applied sciences

MDPI

Article

Default Detection Rate-Dependent Software Reliability Model with Imperfect Debugging

Ce Zhang [1,*], Wei-Gong Lv [1], Sheng Sheng [2], Jin-Yong Wang [3], Jia-Yao Su [1] and Fan-Chao Meng [1]

[1] School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai 264209, China
[2] Shenzhen Huantai Technology Co., Ltd., Shenzhen 518063, China
[3] School of Automation and Software Engineering, Shanxi University, Taiyuan 030006, China
* Correspondence: zhangce@hitwh.edu.cn

Abstract: From the perspective of FDR (fault detection rate), which is an indispensable component in reliability modeling, this paper proposes two kinds of reliability models under imperfect debugging. This model is a relatively flexible and unified software reliability growth model. First, this paper examines the incomplete phenomenon of debugging and fault repair and established a unified imperfect debugging framework model related to FDR, which is called imperfect debugging type I. Furthermore, it considers the introduction of new faults during debugging and establishes a unified imperfect debugging framework model that supports multiple FDRs, called imperfect debugging type II. Finally, a series of specific reliability models are derived by integrating multiple specific FDRs into two types of imperfect debugging framework models. Based on the analysis of the two kinds of imperfect debugging models on multiple public failure data sets, and the analysis of model performance differences from the perspective of fitting metrics and prediction research, a fault detection rate function that can better describe the fault detection process is found. By incorporating this fault detection rate function into the two types of imperfect debugging models, a more accurate model is obtained, which not only has excellent performance and is superior to other models but also describes the real testing process more accurately and will guide software testers to quantitatively improve software reliability.

Keywords: software reliability; software reliability growth model (SRGM); imperfect debugging (ID); fault detection rate (FDR)

check for updates

Citation: Zhang, C.; Lv, W.-G.; Sheng, S.; Wang, J.-Y.; Su, J.-Y.; Meng, F.-C. Default Detection Rate-Dependent Software Reliability Model with Imperfect Debugging. *Appl. Sci.* **2022**, *12*, 10736. https://doi.org/10.3390/app122110736

Academic Editor: Juan Pavón

Received: 27 June 2022
Accepted: 20 October 2022
Published: 23 October 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 testing is an important method to continuously improve software reliability. Currently, using the software reliability growth model SRGM to quantitatively study the improvement of software reliability during the test phase has been widely adopted [1]. To effectively measure and predict software reliability, the study is supposed to be based on accurate mathematical models. The software testing process is the unification of multiple complex stochastic processes, as shown in Figure 1. To assume this process, with several factors removed, the SRGM establishes a mathematical model from fault detection to repair and solves the cumulative number of faults $m(t)$ detected at a certain time t with the model. Furthermore, using the correlation between $m(t)$ and reliability $R(t)$, the model obtains $R(t)$ and further studies on the reliability $R(t)$.

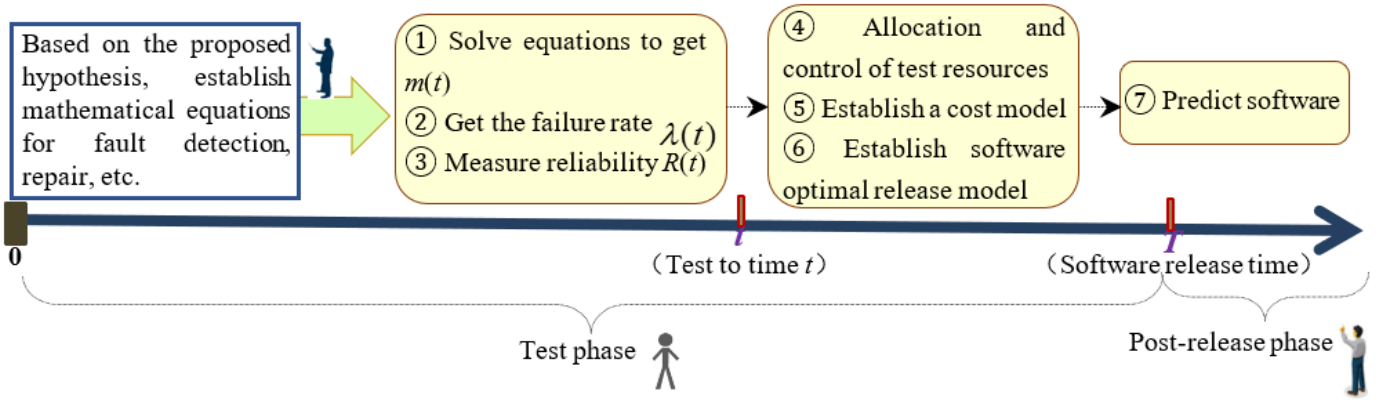


Figure 1. Research Ideas and Functions of SRGM.

It can be seen that in this essential period before software release, it is very important to establish a mathematical model that accurately describes the testing process to improve the reliability. Therefore, in SRGM research, obtaining $m(t)$ and then obtaining $R(t)$ by establishing an effective mathematical model is important.

As the test progresses, it is foreseeable that the faults in the software are constantly eliminated and the software reliability has a continuous growth. In fact, in addition to improving reliability, the SRGM can also be used to test resource allocation and management, test cost estimation and management, determine software release time, calculate metrics directly related to reliability, etc. Hence, studying reliability with SRGM plays an important role in reliability research.

The research on software reliability carried out through SRGM has gained the attention of many researchers. These researches on TE, fault reduction factor, release time and fault correlation [2–8] further deepen people's understanding of software reliability under the influence of various factors, and make positive contributions to the design and development of more reliable software.

The common software testing process includes many periods, such as designing test cases and repairing faults. This will be affected by many random factors. Therefore, it is a complex random process with imperfect phenomena. Incorporating this imperfect phenomenon [9–12] into reliability research results in the branch field of imperfect debugging. The current research on imperfect debugging mainly consists of two aspects: (i) the detected errors are not completely eliminated (which can be called incomplete debugging) [12,13] and (ii) new faults are produced in debugging [9–11]. In terms of the accumulated number of faults $a(t)$ in the software [14–16], imperfect debugging, such as introducing a new fault, will cause $a(t)$ to increase.

It is also a way to study the change of reliability from the perspective of $a(t)$ change. For example, in reference [17], such research has been carried out and positive progress has been made. Different from reference [17], it is more pertinent to study reliability from the perspective of $b(t)$. This is because $b(t)$ is a description of the entire test environment, which can reflect the impact of different test strategies on reliability, so it is more valuable.

Software reliability research considering imperfect debugging is an important branch of SRGM [17–20], because imperfect debugging is more consistent with real software testing and debugging process. Huang and Chen et al. [21] described the error volatility by using the sine periodic function, and described the trend that the impact of the increasing new errors will gradually weaken with the increasing of test time, and proposed a model considering imperfect troubleshooting and change points. Subhashis and Deepjyoti et al. [22] proposed a software reliability growth model considering imperfect troubleshooting, troubleshooting probability, random test environment and test coverage. From the perspective of adjusting R-squared and mean squared error indicators, the model has excellent performance. The imperfect troubleshooting model established in reference [19,20] incorporates new fault generation and change points, making software reliability more able to describe

the real situation in testing. From the perspective of test coverage, reference [23] established an imperfect troubleshooting software reliability model related to the test coverage function.

In general, These studies are from different perspectives and contents, but lack integrated research on imperfect debugging in the software testing process, thus can not accurately establish the quantitative relationship between fault detection, repair and introduction in the software testing process.

The purpose of the test process is to verify and confirm whether the software meets the requirements. It can help to find problems that do not meet the requirements during the testing process and inform the developer to modify them in time to improve the reliability, so as to meet the desired goals. Objectively speaking, due to the difference in testing environment and testing strategy adopted by testers, test indicators have differences in the software testing process, which is representatively reflected in FDR (fault detection rate). From the perspective of establishing a mathematical model, the difference between models has a close relation to FDR. FDR can macroscopically describe the fault detection ability in a testing environment, which makes it the major point in evaluating the performance of the SRGM. In addition to testing coverage and all faults in software, FDRF (FDR Function) $b(t)$ is an important factor affecting SRGM. Therefore, the study needs to analyze the influence of FDRF $b(t)$ on models or reliability.

This paper first develops a unified fault detection (including repair) model that establishes a single differential equation used in FDR studies. Then, the paper establishes another fault detection model including fault detection, repair and introduction and builds a system of differential equations used in FDR study. Through research on the performance of these two reliability models supported by FDR, it is possible to distinguish different FDR performances. In addition, the research verifies the importance of establishing a reliability model with abundant test information to enhance reliability. The method of software reliability process analysis with imperfect debugging proposed in this paper focuses on the influence of FDR. It can describe the software testing process more accurately. It is of great significance to improve software reliability.

In this paper, the structure is arranged as follows: Section 2 gives a general review of imperfect debugging and FDR; Section 3 gives the SRGM modeling process related to the fault detection rate under imperfect debugging, proposes two specific models of imperfect debugging frameworks, and obtains a unified cumulative fault detection function; and Section 4 gives the specific imperfect debugging models supported by five types of FDR. Section 5 verifies the validity and rationality of the proposed model through published failure data sets. Finally, the summary makes conclusions and points out the directions for future research of SRGM.

2. Imperfect Debugging and FDR

Current imperfect debugging contains incomplete debugging and/or introduces new faults as a whole. These simulate software testing in real life. Here are the following classifications:

- imperfect debugging: Consider the repair work in the test, but the repair is incomplete. There are some phenomena in which some faults that are not completely repaired may be detected in subsequent tests [9,12,24].

This situation can be interpreted as the difference between the experience and proficiency of the software tester and factors such as the environment of the debugging, which will lead to the possibility that the detected faults will not be completely repaired.

- Introducing new faults: Consider the repair work in testing, but the repair is incomplete, and new faults may be introduced in the repair process [25,26].

This situation can be interpreted as follows: in software testing, the detected faults will undergo reporting, diagnosis, isolation, repair (correction) and verification [9,27]. Due to

the developer's thoughtlessness, there may be the possibility of introducing new faults [28] in the debugging process.

In the current research of SRGM, FDR is in all kinds of models, and FDR has become an indispensable element in reliability research. FDR has the ability to describe fault detection in the overall test environment, and it has an important impact on fault detection, repair, etc., which will directly affect software reliability. Although some FDR functions have been proposed to directly participate in the establishment of reliability models, the performance of FDR has not yet been analyzed.

In summary, the incompleteness of debugging and the introduction of new faults are typical imperfect debugging. FDR is an important parameter of SRGM, but it still lacks comprehensive and in-depth research on the three in the current research.

3. Fault Detection Rate-Dependent SRGM with Imperfect Debugging

3.1. Basic Assumptions

Based on the recognition of the test environment, the assumptions of the imperfect debugging model considering the correlation of the fault detection rate are as follows [29–34]:

- The fault detection and repair process obeys the nonhomogeneous Poisson distribution process NHPP (nonhomogeneous Poisson process);
- Software failure is caused by the remaining faults in the software;
- In the time interval $(t, t + \Delta t)$, at most one fault occurs, detected faults are proportional to remaining faults, with a proportional coefficient of $p(t)$;
- In the time interval $(t, t + \Delta t)$, the repaired faults are proportional to the number of faults detected;
- In the process of fault repair, new faults are introduced. The number of introduced faults is proportional to the number of accumulated repaired faults, and the proportional coefficient is $R(t)$.

Based on the above assumptions, two types of imperfect debugging models can be established.

3.2. Imperfect Debugging Type I: Unified Fault Detection and Repair Framework Model

FDR is the result of the comprehensive application of testing technology in the software testing process. It can be modeled from the perspective of test coverage, can be set from the test workload, and can be set directly according to the actual situation. FDF is closely related to reliability modeling and measurement. In the current verifiable SRGM research, all models are based on the above assumptions To this end, a general fault detection and repair process model is proposed:

$$\frac{dm(t)}{dt} = b(t)[a(t) - p(t) \cdot m(t)] \tag{1}$$

In this paper, the number of faults detected at time t is considered to be proportional to the total number of faults remaining at present. The proportional coefficient is $b(t)$, and $p(t)$ indicates the probability of bug fixing, that is, the fault repair is incomplete.

Bring the initial conditions: $m(0) = 0$, $a(0) = a$, the homogeneous equation of (1) corresponding to (one-dimensional differential equation) is:

$$\frac{dm(t)}{dt} + b(t)p(t)m(t) = 0 \tag{2}$$

Its solution is

$$m(t) = Ce^{-\int_0^t b(t)p(t)dt}$$

Let

$$C = u(t)$$

$$m(t) = u(t)e^{-\int_0^t b(t)p(t)dt}$$

Substituting the above equation into Equation (2), the equation about $u(t)$ can be obtained as

$$u'(t) = \frac{b(t)a(t)}{e^{-\int_0^t b(t)p(t)dt}}$$

Substituting $u(t)$ for $u'(t)$ integral into the original equation, the equation about $m(t)$ can be obtained as

$$m(t) = Ce^{-\int_0^t b(u)p(u)du} + e^{-\int_0^t b(u)p(u)du} \int_0^t b(v)a(v)e^{\int_0^v b(s)p(s)ds}dv$$

For $m(0) = 0$, $C = 0$

$$m(t) = e^{-\int_0^t b(u)p(u)du} \int_0^t b(v)a(v)e^{\int_0^v b(s)p(s)ds}dv \tag{3}$$

Obviously, the multiple settings of $b(t)$, $p(t)$ and $a(t)$ in Equation (3) obtain multiple $m(t)$. Therefore, this paper presents a framework model with strong flexibility.

3.3. Imperfect Debugging Type II: Imperfect Debugging Framework Model Considering Fault Detection, Repair, and Introduction under Fault Detection Rate

Because of the complexity of the software structure, the programmer's repair is likely to damage the original structure of the program, especially in large software systems, which will introduce new faults. To this end, based on model (1) established above, from the perspective of fault detection, repair and introduction, a unified software test model considering imperfect debugging is proposed as follows:

Based on the five assumptions mentioned in Section 3.1, the system of Equation (4) can be obtained:

$$\begin{cases} \frac{dm(t)}{dt} = b(t) \cdot [a(t) - p(t) \cdot m(t)] \\ \frac{da(t)}{dt} = r(t) \cdot \frac{d(p(t) \cdot m(t))}{dt} \end{cases} \tag{4}$$

The second differential equation describes fault introduction, which is that the number of faults added at time t is proportional to the number of faults repaired at that time. Since the introduction of a new fault occurs during fault repair, rather than fault detection, faults added at time t are proportional to the faults repaired at time t. Equation (4) is modeled to describe the relationship between fault detection, repair and introduction. $m(t)$ and $a(t)$ are functions to be solved, $b(t)$, $p(t)$ and the fault introduction rate $R(t)$ is the description of the test level in the overall test environment.

$m(t)$ and $a(t)$ are characterized and determined by three parametric function functions $k(t)$, $p(t)$ and $R(t)$. Setting the three parameter functions reasonably, it is possible to obtain an accurate model of the actual test process. The above formula is beyond the solution of the conventional differential equations. In addition, the main focus here is on the impact of the FDR $k(t)$ on the model, so let $p(t) = p$.

The functions to be solved are $m(t)$ and $a(t)$, and the solution of Equation (4) is performed under the initial conditions of $m(0) = 0$ and $a(0) = a$.

The above formula is complicated to solve. To reduce the complexity, considering the convenience of the solution and not affecting the overall efficiency of the model, let $p(t) = p$. The boundary conditions of the above equation are $m(t) = 0$ and $a(t) = a$. Let

$$x(t) = a(t) - p \cdot m(t) \tag{5}$$

Simultaneously, differentiate on both sides

$$\frac{dx(t)}{dt} = \frac{da(t)}{dt} - p \cdot \frac{dm(t)}{dt} \tag{6}$$

Substituting the second equation of Equation (4) into Equation (6), under the condition that $p(t) = p$

$$\frac{dx(t)}{dt} = p \cdot r(t) \cdot \frac{dm(t)}{dt} - p \cdot \frac{dm(t)}{dt} = p \cdot (r(t) - 1) \cdot \frac{dm(t)}{dt} \tag{7}$$

Substituting the first equation of Equation (4) into Equation (7)

$$\frac{dx(t)}{dt} = p \cdot (r(t) - 1) \cdot b(t) \cdot (a(t) - p \cdot m(t)) \tag{8}$$

Substituting Equation (5) into Equation (8)

$$\frac{dx(t)}{dt} = p \cdot (r(t) - 1) \cdot b(t) \cdot x(t) \tag{9}$$

From the initial conditions, the equation $x(0) = a(0) - p \cdot m(0) = a$ can be obtained. Striving through differential equations, the solution of Equation (9) is obtained as

$$x(t) = ae^{-\int_0^t p \cdot (1 - r(\tau)) \cdot b(\tau) d\tau} \tag{10}$$

Substituting Equation (5) into the first equation of Equation (4)

$$\frac{dm(t)}{dt} = b(t) \cdot x(t) \tag{11}$$

Substituting Equation (10) into Equation (11), the solution of $m(t)$ can be obtained by integrating both sides of the equation simultaneously as

$$m(t) = \int_0^t b(u) \cdot x(u) du = a \int_0^t b(u) \cdot e^{-\int_0^u p \cdot (1 - r(\tau)) \cdot b(\tau) d\tau} du \tag{12}$$

Substituting Equation (12) into the second equation of Equation (4), the solution of $a(t)$ can be obtained as

$$a(t) = a\left(1 + p \cdot \int_0^t r(u) b(u) \cdot e^{-\int_0^u p \cdot (1 - r(\tau)) \cdot b(\tau) d\tau} du\right) \tag{13}$$

4. FDR-Related Reliability Model

Here, based on the two unified types of imperfect debugging framework models obtained above, the analysis is carried out according to the $b(t)$ proposed in the reference [32–35]. $b(t)$ is mainly set in the following five forms ($b1(t)$, $b2(t)$, $b3(t)$, $b4(t)$ and $b5(t)$). They are specific $b(t)$ functions, representing different failure detection rates. These $b(t)$ functions composed of different parameters depict the changing trend of fault detection rate under different test environments. Then, cumulative fault detection functions $m(t)$ are obtained in the following five cases:

(i) If $b_1(t) = b\alpha\beta e^{-\beta t}$, $m(t)$ can be obtained as follows:

$$m_{IDI_1}(t) = e^{-\int_0^t b\alpha\beta e^{-\beta u} p du} \int_0^t b\alpha\beta e^{-\beta v} a e^{\int_0^v b\alpha\beta e^{-\beta s} p ds} dv \tag{14}$$

$$m_{IDII_1}(t) = a \int_0^t b\alpha\beta e^{-\beta u} e^{-\int_0^u p(1 - r(\tau)) b\alpha\beta e^{-\beta\tau} d\tau} du \tag{15}$$

$m_{IDI_1}(t)$ and $m_{IDII_1}(t)$ represent the specific cumulative fault detection function $m(t)$ obtained by two kinds of imperfect debugging framework models with the participation of the same $b(t)$. IDI indicates imperfect debugging type one, and IDII means imperfect debugging type two. The following situation is the same.

At this point, as the test continues, $b_1(t \to \infty) = 0$.

(ii) If $b_2(t) = \frac{b}{1+\beta e^{-bt}}$, $m(t)$ can be obtained as follows:

$$m_{IDI_2}(t) = e^{-\int_0^t \frac{b}{1+\beta e^{-bu}} p du} \int_0^t \frac{b}{1+\beta e^{-bv}} a e^{\int_0^v \frac{b}{1+\beta e^{-bs}} p ds} dv \tag{16}$$

$$m_{IDII_2}(t) = a \int_0^t \frac{b}{1+\beta e^{-bu}} e^{-\int_0^u p(1-r(\tau)) \frac{b}{1+\beta e^{-b\tau}} d\tau} du \tag{17}$$

At this point, as the test continues, $b_2(t \to \infty) \to b$.

(iii) If $b_3(t) = b\alpha\beta t e^{-\beta t^2/2}$, the corresponding $m(t)$ is:

$$m_{IDI_3}(t) = e^{-\int_0^t b\alpha\beta u e^{-\beta u^2/2} p du} \int_0^t b\alpha\beta v e^{-\beta v^2/2} a e^{\int_0^v b\alpha\beta s e^{-\beta s^2/2} p ds} dv \tag{18}$$

$$m_{IDII_3}(t) = a \int_0^t b\alpha\beta u e^{-\beta u^2/2} e^{-\int_0^u p(1-r(\tau)) b\alpha\beta\tau e^{-\beta\tau^2/2} d\tau} du \tag{19}$$

At this point, as the test continues, $b_3(t \to \infty) = 0$.

(iv) If $b_4(t) = b^2 t/(1+bt)$, $m(t)$ can be obtained as follows:

$$m_{IDI_4}(t) = e^{-\int_0^t b^2 u/(1+bu) p du} \int_0^t b^2 u/(1+bu) a e^{\int_0^v b^2 s/(1+bs) p ds} dv \tag{20}$$

$$m_{IDII_4}(t) = a \int_0^t b^2 u/(1+bu) e^{-\int_0^u p(1-r(\tau)) b^2 u/(1+bu) d\tau} du \tag{21}$$

At this point, as the test continues, $b_4(t \to \infty) \to b$.

(v) If $b_5(t) = \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)t}}$, the corresponding $m(t)$ is:

$$m_{IDI_5}(t) = e^{-\int_0^t \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)u}} p du} \int_0^t \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)v}} a e^{\int_0^v \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)s}} p ds} dv \tag{22}$$

$$m_{IDII_5}(t) = a \int_0^t \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)u}} e^{-\int_0^u p(1-r(\tau)) \frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)\tau}} d\tau} du \tag{23}$$

At this point, as the test continues, $b_5(t \to \infty) \to b(1+\sigma)$.

5. Numerical Example

5.1. Model and Failure Data Set

Here, three sets of published failure data sets, DS1 [36], DS2 [37], and DS3 [35] were selected to verify the performance of the model, which have been widely used to explain the performance of the SRGM. At the same time, five models that consider imperfect debugging to participate in the comparison were selected, as shown in Table 1, to verify the validity and flexibility of the proposed model in fitting and predicting failure data. In these $m(t)$ models, a represents the total number of faults in the software, b represents the fault detection rate, and other parameters are the constituent elements of $b(t)$ (referring to the corresponding references).

Table 1. Participating models.

Model	Cumulative Fault Detection Quantity $m(t)$
M-1: Y-Exp [38]	$m(t) = \frac{ab\left(e^{\alpha t} - e^{-bt}\right)}{a+b}$; $b(t) = b$
M-2: Y-Lin [38]	$m(t) = a(1 - e^{-bt})(1 - \frac{\alpha}{b}) + \alpha at$; $b(t) = b$.
M-3: Pham Zhang IFD [14]	$m(t) = a\left[1 - e^{-bt}\left(1 + (b+d)t + bdt^2\right)\right]$; $b(t) = b$.
M-4: P-Z [35]	$m(t) = a(1+\beta t)\left[1 - e^{-\frac{b^2 t^2}{2(1+bt)}}\right]$; $b(t) = b$.
M-5: Ohba-Chou [39]	$m(t) = \frac{a}{1-r}\left[1 - e^{-(1-r)bt}\right]$; $b(t) = b$.
IDI Frame Series Model: IDI_1—IDI_5	$m(t) = e^{-\int_0^t k(u)p(u)du}\int_0^t k(v)p(v)e^{\int_0^v k(s)p(s)ds}dv$
IDII Frame Series Model: IDII_1—IDII_5	$m(t) = a\int_0^t k(u) \cdot e^{-\int_0^u p\cdot(1-\gamma(\tau))\cdot k(\tau)d\tau}du$

5.2. Comparative Standard

The fitting indicators are the mean square error (*MSE*), *Variance*, *RMS-PE*, *BMMRE* and *R-square* of the regression curve equation, and the predictive indicator is the relative error (*RE*).

$$MSE = \sum_{i=1}^{k} \frac{(y_i - m(t_i))^2}{k} \tag{24}$$

$$R - square = \frac{\sum_{i=1}^{k}[m(t_i) - \overline{y}]^2}{\sum_{i=1}^{k}[y_i - \overline{y}]^2},\ \overline{y} = \frac{1}{k}\sum_{i=1}^{k} y_i \tag{25}$$

$$RE = \frac{m(t_q) - q}{q} \tag{26}$$

$$Variance = \sqrt{\frac{\sum_{i=1}^{k}(y_i - m(t_i) - Bias)^2}{k-1}},\ Bias = \frac{\sum_{i=1}^{k}[m(t_i) - y_i]}{k} \tag{27}$$

$$RME - PE = \sqrt{Bias^2 + Variance^2} \tag{28}$$

$$BMMRE = \frac{1}{k} \cdot \sum_{i=1}^{k}\left|\frac{m(t_i) - y_i}{\min(m(t_i), y_i)}\right| \tag{29}$$

where y_i represents the number of cumulative failures at t_i, $m(t_i)$ represents the estimated values at t_i, and k represents the number of failure data samples. Obviously, the smaller the values of *MSE*, *Variance*, *RMS-PE* and *BMMRE* are, the closer the *R-square* value is to 1, the better the fitting effect. The more *RE* approaches 0, the better the model predicts.

5.3. Performance Verification

First, the model in Table 1 is fitted on three failure data sets. Based on the parameters obtained after fitting (limited to length, not listed, can be obtained by the author), there is a brief analysis of the performance of the model.

5.3.1. Performance Analysis of the FDR-Related Imperfect Debugging Model

To verify the difference in FDRF, the experiment will be verified and observed in the type I model that only considers incomplete debugging. In the type I imperfect debugging model, the parameters are $a(t)$, p and FDRF, so it is easy to analyze the performance of FDR. The type I imperfect debugging model, namely, the IDI model, is tested on three data sets. Figure 2 shows the fitting curve of the IDI model.

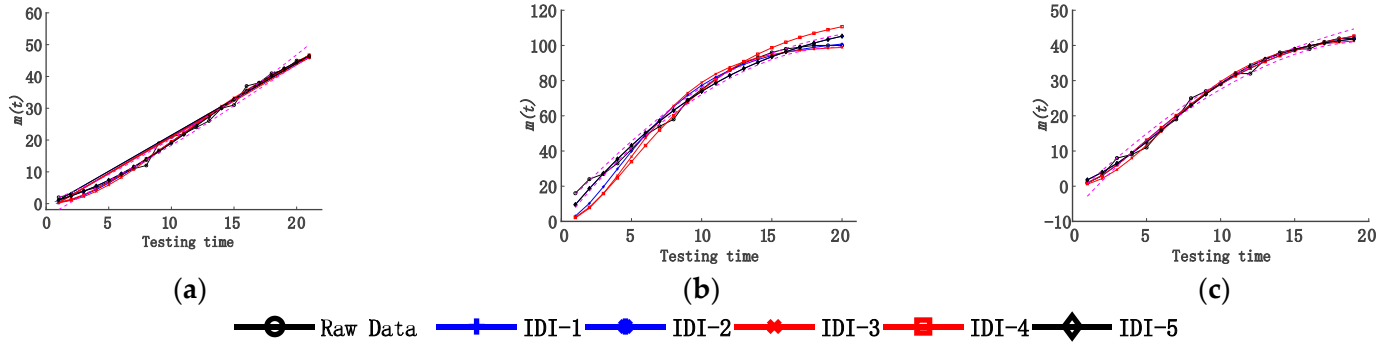


Figure 2. IDI Model Fitting Curve. (**a**) IDI Model Fitting Curve on DS1; (**b**) IDI Model Fitting Curve on DS2; (**c**) IDI Model Fitting Curve on DS3.

The first model is a unified incomplete debugging model. As shown directly from Figure 2, these models have the same curve shape on three data sets, which reflects that five different FDRs have no substantial impact on the essential trend of the curve.

To distinguish the performance of different models, especially to determine the performance differences of different FDRs, Table 2 gives the metrics of the model.

Table 2. Performance indicators of IDI-type models.

Model	DS	*MSE*	*R-Square*	*Variation*	*RMS-PE*	*BMMRE*
IDI_1	DS1	1.511895	1.045318	1.335027	1.357991	0.382202
IDI_2		1.194023	1.001295	1.121184	1.121655	0.092767
IDI_3		1.860862	1.071312	1.546524	1.590827	0.524525
IDI_4		1.431813	1.035174	1.274901	1.289997	0.329218
IDI_5		1.194023	1.001209	1.121203	1.12168	0.092741
IDI_1	DS2	25.64698	1.211092	5.779541	5.952458	0.325066
IDI_2		9.980156	1.047866	3.264629	3.272011	0.078299
IDI_3		39.52952	1.301613	7.372471	7.641251	0.550163
IDI_4		51.54921	1.528432	7.635519	7.718815	0.491672
IDI_5		9.980156	1.047869	3.264629	3.272011	0.078299
IDI_1	DS3	0.968407	0.99874	1.011346	1.011442	0.057902
IDI_2		0.934999	0.980322	0.998502	1.000092	0.095439
IDI_3		1.425435	1.060463	1.299476	1.321645	0.127577
IDI_4		0.9799	0.995025	1.017026	1.017026	0.057254
IDI_5		0.934999	0.98032	0.998502	1.000092	0.09544

From Table 2, it can be seen that IDI_2 and IDI_5 are almost the same in five standards, which is obviously superior to the other three models. This shows that the FDR (i.e., $b2(t)$ and $b5(t)$) adopted by IDI_2 and IDI_5 are more in line with the real fault detection, and its performance is better than the other three FDR models. In fact, the two $b(t)$ functions in IDI_2 and IDI_5 have the same structure in form. They all show the S-shaped trend of change and can adapt to various changes in software testing. Therefore, the cumulative fault detection function $m(t)$ corresponding to IDI_2 and IDI_5 has similar performance.

Then, Figure 3 shows the fitting results of the type II imperfect debugging model on three failure data sets.

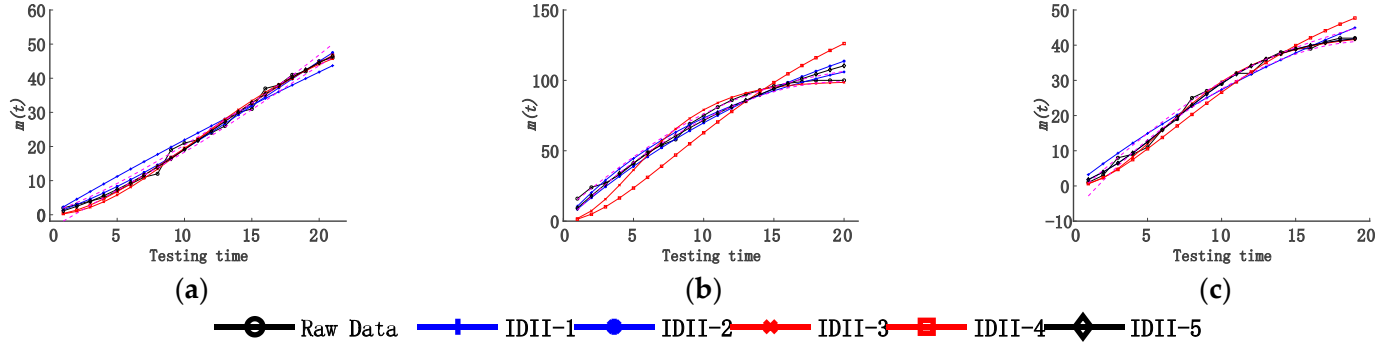


Figure 3. IDII Model Fitting Curve. (**a**) IDII Model Fitting Curve on DS1; (**b**) IDII Model Fitting Curve on DS2; (**c**) IDII Model Fitting Curve on DS3.

It can also be seen that the five models of type II imperfect debugging have the same curve shape, but the distance between the curves is larger than that of type I ID. The reason is that the phenomenon of new faults is considered in the modeling of the type II ID model. A set of differential equations is used to model and describe the model, and the final solution is obtained. The cumulative fault detection function $m(t)$ is quite different from each other. To further distinguish the performance differences among the five models in the type II imperfect debugging, Table 3 gives the fitting metrics of the five models.

Table 3. Performance comparison of IDII-type models.

Model	DS	*MSE*	*R-Square*	*Variation*	*RMS-PE*	*BMMRE*
IDII_1	DS1	8.117125	0.739869	3.414002	3.55666	0.232516
IDII_2		1.654242	0.955286	1.329353	1.332958	0.095028
IDII_3		1.975284	1.076242	1.60773	1.657388	0.544859
IDII_4		1.431813	1.035186	1.274885	1.289977	0.329222
IDII_5		1.195144	0.998241	1.120726	1.120886	0.091808
IDII_1	DS2	12.34467	0.996335	3.607085	3.607817	0.078214
IDII_2		31.55117	1.243937	6.084704	6.183099	0.118332
IDII_3		40.92279	1.313284	7.530967	7.812453	0.568511
IDII_4		211.9678	2.033129	17.18076	17.83243	1.040678
IDII_5		9.980156	1.047863	3.264631	3.272013	0.078299
IDII_1	DS3	4.558851	0.860654	2.315218	2.352301	0.261431
IDII_2		0.934999	0.980322	0.998502	1.000092	0.09544
IDII_3		1.186456	1.044917	1.170387	1.186124	0.11259
IDII_4		7.295519	1.249245	2.868974	2.898006	0.16295
IDII_5		0.934999	0.980321	0.998502	1.000092	0.09544

As Table 3 shows, model IDII_5 is either optimal or suboptimal in each performance index and has the best overall performance. Model IDII_5 has the same modeling process as the other four similar models, but the difference lies only in the use of $b5(t)$, which proves that $b5(t)$ has better performance than the other four FDR functions.

In summary, through the validation of two kinds of imperfect debugging models on failure data sets, the experimental results show that the model with $b5(t)$ participation has the best performance, which is superior to other models of the same kind, and both kinds of imperfect debugging models adopt their own unified models, so it can be concluded that $b5(t)$ is more accurate in describing the number of fault detections.

Thus far, the optimal models of two kinds of imperfect debugging models were obtained, namely, IDI_5 (the performance of IDI_2 is very close to that of IDI_5 because they have the same modeling process and the same FDR function with the same results) and IDII_5. These two optimal models are all $b5(t)$-related models.

5.3.2. Performance Analysis of the Type II Imperfect Debugging Model

It is obvious that the type II imperfect debugging model establishes the mathematical relationship between fault detection, repair and introduction and considers more random factors in the test. To this end, the experiment will be conducted on the performance of the type II imperfect debugging model. The comparisons of the curves of all the models in Table 1 on the failure data are shown graphically in Figure 4.

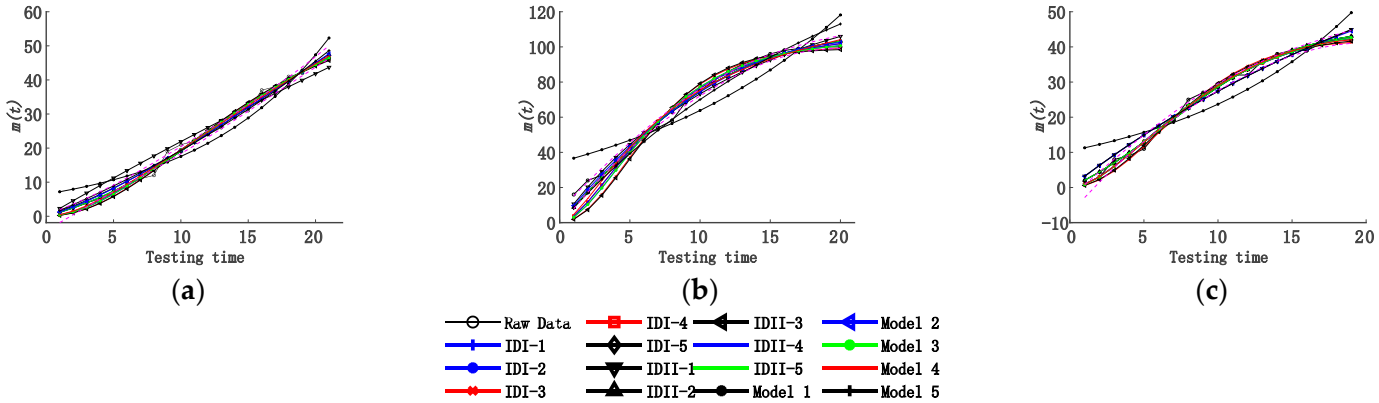


Figure 4. Fitting Curves of Models. (**a**) Fitting Curves of Models on DS1; (**b**) Fitting Curves of Models on DS2; (**c**) Fitting Curves of Models on DS3.

As seen directly from Figure 4, only several models have serious deviation compared with raw data, which makes it impossible to describe the trend of the fault detection quantity. To distinguish the performance between the models, Table 4 gives the fitting performance indicators for M-1~5 models in Table 1. The fitting performance indicators for the type I imperfect debugging models and the type II imperfect debugging models can be referred to Tables 2 and 3.

Table 4. Comparison of Fitting Performance of M-1~5 Models.

Model	DS	*MSE*	*R-Square*	*Variation*	*RMS-PE*	*BMMRE*
M-1	DS1	12.99494	0.838505	3.78606	3.81486	0.431939
M-2		1.642027	0.964688	1.32519	1.329018	0.103872
M-3		1.480686	1.042056	1.313155	1.333504	0.365081
M-4		1.417172	1.022353	1.245078	1.252981	0.276839
M-5		2.516028	0.938045	1.679348	1.696125	0.116178
M-1	DS2	124.2525	0.743409	11.48903	11.50564	0.22644
M-2		11.61711	1.001203	3.50065	3.501827	0.077979
M-3		25.25638	1.211331	5.739474	5.912213	0.319313
M-4		18.92156	1.120702	4.707464	4.782304	0.195724
M-5		11.61711	1.001135	3.500646	3.501821	0.077979
M-1	DS3	29.02455	0.720307	5.592244	5.610177	0.894779
M-2		4.474233	0.858111	2.302146	2.341391	0.260841
M-3		0.975441	1.001077	1.015523	1.01578	0.065226
M-4		1.305707	0.969718	1.183524	1.18652	0.068236
M-5		4.474233	0.858111	2.302155	2.341403	0.260842

5.4. Performance Comparison between the Type II Imperfect Debugging Model and the Type I Imperfect Debugging Model

Here, with the comparison of representatives of the two types, IDII_5 and IDI_5, it can be found that:

- On DS1, IDII_5 has three indicators (*Variation*, *RMS-PE*, *BMMRE*) that are optimal, one indicator (*MSE*) is suboptimal, and one indicator (*R-square*) is very close to its optimal indicator;
- On DS2, IDII_5 has one indicator (*MSE*) that is optimal, and the other four indicators (*R-square*, *Variation*, *RMS-PE*, *BMMRE*) are suboptimal, and four indicators differ from the optimal value only in the fifth or sixth decimal places.
- On DS3, IDII_5 has three indicators (*MSE*, *variation*, *RMS-PE*) that are optimal. The *R-square* differs from the optimal value only between 0.01 and 0.02, and the *BMMRE* differs from the optimal value between 0.03 and 0.04.

Therefore, it can be seen that IDII_5 is slightly better than IDI_5, which means that type II considering the ID and the introduction of a new fault is superior to the type I model, which only considers ID. This is because the imperfect model that only considers imperfect debugging ignores the phenomena of the introduction of a new fault in debugging and loses important random factors in the real software debugging process, which causes distortion of the function.

5.5. Comparison of the Type II Imperfect Debugging Model and Other Models

According to the above analysis, it can be found that the performance of IDII_5 is significantly better than that of the M1~5 models, showing a good advantage.

The type II unified imperfect debugging model proposed in this paper quantitatively describes the fault detection, repair and introduction relationship through two differential equations, which can more accurately model the whole test process of software, fully considering the imperfections of debugging and the introduction of new faults.

Furthermore, the predicted *RE* curves of all models on the failure data had been plotted, as shown in Figure 5.

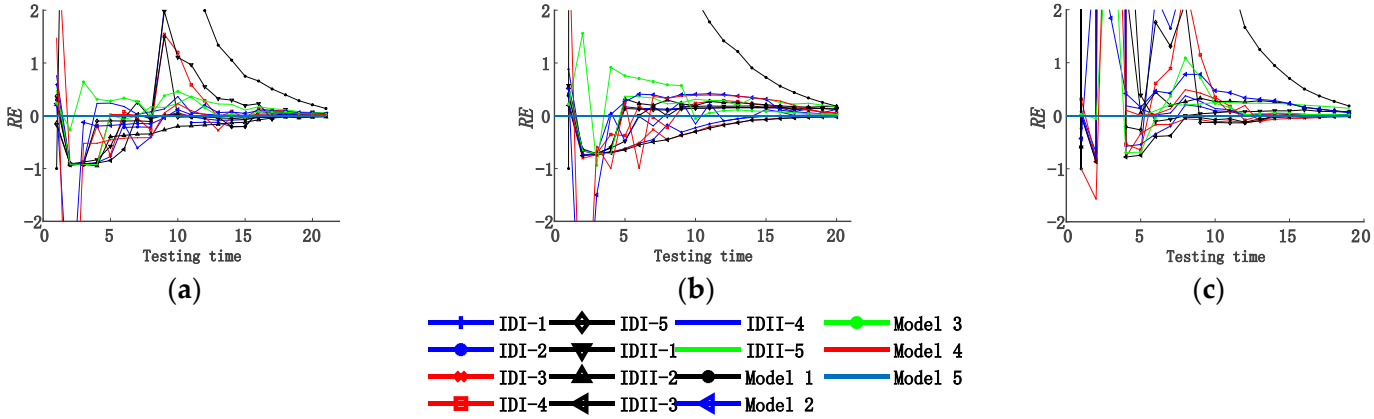


Figure 5. Prediction Curves of Models. (**a**) Prediction Curves of Models on DS1; (**b**) Prediction Curves of Models on DS2; (**c**) Prediction Curves of Models on DS3.

In the *RE* prediction curve of Figure 5, it can be found that the type II imperfect debugging model IDII_5 can quickly move closer to the zero-standard curve, showing excellent prediction performance.

Based on the comparison of the graphs of Figures 4 and 5 and the data of Table 4, it is clear that IDII_5 f type II considers not only the imperfect debugging but also introducing new faults, which is better than other models in Table 1.

In summary, this paper proposes two types of ID models. In particular, fully considering FDR, the type II imperfect debugging model fits the failure data better and has better

prediction performance than other models, which indicates that the modeling is capable of better describing the real test process. In addition, it can be seen that $b5(t)$ with an *S*-shaped trend shape has better performance than other fault detection rate functions, which can better describe the change in FDR during software testing.

6. Conclusions and Future Research

Based on the classification of imperfect debugging, this paper proposes two types of imperfect debugging models and establishes a unified imperfect debugging framework model related to FDR. With the experiment and analysis on the real and open failure data set, the validity of the model is verified. Considering the *S*-type FDR function, imperfect debugging and the introduction of new faults, this imperfect debugging model has a good fit and predictive performance, which is superior to other models. Considering more random factors in the test process, the established model contains rich information and has strong adaptability, which is more suitable for describing the actual test process. With the increasing random factors in software testing, the SRGM will become increasingly complicated and difficult to solve. Therefore, nonparametric solving methods can be used in subsequent research (such as artificial neural networks, deep learning, etc.).

Author Contributions: Conceptualization, C.Z. and W.-G.L.; methodology, S.S.; software, J.-Y.W.; validation, C.Z., W.-G.L. and J.-Y.W.; formal analysis, F.-C.M.; investigation, J.-Y.W.; resources, S.S.; data curation, S.S.; writing—original draft preparation, W.-G.L.; writing—review and editing, J.-Y.S.; visualization, S.S.; supervision, J.-Y.S.; project administration, C.Z.; funding acquisition, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: Shandong Province Natural Science Foundation (ZR2021MF067); Weihai Science and Technology Development Program (ITEAZMZ001807); Shanxi Province Basic Research Program (201801D121120).

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

References

1. Almering, V.; van Genuchten, M.; Cloudt, G.; Sonnemans, P.J. Using Software Reliability Growth Models in Practice. *IEEE Softw.* **2007**, *24*, 82–88. [CrossRef]
2. Yadav, S.S.; Kumar, A.; Johri, P.; Singh, J.N. Testing effort-dependent software reliability growth model using time lag functions under distributed environment. *Syst. Assur.* **2022**, 85–102. [CrossRef]
3. Nagaraju, V.; Wandji, T.; Fiondella, L. Improved algorithm for non-homogeneous poisson process software reliability growth models incorporating testing-effort. *Int. J. Perform. Eng.* **2019**, *15*, 1265–1272. [CrossRef]
4. Pradhan, V.; Kumar, A.; Dhar, J. Modelling software reliability growth through generalized inflection S-shaped fault reduction factor and optimal release time. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2022**, *236*, 18–36. [CrossRef]
5. Peng, R.; Ma, X.; Zhai, Q.; Gao, K. Software reliability growth model considering first-step and second-step fault dependency. *J. Shanghai Jiaotong Univ. (Sci.)* **2019**, *24*, 477–479. [CrossRef]
6. Kim, Y.S.; Song, K.Y.; Pham, H.; Chang, I.H. A software reliability model with dependent failure and optimal release time. *Symmetry* **2022**, *14*, 343. [CrossRef]
7. Munde, A. An empirical validation for predicting bugs and the release time of open source software using entropy measures—Software reliability growth models. *Syst. Assur.* **2022**, 41–49.
8. Haque, M.A.; Ahmad, N. An effective software reliability growth model. *Saf. Reliab.* **2021**, *40*, 209–220. [CrossRef]
9. Kapur, P.K.; Pham, H.; Anand, S.; Tadav, K. A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation. *IEEE Trans. Reliab.* **2011**, *60*, 331–340. [CrossRef]
10. Singh, O.; Kapur, R.; Singh, J. Considering the effect of learning with two types of imperfect debugging in software reliability growth modeling. *Commun. Dependability Qual. Manag.* **2010**, *13*, 29–39.
11. Kumar, D.; Kapur, R.; Sehgal, V.K.; Jha, P.C. On the development of software reliability growth models with two types of imperfect debugging. *Commun. Dependability Qual. Manag.* **2007**, *10*, 105–122.
12. Kapur, P.K.; Shatnawi, O.; Aggarwal, A.G.; Kumar, R. Unified Framework for Development Testing Effort Dependent Software Reliability Growth Models. *Wseas Trans. Syst.* **2009**, *8*, 521–531.
13. Goseva-Popstojanova, K.; Trivedi, K. Failure correlation in software reliability models. *IEEE Trans. Reliab.* **2000**, *49*, 37–48. [CrossRef]
14. Pham, H. *System Software Reliability*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2006.

15. Huang, C.-Y.; Kuo, S.-Y.; Lyu, M.R. An Assessment of Testing-Effort Dependent Software Reliability Growth Models. *IEEE Trans. Reliab.* **2007**, *56*, 198–211. [CrossRef]
16. Ahmad, N.; Khan, M.; Rafi, L. A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging. *Int. J. Qual. Reliab. Manag.* **2010**, *27*, 89–110. [CrossRef]
17. Zhang, C.; Yuan, Y.; Jiang, W.; Sun, Z.; Ding, Y.; Fan, M.; Li, W.; Wen, Y.; Song, W.; Liu, K. Software Reliability Model Related to Total Number of Faults Under Imperfect Debugging. *Adv. Intell. Autom. Soft Comput.* **2022**, *80*, 48–60.
18. Aggarwal, A.G.; Gandhi, N.; Verma, V.; Tandon, A. Multi-Release software reliability growth assessment: An approach incorporating fault reduction factor and imperfect debugging. *Int. J. Math. Oper. Res.* **2019**, *15*, 446–463. [CrossRef]
19. Saraf, I.; Lqbal, J. Generalized multi-release modelling of software reliability growth models from the perspective of two types of imperfect debugging and change point. *Qual. Reliab. Eng. Int.* **2019**, *35*, 2358–2370. [CrossRef]
20. Saraf, I.; Lqbal, J. Generalized software fault detection and correction modeling framework through imperfect debugging, error generation and change point. *Int. J. Inf. Technol.* **2019**, *11*, 751–757. [CrossRef]
21. Huang, Y.S.; Chiu, K.C.; Chen, W.M. A software reliability growth model for imperfect debugging. *J. Syst. Softw.* **2022**, *188*, 111267. [CrossRef]
22. Chatterjee, S.; Saha, D.; Sharma, A.; Verma, Y. Reliability and optimal release time analysis for multi up-gradation software with imperfect debugging and varied testing coverage under the effect of random field environments. *Ann. Oper. Res.* **2022**, *312*, 65–85. [CrossRef]
23. Zhang, C.; Lv, W.; Qiu, Z.; Gao, T.; Jiang, W.; Meng, F. Testing coverage software reliability model under imperfect debugging. *J. Hunan Univ. Nat. Sci.* **2021**, *48*, 26–35.
24. Chin, Y.; Huang, W. Software reliability analysis and measurement using finite and infinite server queueing models. *IEEE Trans Reliab.* **2008**, *57*, 192–203. [CrossRef]
25. Xie, M.; Yang, B. A study of the effect of imperfect debugging on software development cost. *IEEE Trans. Softw. Eng.* **2003**, *29*, 471–473.
26. Shyur, H.J. A stochastic software reliability model with imperfect-debugging and change-point. *J. Syst. Softw.* **2003**, *66*, 135–141. [CrossRef]
27. Wu, Y.P.; Hu, Q.P.; Xie, M.; Ng, S.H. Modeling and Analysis of Software Fault Detection and Correction Process by Considering Time Dependency. *IEEE Trans. Reliab.* **2007**, *56*, 629–642. [CrossRef]
28. Xie, M.; Hu, Q.P.; Wu, Y.P.; Ng, S.H. A study of the modeling and analysis of software fault-detection and fault-correction processes. *Qual. Reliab. Eng. Int.* **2007**, *23*, 459–470. [CrossRef]
29. Goel, L.; Okumoto, K. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Reliab.* **1979**, *28*, 206–211. [CrossRef]
30. Huang, C.Y.; Lyu, M.R.; Kuo, S.Y. A Unified Scheme of Some Nonhomogenous Poisson Process Models for Software Reliability Estimation. *IEEE Trans. Softw. Eng.* **2003**, *29*, 261–269. [CrossRef]
31. Hsu, C.J.; Huang, C.Y.; Chang, J.R. Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor. *Appl. Math. Model.* **2011**, *35*, 506–521. [CrossRef]
32. Pham, H. Software reliability and cost models: Perspectives, comparison, and practice. *Eur. J. Oper. Res.* **2003**, *149*, 475–489. [CrossRef]
33. Chiu, K.C.; Huang, Y.S.; Lee, T.Z. A study of software reliability growth from the perspective of learning effects. *Reliab. Eng. Syst. Saf.* **2008**, *93*, 1410–1421. [CrossRef]
34. Pham, H.; Nordmann, L.; Zhang, X. A general imperfect-software-debugging model with S-shaped fault-detection rate. *IEEE Trans. Reliab.* **1999**, *48*, 169–175. [CrossRef]
35. Pham, H.; Zhang, X.M. NHPP software reliability and cost models with testing coverage. *Eur. J. Oper. Res.* **2003**, *145*, 443–454. [CrossRef]
36. Pham, H. *Software Reliability*; Springer: Singapore, 2000.
37. Wood, A. Predicting software reliability. *Computer* **1996**, *29*, 69–77. [CrossRef]
38. Yamada, S.; Tokuno, K.; Osaki, S. Imperfect debugging models with fault introduction rate for software reliability assessment. *Int. J. Syst. Sci.* **1992**, *23*, 2241–2252. [CrossRef]
39. Ohba, M.; Chou, X.M. Does imperfect debugging affect software reliability growth? In Proceedings of the 11th International Conference on Software Engineering, Pittsburgh, PA, USA, 15–18 May 1989; pp. 237–244.