




Article

Efficiency Evaluation of Software Faults Correction Based on Queuing Simulation

Yuka Minamino ^{1,*}, Yusuke Makita ¹, Shinji Inoue ² and Shigeru Yamada ¹

¹ Faculty of Engineering, Tottori University, 4-101, Minami, Koyama-cho, Tottori-shi, Tottori 680-8552, Japan; b18t4093a@edu.tottori-u.ac.jp (Y.M.); yamada@tottori-u.ac.jp (S.Y.)

² Faculty of Informatics, Kansai University, 2-1-1, Ryozenji-cho, Takatsuki-shi, Osaka 569-1095, Japan; ino@kansai-u.ac.jp

* Correspondence: minamino@tottori-u.ac.jp

Abstract: Fault-counting data are collected in the testing process of software development. However, the data are not used for evaluating the efficiency of fault correction activities because the information on the fault detection and correction times of each fault are not recorded in the fault-counting data. Furthermore, it is difficult to collect new data on the detection time of each fault to realize efficiency evaluation for fault correction activities from the collected fault-counting data due to the cost of personnel and data collection. In this paper, we apply the thinning method, using intensity functions of the delayed S-shaped and inflection S-shaped software reliability growth models (SRGMs) to generate sample data of the fault detection time from the fault-counting data. Additionally, we perform simulations based on the infinite server queuing model, using the generated sample data of the fault detection time to visualize the efficiency of fault correction activities.

Keywords: software reliability growth model (SRGM); nonhomogeneous Poisson process (NHPP) model; queuing model; thinning method

MSC: 90B25



Citation: Minamino, Y.; Makita, Y.; Inoue, S.; Yamada, S. Efficiency Evaluation of Software Faults Correction Based on Queuing Simulation. *Mathematics* **2022**, *10*, 1438. <https://doi.org/10.3390/math10091438>

Academic Editor: Tzong-Ru Tsai

Received: 15 March 2022

Accepted: 20 April 2022

Published: 24 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Software development management is required to optimize efficiency due to the cost of testing efforts in detecting and correcting several faults during the testing process. Therefore, evaluating testing efficiency is important both during and after the project to get feedback. One way to get useful feedback is to analyze the data collected in current and past projects. Fault-counting data are a typical data set collected in the actual testing process, with records of only the total faults during the testing period, hourly, daily, or monthly. It is used to quantitatively evaluate the final quality and reliability of software products. However, the detection and correction times of each fault are not recorded.

In large-scale software and open source software development, bug tracking systems are introduced, and various data including fault detection and correction completion times can be easily collected. However, fault-counting data are recorded using a company-specific format without introducing the bug tracking system in small- or medium-scale software development. In the latter case, it is difficult to collect data, such as fault detection and correction times, due to the cost of personnel and data collection time. Therefore, we obtain new knowledge as feedback by analyzing existing fault-counting data.

Generally, when evaluating software reliability using the fault-counting data, non-homogeneous Poisson process (NHPP) models, which have high practicality due to their model structure, are used among the software reliability growth models (SRGMs) [1–3]. In previous studies, the delayed S-shaped (DSS) SRGM, incorporated in two stages of software failure occurrence and fault isolation processes, was developed [3,4]. Additionally, new SRGMs, using the concept of separating the fault detection process, expressed the

physical aspect of the fault detection process using the infinite server queuing model [4,5]. However, these proposed models cannot be used to evaluate the efficiency of fault correction activities because SRGMs, including NHPP models, express only the software failure/fault detection process, and cannot express a series of processes from the failure/fault occurrence to fault correction completion.

For example, the intensity function of NHPP models, which indicates the instantaneous fault detection rate, can estimate when most faults were detected. Knowing the peak time for fault detection helps plan future testing phases. However, estimating the busiest correction time and considering appropriate measures to eliminate the delay are factors that improve the efficiency of fault correction activities. Therefore, it is necessary to visualize fault correction activities. Equally, the software development manager needs to understand the status of fault correction activities.

In this paper, we generated the sample data of the fault detection time from the actual fault-counting data to conduct the efficiency evaluation of the fault correction activities. Next, we estimated the detection time of each fault using the thinning method [6]. Then, we applied the DSS and ISS intensity functions. Furthermore, the process from fault detection to correction completion was represented by the infinite server queuing model. Visualizing the temporal behavior of the faults being corrected in the testing phase reveals the peak time when the fault correction activities are the busiest.

2. Software Reliability Growth Model

Now, let $N(t)$ ($t \geq 0$) denote a counting process representing the total faults detected up to the testing time t . Assumed $N(t)$ follows an NHPP, the stochastic behavior of $N(t)$ is given by

$$\Pr\{N(t) = m\} = \frac{\{H(t)\}^m}{m!} \exp[-H(t)] \quad (m = 0, 1, 2, \dots), \tag{1}$$

$$h(t) = \int_0^t h(x)dx. \tag{2}$$

where $H(t)$ is the mean value function of NHPP and represents the expected cumulative number of faults detected in the time-interval $(0, t]$. $h(t)$ is the intensity function, indicating the instantaneous fault detection rate at time t . When the intensity function is constant at time t , $N(t)$ follows the homogeneous Poisson process (HPP). The instantaneous fault detection rate and mean value function of HPP are defined by the following equations:

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

$$H(t) = \lambda t. \tag{4}$$

In NHPP models with a finite number of detectable faults, the number of faults detected per unit time is assumed to be proportional to the number of remaining faults in the software at that time and is given by the following differential equation:

$$\frac{dH(t)}{dt} = b(t)[a - H(t)] \quad (b(t) > 0, t \geq 0), \tag{5}$$

where a is the initial fault content and $b(t)$ is the fault detection rate at time t .

First, assuming that the initial conditions of Equation (5) are $H(0) = 0$ and $b(t) = b$ ($b > 0$), the exponential (EXP) SRGM [1–3,7] is derived as

$$H(t) \equiv m(t) = a(1 - \exp[-bt]). \tag{6}$$

Suppose $b(t) = b$ and $a = m(t)$, the following differential equation is obtained.

$$\frac{dH(t)}{dt} = b[m(t) - H(t)]. \tag{7}$$

Solving Equation (7) under Equation (12), the DSS SRGM [1–3] is derived as

$$H(t) \equiv M(t) = a(1 - (1 + bt)\exp[-bt]) \quad (a > 0, b > 0). \tag{8}$$

Thus, the intensity function of DSS SRGM is as follows:

$$h(t) \equiv h_M(t) = ab^2t \cdot \exp[-bt]. \tag{9}$$

Next, the fault detection rate is assumed as follows:

$$b(t) \equiv b_I(t) = b \left\{ l + (1 - l) \frac{H(t)}{a} \right\} \quad (a > 0, b > 0, 0 < l \leq 1), \tag{10}$$

where l is the inflection coefficient for the fault detection ability. From Equations (5) and (10), the ISS SRGM [1–3] is derived as

$$H(t) \equiv I(t) = \frac{a(1 - \exp[-bt])}{(1 + c \cdot \exp[-bt])} \quad (c > 0), \tag{11}$$

where $c = (1 - l)/l$. The intensity function of ISS SRGM is as follows:

$$h(t) \equiv h_I(t) = \frac{ab(1 + c)\exp[-bt]}{(1 + c \cdot \exp[-bt])^2}. \tag{12}$$

3. Sample Data Generation of Fault Detection Time Using the Thinning Method

3.1. Thinning Method

The thinning method [6] is a sampling method that extended the rejection sampling method. It is used when random numbers cannot be directly obtained from the probability density function. Here, we simulated NHPP using the intensity function $h(t)$ to generate the sample data of the fault detection time. The conceptual figure of the thinning method is shown in Figure 1.

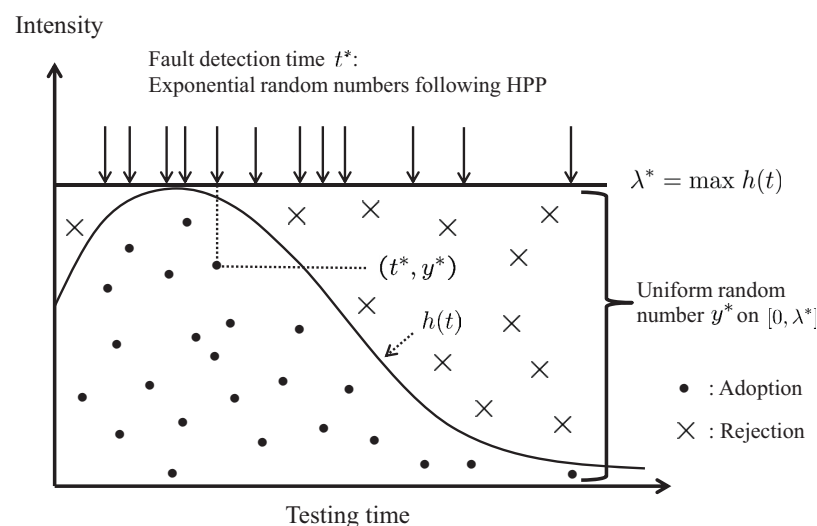


Figure 1. Conceptual figure of the thinning method.

The procedure is as follows: First, the maximum value of the intensity function of NHPP $h(t)$ is set to $\lambda^* = \max h(t)$. Furthermore, we generate an exponential random number, t^* , according to HPP as a candidate for the fault detection time. Next, we generate a uniform random number y^* on $[0, \lambda^*]$ and simulate the coordinate of the candidate point (t^*, y^*) . The candidate points in the upper region of the intensity function are rejected,

and those in the lower region are adopted. Note that candidate points are adopted if $\lambda^* \cdot y^* \leq h(t^*)$. Each adopted time candidate point is used as the sample data of the fault detection time.

3.2. Sample Data Generation of Fault Detection Time

To determine the intensity functions, we estimated parameters of SRGMs using the following fault-counting data [8,9] observed in the actual testing process.

$$DS : (t_k, y_k) (k = 1, 2, \dots, 36 \quad t_{36} = 36, y_{36} = 290)$$

In the above data, t_k and y_k are the calendar time and total detected faults, respectively. In this study, the DSS and ISS SRGMs were applied since the above fault-counting data are an S-shaped reliability growth curve data. The estimated values \hat{a} , \hat{b} , and \hat{l} of the parameters a , b , and l included in the DSS and ISS SRGMs are estimated using the maximum likelihood method. Furthermore, mean squared errors (MSE) were used as assessment criteria for the goodness-of-fit comparison. Here, MSE is defined as

$$MSE = \frac{1}{K} \sum_{k=1}^K (y_k - \hat{y}_k)^2, \tag{13}$$

where K is the total number of data, y_k is the actual value, and \hat{y}_k is the estimated value of the cumulative number of faults.

Figures 2 and 3 are the estimated DSS and ISS SRGMs. Table 1 presents the results of the parameter estimation and MSE. From the result of the goodness-of-fit comparison for DSS and ISS SRGMs, we observe that the ISS SRGM has better performance. The intensity functions estimated above were used to generate sample data of the fault detection time. Figures 4 and 5 show each piece of generated data using the thinning method, and each intensity function of the DSS and the ISS SRGMs. These data were called ‘‘DSS data’’ and ‘‘ISS data’’, respectively.

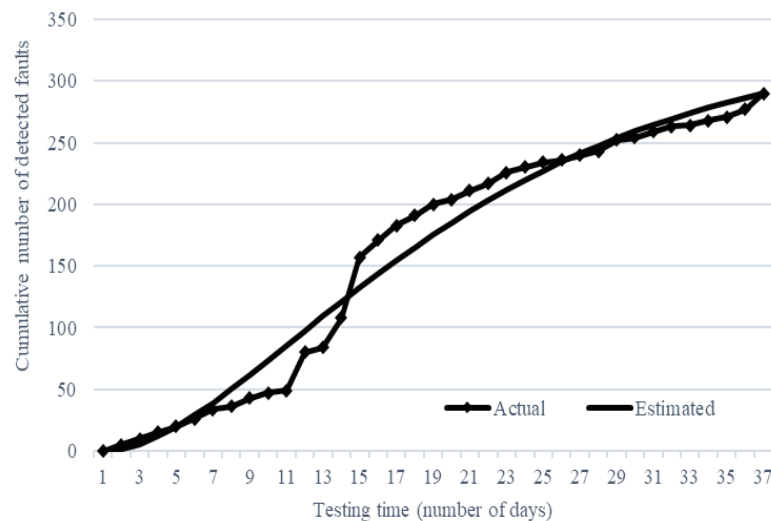


Figure 2. Estimated delayed S-shaped SRGM.

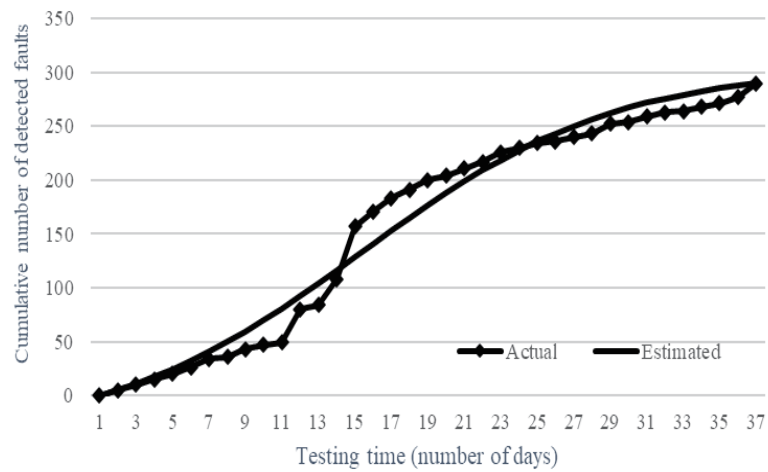


Figure 3. Estimated inflection S-shaped SRGM.

Table 1. Estimated parameter and MSE.

	\hat{a}	\hat{b}	\hat{t}	MSE
DSS	335.536	0.097	-	247.752
ISS	304.876	0.143	0.113	245.909

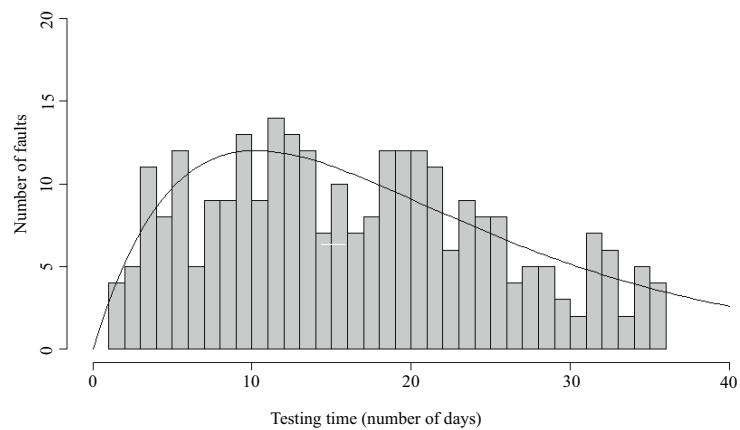


Figure 4. Generated sample data based on the intensity function of the DSS SRGM (DSS data).

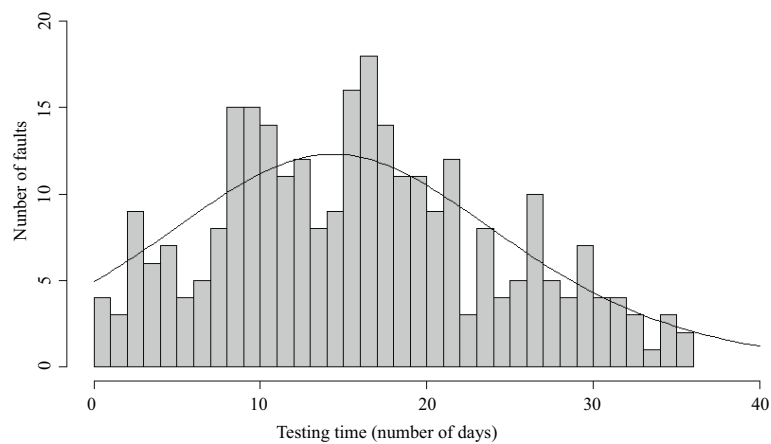


Figure 5. Generated sample data based on the intensity function of the ISS SRGM (ISS data).

4. Queuing Simulation Using Sample Data of Fault Detection Time

The infinite server queuing model is used for simulating fault correction activities. Detected faults enter the infinite server and the exit server when the correction is complete, as shown in Figure 6. Suppose that the n -th fault detection and correction times are represented by t_n and c_n , respectively, the correction completion time d_n is defined as the following equation:

$$d_n = t_n + c_n. \tag{14}$$

The number of faults corrected is obtained by counting the number of faults n which fills $t_n < t < d_n$ with arbitrary time t . Assuming that the correction time follows the exponential distribution, we clarify the busy fault correction activities from the fault detection to correction completion using the queuing simulation.

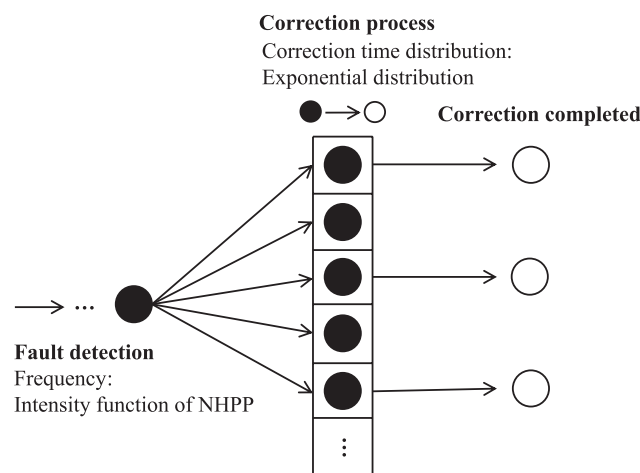


Figure 6. Conceptual figure of the infinite server queuing model.

Figure 7 shows the simulation result, assuming that the fault correction time follows the exponential distribution with an average of 0.5 days. Observe that the fault correction activities became the busiest around the 11th and 18th day, respectively. Figure 8 shows the simulation result, assuming that the fault correction time follows the exponential distribution with an average of 1 day. Here, the number of faults being corrected doubled. From Figures 7 and 8, no significant difference exists between the most fault detection time and the busiest fault correction time at the first peak. However, there is a large time difference at the second peak.

Figure 9 shows the simulation result, assuming that the fault correction time follows the exponential distribution with an average of 0.5 days. Here, the fault correction activities became the busiest around the 12th and 17th day, respectively. Figure 10 shows the simulation result assuming that the fault correction time follows the exponential distribution with an average of 1 day. Figures 9 and 10 show that fault correction activities are busy in the first peak due to factors such as the difficulty of detecting or correcting faults, rather than the number of detected faults.

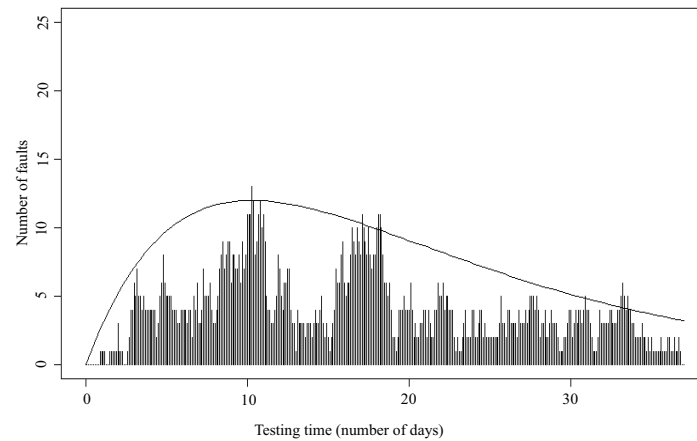


Figure 7. Behavior of the number of faults being corrected (the correction time is 0.5 day/fault on average, DSS data).

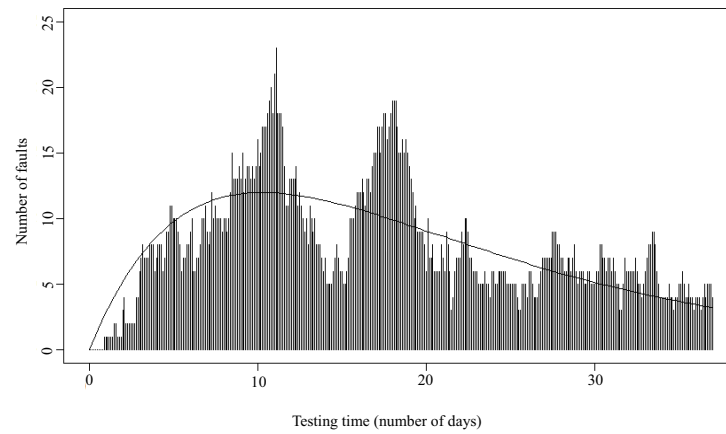


Figure 8. Behavior of the number of faults being corrected (the correction time is 1.0 day/fault on average, DSS data).

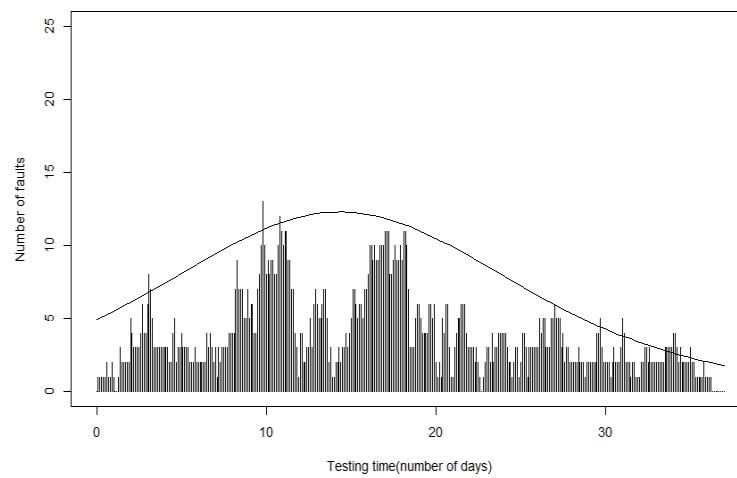


Figure 9. Behavior of the number of faults being corrected (the correction time is 0.5 day/fault on average, ISS data).

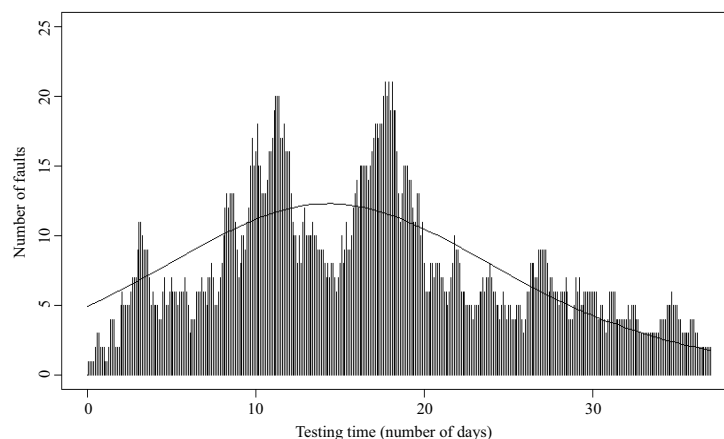


Figure 10. Behavior of the number of faults being corrected (the correction time is 1.0 day/fault on average, ISS data).

From the above, it clear that the fault correction activities became the busiest at the time slightly different from when the fault was most detected. Additionally, model selection must be according to the goodness-of-fit and characteristics of the model because of the changing behavior of the intensity function.

5. Conclusions

In this paper, we discussed a method for evaluating the efficiency of fault correction activities in a testing phase. The sample data of fault detection time were generated to obtain new development management knowledge from existing fault-counting data. The DSS and ISS SRGMs were adopted from the behavior of the actual data, and the parameters of each model were estimated using the maximum likelihood method. The intensity functions of the DSS and ISS SRGMs were determined from the estimated parameters, and the sample data were generated using the thinning method. Furthermore, we performed a simulation using the infinite server queuing model to quantitatively evaluate the efficiency of fault correction activities. Specifically, assuming that the correction time follows the exponential distribution, the time behavior of the number of faults being corrected was presented. Therefore, we obtained that there were time lags between the peaks for the number of fault detection and correction, respectively. As a future study, it is necessary to perform the simulation using the queuing models that can express the actual testing process more and confirm their effectiveness.

Author Contributions: Conceptualization, Y.M. (Yuka Minamino); methodology, Y.M. (Yuka Minamino), S.I.; programming, Y.M. (Yuka Minamino) and Y.M. (Yusuke Makita); validation, Y.M. (Yuka Minamino) and Y.M. (Yusuke Makita); investigation, Y.M. (Yuka Minamino) and Y.M. (Yusuke Makita); resources, S.I. and S.Y.; data curation, S.I.; writing—original draft preparation, Y.M. (Yuka Minamino) and Y.M. (Yusuke Makita); writing—review and editing, S.I. and S.Y.; visualization, Y.M. (Yuka Minamino) and Y.M. (Yusuke Makita); supervision, S.I. and S.Y.; project administration, Y.M. (Yuka Minamino); funding acquisition, Y.M. (Yuka Minamino). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JSPS KAKENHI Grant Number 20K14983.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pham, H. *System Software Reliability*; Springer: London, UK, 2006.
2. Zhu, M.; Pham, H. Software reliability modeling and methods: A state of the art review. In *Optimization Models in Software Reliability*; Aggarwal, A.G., Tandon, A., Pham, H., Eds.; Springer: Cham, Switzerland, 2022; pp. 1–29.
3. Yamada, S. *Software Reliability Modeling—Fundamentals and Applications*; Springer: Tokyo, Japan; Heidelberg, Germany, 2014.
4. Kapur, P.K.; Aggarwal, A.G.; Anand, S. A new insight into software reliability growth modeling. *Int. J. Perform. Eng.* **2009**, *5*, 267–274.
5. Kapur, P.K.; Anand, S.; Inoue, S.; Yamada, S. A unified approach for developing software reliability growth model using infinite server queueing model. *Int. J. Reliab. Qual. Saf. Eng.* **2010**, *17*, 401–424. [[CrossRef](#)]
6. Omi, T.; Nomura, S. *Time Series Analysis for Point Processes*; Kyoritsu-Publication: Tokyo, Japan, 2019. (In Japanese)
7. Kazuhira, O. An overview practical software reliability prediction. In *Reliability Modeling with Computer and Maintenance Applications*; Nakamura, S., Qian, C.H., Nakagawa, T., Eds.; World Scientific: Singapore, 2017; pp. 3–21.
8. Zhang, X.; Pham, H. Comparisons of nonhomogeneous Poisson process software reliability models and its applications. *Int. J. Syst. Sci.* **2000**, *31*, 1115–1123. [[CrossRef](#)]
9. Tohma, Y.; Yamano, H.; Ohba, M.; Jacoby, R. The estimation of parameters of the hypergeometric distribution and its application to the software reliability growth model. *IEEE Trans. Softw. Eng.* **1991**, *17*, 483–489. [[CrossRef](#)]