is a type of probability distribution where the probability of an event or value is proportional to the logarithm of its value. This means that as the value of the event or variable increases, the probability decreases exponentially. In contrast to the typical bathtub-shaped curve produced by the Weibull distribution, the log–log distribution offers a Vtub-shaped curve. In statistical modeling, the Weibull distribution is widely used. However, Vtub-shaped curves cover a wider range of monotone failure rates in addition to the characteristics of bathtub-shaped curves with increasing or decreasing failure rates. Other advantages of log–log distribution in software reliability modeling include:

1. Ability to detect early failure trends: The log–log distribution is very sensitive to changes in the failure rate, especially at the beginning of the testing process. This makes it easier to detect early failure trends and take corrective action before the software is released.
2. Non-linear behavior: The log–log distribution captures the non-linear behavior of failure rates in software testing, which is a more accurate representation of real-world failure patterns. This can help software developers better understand how their software behaves under different conditions and make more informed decisions about how to improve its reliability.
3. Model fitting: The log–log distribution is easy to fit to the data using regression techniques. This makes it a practical choice for software reliability growth modeling, where large amounts of data need to be analyzed.

The rest of the paper is structured as follows. Section 2, explains the derivation of a new SRGM. Section 3 describes the steps involved to analyze the model. Section 4 compares the model performance with some well-known models. And

Sect. 5 concludes the paper outlining the findings, strengths and limitations of the study.

2 Proposed SRGM

A mathematical model based on the Verhulst logistic equation was designed to predict population growth. The model is represented as:

$$\frac{dp}{dt} = r \times p \left(1 - \frac{p}{k}\right) \tag{1}$$

where, ‘*p*’ and ‘*r*’ represent the current population and the growth rate respectively. ‘*k*’ is the carrying capacity, i.e., maximum number of population that can survive in a specific environment with the available resources. In software testing process, it is observed that the fault detection rate at the beginning is very high, and this rate gradually drops as the testing continues. Anticipating this behavior of the testing process, the above model can be replicated as [29, 30]:

$$\frac{dm(t)}{dt} = b(t)m(t) \left(1 - \frac{m(t)}{N}\right) \tag{2}$$

m(t) is the total number of faults detected by time *t*, *b(t)* is the time-dependent fault detection rate function, and ‘*a*’ is total no. of faults that exist in the software before testing. By solving Eq. (2), we can find the value of *m(t)*.

$$m(t) = \frac{N}{1 + e^{c - \int_0^t b(t) dt}} \tag{3}$$

Here, ‘*c*’ is an integral constant. We assume that the function *b(t)* follows Vtub-shaped based on log–log distribution that has been considered in a few models suggested by Pham [12–14]. It is expressed as:

$$b(t) = \beta \ln(\alpha) t^{\beta-1} \alpha^{t^\beta} \tag{4}$$

where ‘*α*’ is the scale parameter and ‘*β*’ is the shape parameter of the Loglog distribution. Replacing the value of *b(t)* in (3) and considering a constant $\gamma = c + 1$, we can derive the following:

$$m(t) = \frac{N}{1 + e^{(\gamma - \alpha^{t^\beta})}} \tag{5}$$

This is our new model which uses NHPP framework. The mean value function also known as *m(t)*, is the characteristic that NHPP models use to describe data. The suggested model, some well-known models, and their related functions are listed in Table 1.

Table 1 A list of SRGMs

Model	Mean value function <i>m(t)</i>
G-O model [15]	$a(1 - e^{-bt})$
DSS model [16]	$a(1 - (1 + bt)e^{bt})$
ISS model [17]	$\frac{a(1 - e^{-bt})}{1 + \beta e^{-bt}}$
P-N-Z model [35]	$\frac{a(1 - e^{-bt})(1 - \frac{a}{\beta}) + \alpha at}{1 + \beta e^{-bt}}$
Testing coverage model [23]	$N \left(1 - \left(\frac{\beta}{\beta + (at)^\beta}\right)^\alpha\right)$
Loglog model [33]	$N(1 - e^{-(a^{t^\beta} - 1)})$
New model	$\frac{N}{1 + e^{(\gamma - \alpha^{t^\beta})}}$

3 Model analysis and comparison

The suitability of the SRGMs is assessed in two steps: first, the model’s parameters are calculated, and second, the model fittings are verified using various comparison criteria. In this section, we validate the proposed model and compare the findings to various known models given in Table 1.

3.1 Evaluation criteria

SRGMs are not precise representations of reality; assumptions and many environmental factors affect the validity and accuracy of their predictions. Therefore, it is essential to evaluate a model’s appropriateness, which entails figuring out its advantages and disadvantages as well as the degree to which the results offered may be trusted. The potential of an SRGM to recreate the observed software behavior and predict future behavior with the help of observed failure data can be used to evaluate it. There are a variety of comparison standards called "goodness-of-fit" or GoF criteria that are used to assess how well a model fits a given collection of data. In this study, we use four criteria (i.e., the mean square error (MSE), PRR, *R*² and AIC) that are defined as follows [36–38]:

- MSE: It measures the difference between observed data and their expected values.

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

Here, ‘*k*’ denotes the database size, and ‘*p*’ is the number of parameters of a model.

- PRR: It measures the deviation per estimate of a model.

$$PRR = \sum_{i=1}^k \left(\frac{(m(t_i) - m_i)}{m(t_i)} \right)^2$$

- R^2 : It represents how close the data are to the fitted regression line.

$$R^2 = 1 - \frac{\sum_{i=1}^k (m_i - m(t_i))^2}{\sum_{i=1}^k \left(m_i - \sum_{j=1}^k \frac{m_j}{k} \right)^2}$$

- AIC: It is a single number score that is used to decide which of many models is most likely to be the best one for the dataset.

$$AIC = k \times \ln(\text{MSE}) + 2p$$

The better an SRGM is, the lower its MSE, PRR, and AIC values, and the higher its R^2 value [31, 36].

3.2 Dataset used

Reference [39] reports three sets of failure data for three different releases of sizable medical record software (MRS). We utilized the first two releases' data for our analysis (i.e., MRS-1 and MRS-2). The software has 188 modules in total. It had 173 modules when it was first released, and 15 more modules were added later in the updated versions. In MRS-1, there were a total of 176 errors collected during the 18 weeks of testing. MRS-2 contains 204 errors that were found during the testing period of 17 weeks. Table 2 displays both the MRS-1 and MRS-2 datasets.

3.3 Parameter estimation

A software reliability model is a function of various parameters. The first step in employing an SRGM is to determine its parameters on a sample dataset. The process of parameter estimation can be done in a variety of ways. In the case of small datasets, the Least Square Estimation (LSE) approach is considered a quite efficient method for parameter estimation [37]. We have used the "CurveExpert Professional" tool to smoothly carry out the experimental work. It generates LSE results by fitting failure data to a non-linear model equation on a given dataset. Table 3 shows the parameter values of all seven models on both datasets.

4 Results and comparison

Now with the help of the parameter values we can easily calculate the GoF criteria (i.e., MSE, PRR, AIC, and R^2) from their formula. In statistical models, there is no right or

Table 2 Failure data of MRS

Test week	Cumulative faults	
	Release-1	Release-2
1	28	90
2	29	107
3	29	126
4	29	145
5	29	171
6	37	188
7	63	189
8	92	190
9	116	190
10	125	190
11	139	192
12	152	192
13	164	192
14	164	192
15	165	203
16	168	203
17	170	204
18	176	–

wrong. It's all about how good or bad they are. The suggested model produced the best fitting results on both datasets. The model's accuracy has been compared to other SRGMs in Tables 4 and 5. We can summarize the results of the new model as follows:

- MSE: The new model offers the smallest MSE in both datasets, i.e., 52.91 (MRS-1) and 26.855 (MRS-2). The second smallest MSEs achieved in the study were 116.3 for the ISS model and 53.531 for the Loglog Model in these datasets. The MSE values of the new model are near about half of the second best models.
- PRR: The PRR values are 0.477 and 0.009. Both are the smallest among all models and the second smallest values are more than four times in the respective datasets.
- R^2 : R-squared is the "percent of variance explained" by the model. A higher R-squared value indicates a higher amount of variability being explained by our model and vice-versa. The R^2 values of the new model are 0.987 and 0.982. Both are the largest among all model' values in the respective datasets.
- AIC: The AIC values are 71.153 and 54.377. Both are the lowest among all models' values. They indicate that the proposed model is relatively simple with four parameters.

Table 3 Parameter estimation using LSE

Model	MRS-1	MRS-2
G-O model	$a = 985.855, b = 0.012$	$a = 197.386, b = 0.399$
DSS model	$a = 226.062, b = 0.174$	$a = 192.528, b = 0.882$
ISS model	$a = 176.493, b = 0.423, \beta = 26.958$	$a = 197.386, b = 0.399, \beta = 0.00001$
P-N-Z model	$a = 171.371, b = 0.429, \alpha = 0.002, \beta = 27.217$	$a = 183.125, b = 0.463, \alpha = 0.007, \beta = 0.0001$
Testing coverage model	$N = 174.38, a = 0.027, b = 2.471, \alpha = 104.137, \beta = 3.329$	$N = 204.232, a = 0.009, b = 0.775, \alpha = 95.734, \beta = 5.298$
Loglog model	$N = 170.524, a = 1.011, b = 1.866$	$N = 199.153, a = 1.505, b = 0.572$
New model	$N = 167.912, \alpha = 1.142, \beta = 1.044, \gamma = 3.216$	$N = 194.779, \alpha = 1.206, \beta = 1.143, \gamma = 1.343$

Table 4 Performance analysis of the models on MRS-1

Model	MSE	PRR	R^2	AIC
G-O model	299.895	2.616	0.924	104.542
DSS model	202.853	2.363	0.949	97.505
ISS model	116.3	1.822	0.972	88.329
P-N-Z model	125.504	1.838	0.972	90.458
Testing coverage model	172.573	2.381	0.965	96.857
Loglog model	134.727	2.002	0.969	90.977
New model	52.91	0.477	0.987	71.153

Table 5 Performance analysis of the models on MRS-2

Model	MSE	PRR	R^2	AIC
G-O model	80.678	0.101	0.939	74.510
DSS model	232.628	0.333	0.823	92.513
ISS model	86.44	0.101	0.939	76.510
P-N-Z model	84.905	0.087	0.944	76.945
Testing coverage model	72.389	0.048	0.956	74.874
Loglog model	53.531	0.039	0.962	68.364
New model	26.855	0.009	0.982	54.377

The graphs in Figs. 1 and 2 show the model's predictive capability on each dataset, comparing the estimated faults to the observed faults.

5 Conclusion

While many software reliability growth models have been proposed in the literature, there is often a lack of real-world validation to assess their effectiveness and applicability in practical settings. Investigating various approaches is necessary to comprehend software behavior and perform an efficient analysis of failure data. The paper proposes a software reliability model based on a logistic growth model, whereas the fault detection rate follows a log-log distribution. The main advantage of using log-log distributions in software reliability growth modeling is their suitability for

capturing long-tailed data. In many software systems, the fault detection rate at the beginning is very high, and this rate gradually drops as the testing continues. Based on this behavior, we applied the logistic growth model to describe this behavior of the fault detection rate. The mathematical model for estimating the total number of faults over time is developed. The performance of the suggested model is then examined using numerical data from four standard GOF measurements and compared to a number of existing models. The outcomes clearly demonstrate that the recommended model performs better than the other models in all four assessments. The key strengths of the suggested model are its high predictive capability and flexibility. However, the model could be negatively impacted by a few limitations such as dataset dependency, complexity, and its inability to address uncertainty issues. In future, we will carry out more case studies

Fig. 1 Predictive capability of the model for MRS-1

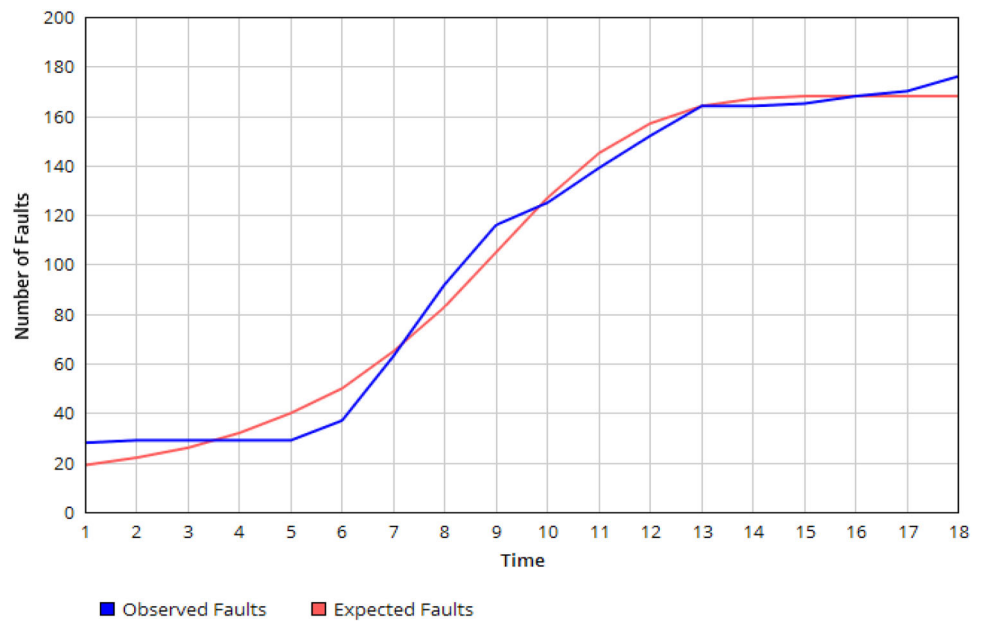
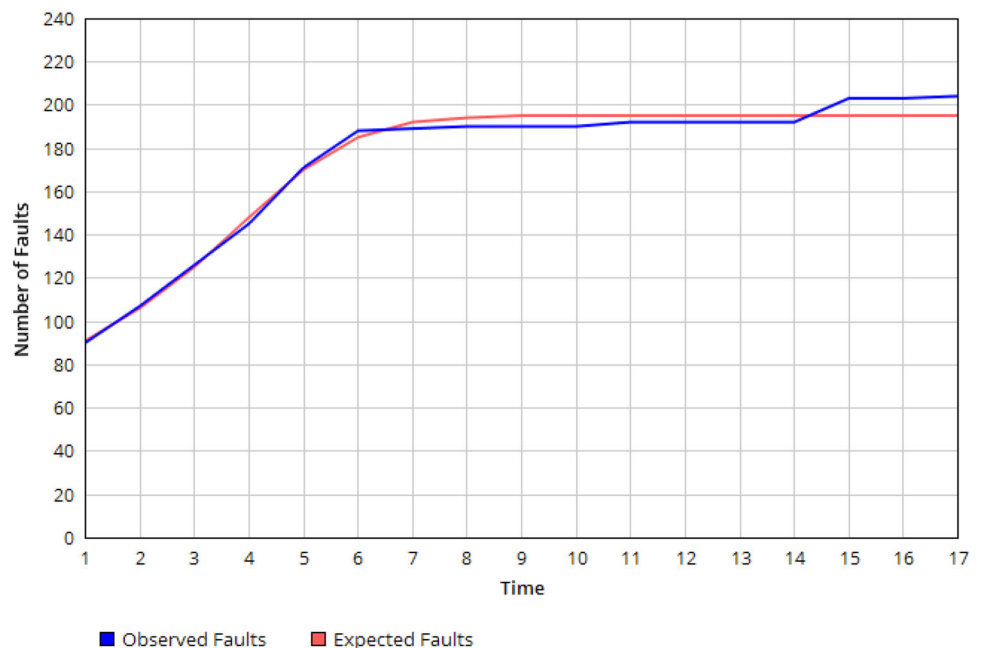


Fig. 2 Predictive capability of the model for MRS-2



to establish the supremacy of the model using different sets of failure data and evaluation criteria.

Author contributions Contribution of M. A. Haque: 1. Study conception and design 2. Material preparation 3. Data collection and analysis Contributions of N.A. : 1. Supervision 2. Writing- Reviewing and Editing 3. Validation

Funding The authors declare that no funds, grants, or other supports were received during the preparation of this manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

- Garg, R., Sharma, K., Kumar, R., Garg, R.: Performance analysis of software reliability models using matrix method. *Int. J. Comput. Inf. Eng.* **4**(11), 1646–1653 (2010)
- Haque, M.A., Ahmad, N.: Key issues in software reliability growth models. *Recent Adv. Comput. Sci. Commun.* **15**(5), e060422186806 (2022)

3. Lyu, M.R. (ed.): Handbook of Software Reliability Engineering. IEEE Computer Society Press, Washington, DC (1996)
4. Xie, M.: Software Reliability Modeling. World Scientific Publishing, Singapore (1991)
5. Kapur, P.K., Garg, R.B.: A software reliability growth model for an error-removal phenomenon. *Softw. Eng. J.* **7**, 291–294 (1992)
6. Haque M.A., Ahmad, N.: Modified Goel-Okumoto software reliability model considering uncertainty parameter. In: MMCITRE-2021, *Advances in Intelligent Systems and Computing*, vol 1405 (2022)
7. Du X., Qiang Z.: Software reliability growth models based on non-homogeneous poisson process. In: 2009 International Conference on Information Engineering and Computer Science, pp. 1–3 (2009)
8. Haque M.A., Ahmad, N.: An NHPP-based SRGM with time dependent growth process. In: 6th International Conference on Signal Processing, Computing and Control, Solan, India (2021)
9. Haque, M.A., Ahmad, N.: An imperfect SRGM based on NHPP. In: 3rd IEEE-International conference on inventive research in computing applications (ICIRCA), pp. 1574–1577. Coimbatore (2021)
10. Asraful Haque, M.: Software reliability models: a brief review and some concerns. In: CSDEIS 2022. *Lecture Notes on Data Engineering and Communications Technologies*, vol 158 (2023)
11. Iqbal, J.: Software reliability growth models: a comparison of linear and exponential fault content functions for study of imperfect debugging situations. *Cogent Eng.* **4**(1), 1286739 (2017)
12. Saraf, I., Iqbal, J., Shrivastava, A.K., Khurshid, S.: Modelling reliability growth for multi-version opensource software considering varied testing and debugging factors. *Qual. Reliab. Eng. Int.* **38**(4), 1814–1825 (2022)
13. Jelinski, Z., Moranda, P.B.: Software reliability research. In: Freiburger, W. (ed.) *Statistical Computer Performance Evaluation*, pp. 465–484. Academic Press, New York (1972)
14. Musa, J.D., Okumoto, K.: A logarithmic Poisson execution time model for software reliability measurement. In: *Proceedings of the 7th Int. Conference on Software Engineering*, pp. 230–238. IEEE Press, Piscataway, NJ, USA (1984)
15. Goel, A.L., Okumoto, K.: Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Reliab.* **R-28**(3), 206–211 (1979)
16. Yamada, S., Ohba, M., Osaki, S.: S-shaped software reliability growth models and their applications. *IEEE Trans. Reliab.* **R-33**(4), 289–292 (1984)
17. Ohba, M.: Inflection S-shaped software reliability growth model. In: Osaki, S., Hatoyama, Y. (eds.) *Stochastic Models in Reliability Theory. Lecture Notes in Economics and Mathematical Systems*, vol. 235. Springer, Berlin (1984)
18. Batool, I., Khan, TA.: Software fault prediction using deep learning techniques. PREPRINT (Version1). Research Square <https://doi.org/10.21203/rs.3.rs-2089478/v1> (2022)
19. Jaiswal, A., Malhotra, R.: Software reliability prediction using machine learning techniques. *Int. J. Syst. Assur. Eng. Manag.* **9**, 230–244 (2018)
20. Devi, B.V., Devi, R.K.: Software reliability models based on machine learning techniques: a review. *AIP Conf. Proc.* **2463**(1), 020038 (2022)
21. Haque, M.A., Ahmad, N.: Software reliability modeling under an uncertain testing environment. *Int. J. Model. Simul.* (2023). <https://doi.org/10.1080/02286203.2023.2201905>
22. Lee, D., Chang, I., Pham, H.: Study of a new software reliability growth model under uncertain operating environments and dependent failures. *Mathematics* **11**(18), 3810 (2023). <https://doi.org/10.3390/math11183810>
23. Chang, I.H., Pham, H., Lee, S.W., Song, K.Y.: A testing-coverage software reliability model with the uncertainty of operation environments. *Int. J. Syst. Sci. Oper. Logist.* **1**(4), 220–227 (2014)
24. Pham, H.: A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments. *Optimization* **63**(10), 1481–1490 (2014)
25. Lee, D.H., Chang, I.H., Pham, H.: Software reliability model with dependent failures and SPRT. *Mathematics* **8**, 1366 (2020)
26. Pradhan, V., Dhar, J., Kumar, A., Bhargava, A.: An S-Shaped Fault Detection and Correction SRGM Subject to Gamma-Distributed Random Field Environment and Release Time Optimization, pp. 285–300. Springer, Berlin (2020)
27. Bacaër, N.: Verhulst and the logistic equation (1838). In: *A Short History of Mathematical Population Dynamics*, pp. 35–43. Springer, London (2011). https://doi.org/10.1007/978-0-85729-115-8_1
28. Yamada, S., Osaki, S.: Software reliability growth modeling: models and applications. *IEEE Trans. Softw. Eng.* **SE-11**(12), 1431–1437 (1985)
29. Pham, H.: A logistic fault-dependent detection software reliability model. *J. Univ. Comput. Sci.* **24**(12), 1717–1730 (2018)
30. Haque, M.A., Ahmad, N.: An effective software reliability growth model. *Saf. Reliab.* **40**, 209–220 (2021). <https://doi.org/10.1080/09617353.2021.1921547>
31. Haque, M.A., Ahmad, N.: A logistic growth model for software reliability estimation considering uncertain factors. *Int. J. Reliab. Qual. Saf. Eng.* **28**(05), 2150032 (2021)
32. Pham, H.: A Vtub-shaped hazard rate function with applications to system safety. *Int. J. f Reliab. Appl.* **3**(1), 1–16 (2002)
33. Pham, H.: Loglog fault-detection rate and testing coverage software reliability models subject to random environments. *Vietnam J. Comput. Sci.* **1**(1), 39–45 (2014)
34. Al-turk, L.I.: Characteristics and application of the NHPP log-logistic reliability model. *Int. J. Stat. Prob.* **8**(1), 44 (2019)
35. Pham, H., Nordmann, L., Zhang, X.: A general imperfect software debugging model with S-shaped fault detection rate. *IEEE Trans. Reliab.* **48**, 169–175 (1999)
36. Sharma, K., Garg, R., Nagpal, C.K., Garg, R.K.: Selection of optimal software reliability growth models using a distance based approach. *IEEE Trans. Reliab.* **59**(2), 266–276 (2010)
37. Haque, M.A., Ahmad, N.: A software reliability growth model considering mutual fault dependency. *Reliab. Theory Appl.* **16**(2), 222–229 (2021)
38. Haque, M.A., Ahmad, N.: A software reliability model using fault removal efficiency. *J. Reliab. Stat. Stud.* **15**(2), 459–472 (2022)
39. Stringfellow, C., Andrews, A.A.: An empirical method for selecting software reliability growth models. *Empir. Softw. Eng.* **7**(4), 319–343 (2002)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.