Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 2476584, 13 pages
http://dx.doi.org/10.1155/2016/2476584

Research Article

Software Reliability Growth Model with Partial Differential Equation for Various Debugging Processes

Jiajun Xu and Shuzhen Yao

School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Correspondence should be addressed to Jiajun Xu; xujiajun@buaa.edu.cn

Received 19 September 2015; Revised 8 December 2015; Accepted 20 December 2015

Academic Editor: Jean-Christophe Ponsart

Most Software Reliability Growth Models (SRGMs) based on the Nonhomogeneous Poisson Process (NHPP) generally assume perfect or imperfect debugging. However, environmental factors introduce great uncertainty for SRGMs in the development and testing phase. We propose a novel NHPP model based on partial differential equation (PDE), to quantify the uncertainties associated with perfect or imperfect debugging process. We represent the environmental uncertainties collectively as a noise of arbitrary correlation. Under the new stochastic framework, one could compute the full statistical information of the debugging process, for example, its probabilistic density function (PDF). Through a number of comparisons with historical data and existing methods, such as the classic NHPP model, the proposed model exhibits a closer fitting to observation. In addition to conventional focus on the mean value of fault detection, the newly derived full statistical information could further help software developers make decisions on system maintenance and risk assessment.

1. Introduction

Software reliability, defined as the probability of failure-free operation under certain conditions and by specific time [1], is one of the significant attributes of software systems development life cycle. As the software systems mature with ever growing complexity, evaluation, prediction, and improvement of their reliability become a crucial and daunting task for developers in both the development and testing phases. Numerous Software Reliability Growth Models (SRGMs) have been developed [2–5], which generally agree that the fault debugging process is a Nonhomogeneous Poisson Process (NHPP) under various diverging assumptions: perfect and imperfect debugging phenomenon [6, 7].

A common assumption is that removing detected faults will not introduce new faults, usually called perfect debugging phenomenon. Based on earlier works of Jelinski and Moranda [8], Goel and Okumoto [9] developed the exponential Software Reliability Growth Models (G-O model) with a constant fault detection rate. In latter models, the smoothly changeable fault detection rates are incorporated. Yamada et al. [10] found the fault detection rate fit to a S-shape growth curve and proposed the delay S-shaped SRGM. Employing the learning phenomenon in software fault detection process, Ohba [11] later developed an alternative inflection S-shaped SRGM. Further work was conducted by Yamada and others [12] by incorporating the relationship between the work effort and the number of faults detected into the model. Huang et al. [13] stated that fault detection and repair process are different in software development and operation. They proposed a unified theory to merge multiple moves into a NHPP-based SRGM and found a regularly changeable fault detection rate in the detection process. Later on Huang and Lyu [6] also used multiple change points to deal with imperfect debugging problems. The common assumption is that the removal of the detected faults will not introduce new faults. This scenario is known as perfect debugging. However, given the complexity of the software debugging process, debugging activity may be imperfect in practical software development environment and thus those perfect models may oversimplify the underlying dynamics.

To solve such problems, imperfect debugging processes are incorporated in latter models. Whenever the software system is developed, new errors can be introduced into the

software during development/testing phase and faults may not be corrected perfectly. This phenomenon is known as imperfect debugging. According to the above two phenomenons, SRGMs can be divided into two categories: perfect and imperfect debugging model. Kapur and Garg [14] discussed their fault removal phenomenon that as the team gains more experience, they may remove detected faults without causing new ones. However, the testing team may not correct a fault perfectly and the original fault may remain or generate new faults, leading to a phenomenon known as imperfect debugging. It was Goel [15] who first introduced the concept of imperfect debugging. Latter, Ohba and Chou [16] developed an error generation model based on the G-O model, named as an imperfect debugging model. Pham et al. [17] proposed a general imperfect software debugging model (P-N-Z model). Pham [18] also developed an SRGM for multiple failure types incorporating error generation. Zhang et al. [19] proposed a testing efficiency model that includes both imperfect debugging and error generation. Kapur et al. [20] proposed a flexible SRGM with imperfect debugging and error generation using a logistic function for the fault detection rate, which reflects the efficiency of the testing and removal team. Employing the learning phenomenon in software fault detection process, Yamada et al. [21] later developed an imperfect debugging model. In imperfect software debugging models, the fault introduction rate per fault is generally assumed to be constant or decreasing changes over time [22]. Recently, Wang et al. [22] also proposed a model to represent the imperfect debugging process with a log-logistic distribution. To sum up, most models presume a deterministic relationship between the cumulative number of detected faults and the time span of the software fault debugging process.

However, software debugging is a stochastic and uncertain process, which can be effected by many environmental factors, such as the distribution of resources, strategy, and running environment [23]. In particular, in the uncertain network environment, the aforementioned assumptions of deterministic model would become problematic. The environmental noise introduces great uncertainties that significantly affects traditional debugging process. To address such challenges, new stochastic models [24–27] were proposed, which treat the debugging process as perfect and stochastic; they assume each failure occurrence is independent and follows identical random distribution. By employing the white noise, that is, temporally uncorrelated random variables, for the environmental factors collectively, they used a flexible stochastic differential equation to model the irregular changes. Comparing to conventional models, such white-noise-based approach assumes the perfect debugging and no doubt provides closer approximation to uncertain fluctuations in reality but with great mathematical simplicity. Debugging activity is usually imperfect in piratical software development and recent data [28, 29] show that the fault detection is highly susceptible to noise and is generally correlated; thus earlier assumptions become problematic. Thus, because of its mathematical simplicity, it may also considerably underestimate the imperfect debugging process and the temporal correlation in a dynamic environment.

In this study, we propose an alternative stochastic framework based on partial differential equation (PDE), to describe the perfect/imperfect debugging processes, in which the environmental noise is of arbitrary correlation. Details of the model is provided in Section 2 with an equation for the probabilistic density function (PDF) of system states. In Section 3, we validate our approach with both historical failure data and results from conventional methods; main features of the model, including the temporal evolutions of its full statistical information, are later addressed in Section 4. The final conclusions are given in Section 5.

2. Problem Formulation

2.1. Model Formulation. Widely used in many practical applications, NHPP model assumes the following:

(1) the fault debugging process is modeled as a stochastic process within a continuous state space;

(2) the system is subject to failures at random times caused by the remaining errors in the system;

(3) the mean number of detected faults in a time interval is proportional to the mean number of remaining faults;

(4) when a fault is detected, it may be removed with the probability p;

(5) when a fault is removed, new faults may be generated with a constant probability q.

In order to describe the stochastic and uncertain behavior of the fault debugging process, in a continuous state space, the temporal evolution of $M(t)$ is routinely described as [12]

$$\frac{\mathrm{d}M}{\mathrm{d}t} = r(t)\left(a + qM - pM\right), \tag{1}$$

where $M(t)$ is the expected number of faults detected in the time interval $(0, t]$, a denotes the number of total faults, and $r(t)$ is a nonnegative function representing the fault detection rate per remaining fault at testing time t. We further note that $q(t)M$ is the number of newly introduced bugs, while $p(t)M$ means the number of successful removal; together they represent a (perfect/imperfect) debugging process in which probabilities q and p are assigned [30–32]. Without a loss of generality, we denote $k(t) = p(t) - q(t)$ to facilitate subsequent presentation and $k = 1$ indicates a perfect debugging process.

Since $r(t)$ is subject to a number of random environmental effects [23], we follow the previous studies [24, 25, 27] and represent $r(t)$ as a sum of its mean value $\langle r(t)\rangle$ and a zero-mean (random) fluctuation $r'(t)$:

$$r(t) = \langle r(t)\rangle + r'(t), \quad \left\langle r'(t)\right\rangle = 0. \tag{2}$$

By substituting (2) into (1), we obtain a stochastic differential equation:

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \left[\langle r(t)\rangle + r'(t)\right]\left[a - k(t)M\right]. \tag{3}$$

In contrast to earlier works of [24, 25, 27] which approximate the fluctuations term $r'(t)$ as a white noise, we relax such assumption and denote the noise's two-point covariance more broadly as

$$\left\langle r'(t)\, r'(s)\right\rangle = \sigma_r^2 \rho\left(\frac{t-s}{\tau}\right), \tag{4}$$

where σ_r^2 represents the strength of the noise, ρ denotes its correlation function, t and s indicate two temporal points, and τ is the noise correlation time.

2.2. PDF Method. Our goal is to derive an equation for the probabilistic density function of M, that is, $f_M(m;t)$. To be specific, we adopt the PDF method originally developed in the context of turbulent flow [33] and use the Dirac delta function $\delta(\cdot)$ to define a "raw" PDF of system states M at time t:

$$\Pi(m, M; t) = \delta[M(t) - m], \tag{5}$$

whose ensemble average over random realisations of M yields $f_M(m;t)$

$$\langle \Pi \rangle \equiv \int \delta\left(M' - m\right) f_M\left(M'; t\right) \mathrm{d}M' = f_M(m;t). \tag{6}$$

Following recent study on PDF method [34], we multiply the original equation (3) with $\partial\Pi/\partial m$ and apply the properties of the Dirac delta function $\delta(m - M)g(M) = \delta(m - M)g(m)$. This would yield the stochastic evolution equation for Π in the probability space ($m \in \Omega$):

$$\frac{\partial \Pi}{\partial t} = -\frac{\partial}{\partial m}\left\{\left[\langle r(t)\rangle + r'(t)\right][a - k(t)\, m]\, \Pi\right\}. \tag{7}$$

Now we take the ensemble average of (7) and apply Definition (6); by employing a closure approximation, variously known as the large-eddy diffusivity (LED) closure [35] or the weak approximation [36], an equation of f_M can be obtained

$$\frac{\partial f_M}{\partial t} = -\frac{\partial}{\partial m}\left(\mathscr{V} f_M\right) + \frac{\partial}{\partial m}\left(\mathscr{D}\frac{\partial f_M}{\partial m}\right), \tag{8a}$$

where the eddy diffusivity $\mathscr{D}$ and effective velocity $\mathscr{V}$ are

$$\begin{aligned}\mathscr{D}(m,t) &\approx \int_0^t \int_\Omega \left\langle r'(t)\, r'(s)\right\rangle [a - k(t)\, m]\,[a - k(s)\, y] \\ &\quad \cdot \mathscr{G}_d \mathrm{d}y\, \mathrm{d}s,\end{aligned} \tag{8b}$$

$$\begin{aligned}\mathscr{V}(m,t) &\approx [a - k(t)\, m]\,\langle r(t)\rangle + \int_0^t \int_\Omega \left\langle r'(t)\, r'(s)\right\rangle \\ &\quad \cdot [a - k(t)\, m]\, k(s)\, \mathscr{G}_d \mathrm{d}y\, \mathrm{d}s.\end{aligned} \tag{8c}$$

And Green's function $\mathscr{G}_d(y, s; m, t)$ satisfies the deterministic partial differential equation

$$\begin{aligned}&\frac{\partial \mathscr{G}_d}{\partial s} + [a - k(s)\, y]\,\langle r(s)\rangle \frac{\partial \mathscr{G}_d}{\partial y} \\ &\quad = -\delta\left(y - m\right)\delta(s - t),\end{aligned} \tag{8d}$$

subject to the homogeneous initial (and boundary) conditions corresponding to their (possibly inhomogeneous) counterparts for the raw PDF (7).

3. Model Validation

In this section, we validate our PDE model (8a), (8b), (8c), and (8d) by computing the mean fault detection $\langle M \rangle$ and compare it with historical data and two other methods, namely, the classic deterministic SRGMs (generalised NHPP model) and the popular stochastic SRGMs (white-noise model). As shown in Table 1, four sets of data are used: the first one (DS1) is from WebERP system [37]; the second (DS2) originated from the open source project management software [38]; and the third (DS3) and fourth (DS4) are from the first and fourth product releases of the software products at Tandem Computers [32]. Among those four sets of data, the two UDP/TCP-based systems (DS1 and DS2) are more likely to be affected by environmental factors, whereas the TCP-based software development at Tandem Computers Inc. takes place in a more enclosed/stable environment. UDP is a connectionless-oriented transport protocol and will introduce larger correlation strength for system based on it, while TCP is connection-oriented and will generate smaller correlation strength. We encapsulate such difference in terms of noise variance, for example, larger σ_r^2 for UDP/TCP-based systems (DS1 and DS2). Moreover, DS1, DS2, and DS3 are from the imperfect debugging process, while DS4 is from the perfect debugging process; we demonstrate this diverse with various configurations of the subsequent presentation k. The testing time of DS1, DS2, DS3, and DS4 is 60 (months), 29 (weeks), 20 (weeks), and 19 (weeks), respectively. For detailed information, please refer to the data values tabulated in the Appendix.

It is also noted here that we consider a special case, that is, constant $k(t) = k$, by adopting earlier works [30, 39, 40]. A stationary Gaussian process is prescribed to the fluctuation term, $r' \sim \mathcal{N}(0, \sigma_r^2)$, with an exponential correlation $\rho(t, s; \tau) = \exp(-|t - s|/\tau)$.

3.1. Model Solution for Constant k. For model validation, we adopt the popular generalized Goel-Okumoto model (G-O), as this enables us to derive an analytical solution of (3) using separation of variables. The expression for the mean fault detection rate per remaining fault using the generalised G-O model is given by [41]

$$\langle r(t)\rangle = bt^d, \tag{9}$$

where $b \in [0, 1]$ is a constant and d is the shape factor.

Following the PDF method, we can obtain an analytical solution of f_M:

$$f_M(m;t) = \begin{cases} \dfrac{\exp\left[-\left(\widetilde{m} + \widetilde{t}\right)^2 / 2\sigma_I^2\right]}{\sigma_I (a - km)\sqrt{2\pi}}, & m < a \\ 0, & m \geq a, \end{cases} \tag{10a}$$

Table 1: Data sets for model validation.

Data set	Time unit	Detected failure	Description
DS1 [37, 38]	60 months	146	Web-based integrated accounting ERP system (WebERP) from August 2003 to July 2008 which uses HTTP on top of UDP/TCP
DS2 [37, 38]	49 weeks	94	Open source project management software (OpenProj) from August 2007 to July 2008 used with Remote Display Protocol (RDP) over UDP/TCP
DS3 [32]	20 weeks	100	Release 1 of software testing data collected from Tandem Computers which uses a simple Session Control Protocol (SCP) on top of TCP
DS4 [32]	19 weeks	42	Release 4 of software testing data collected from Tandem Computers applying (SCP) on top of TCP

Table 2: Approximation schemes [51] of PDF for the random integral: $I_{r'} = \int_0^t r'(t')\mathrm{d}t' = \sum_{i=1}^{N} \int_{(i-1)\Delta}^{i\Delta} r'(t')\mathrm{d}t$, where $\Delta = t/N$.

Cases	$t \ll \tau$	$t \gg \tau$	Others
Variance σ_I^2	$\sigma_r^2 t^2$	$2\sigma_r^2 t$	$2N\sigma_r^2 \left[\int_0^{\Delta} \left(\Delta - t'\right) \rho\left(t'\right) \mathrm{d}t' + \sum_{i=2}^{N} \int_0^{\Delta} \int_{(i-1)\Delta}^{i\Delta} \rho\left(t' - t''\right) \mathrm{d}t' \mathrm{d}t'' \right]$
$\frac{\mathrm{d}\sigma_I}{\mathrm{d}t}(t;T)$	σ_r	$\frac{1}{\sqrt{2(T+t)}}$	$\frac{1}{\sigma_I} \left\{ \int_0^{\Delta} \rho\left(t'\right) \mathrm{d}t' + \sum_{i=2}^{N} \left[\int_0^{\Delta} i\rho\left(i\Delta - t''\right) \mathrm{d}t'' - \int_0^{\Delta} (i-1)\,\rho\left[(i-1)\,\Delta - t''\right] \mathrm{d}t'' + \int_{(i-1)\Delta}^{i\Delta} \rho\left(t' - \Delta\right) \mathrm{d}t' \right] \right\}$

where

$$\widetilde{m} = \frac{\ln\left(1 - km/a\right)}{k}, \qquad \widetilde{t} = \frac{bt^{d+1}}{d+1}, \tag{10b}$$

and σ_I^2 is the variance of the random integral $I_{r'} = \int_0^t r'(s)\mathrm{d}s$, whose PDF follows a Gaussian distribution, $f_{I_{r'}} \sim \mathcal{N}(0, \sigma_I^2)$, as outlined in Table 2.

The cumulative density function (CDF) $F_M(m;t)$ can be derived as $F_M(m;t) = \int f_M(m';t)\mathrm{d}m'$:

$$F_M(m;t) = \begin{cases} \dfrac{1}{2}\left[1 - \operatorname{erf}\left(\dfrac{\widetilde{m} + \widetilde{t}}{\sigma_I\sqrt{2}}\right)\right], & m < a \\ 1, & m \geq a, \end{cases} \tag{11}$$

and $\operatorname{erf}(*) \in [-1, 1]$ represents the error function.

The mean fault detection $\langle M \rangle$ is then

$$\langle M \rangle = \frac{a}{k} \int_{-\infty}^{\infty} \left\{1 - \exp\left[-k\left(I_{r'} + \frac{bt^{d+1}}{d+1}\right)\right]\right\} f_{I_{r'}} \mathrm{d}I_{r'}. \tag{12}$$

It is noted here that the fault detection rate model (12) describes an imperfect debugging process for constant k. In many circumstances, time delay, failed removal, and new faults would lead to the problem that $k(t)$ is a time-dependent variable [6, 7, 13, 42], so that we are not able to get an analytical solution $\langle M \rangle$ and can only obtain a numerical solution. In the current study, our focus lies on the uncertainty quantification framework of environmental factors in imperfect debugging process; without a loss of generality, subsequent approach could also be extended to the imperfect debugging process with time-dependent $k(t)$.

3.2. Performance Criteria. To evaluate the performance of our PDE model, we employ four comparison criteria, for example, the predicted relative error (PRE), mean square error (MSE), Coefficient of Determination (R^2), and root mean square prediction error (RMSPE).

3.2.1. Predicted Relative Error (PRE). The relative difference between observed and estimated number of faults at time t, known as predictive validity [7, 43], is as follows:

$$\mathrm{PRE}(t) = \frac{M(t) - N(t)}{N(t)}, \tag{13}$$

where N is the fault number from observed data at time t. Hence for a better fit to the data, we seek $\mathrm{PRE}(t) \to 0$ [44].

3.2.2. Mean Square Error (MSE). Consider

$$\mathrm{MSE}(t) = \frac{\sum_{i=1}^{K} \left[M\left(t_i\right) - N\left(t_i\right)\right]^2}{N_o}, \tag{14}$$

where $M(t_i)$ and $N(t_i)$ denote the estimated and observed number of faults detected at time t_i, $(i = 1, 2, \ldots, K)$, respectively, K is the total number of observation data points. A lower value of $\mathrm{MSE}(t)$ represents better model prediction [41, 45].

3.2.3. Coefficient of Determination (R^2). We define this coefficient as the ratio of the sum of squares resulting from the trend model to that from constant model:

$$R^2(t) = 1 - \frac{\sum_{i=1}^{K} \left[M\left(t_i\right) - N\left(t_i\right)\right]^2}{\sum_{i=1}^{K} \left[N\left(t_i\right) - \overline{N}\right]^2}, \tag{15}$$

where $\overline{N}$ is the average of detected faults, $\overline{N} = \sum_{i=1}^{K} N(t_i)/K$. $R^2 \in [0, 1]$ measures the percentage of the total variation about the mean accounted for the fitted curve. The lager R^2 is, the better the model explains the variation in the data.

3.2.4. Root Mean Square Prediction Error (RMSPE). The average of prediction errors (PEs) is known as Bias [46]. The standard deviation of predicted relative variation (PRV) is known as predicted relative variation [46]. The root mean square prediction error (RMSPE) is a measure of the closeness with which the model predicts the observation and is a measure of closeness with which a model predicts the observation [46]:

$$\text{RMSPE}(t) = \sqrt{\text{Bias}^2 + \text{PRV}^2}. \tag{16}$$

Given

$$\text{PE}(t_i) = M(t_i) - N(t_i), \tag{17a}$$

$$\text{Bias} = \sum_{i=1}^{K} \left[\frac{\text{PE}(t_i)}{K} \right], \tag{17b}$$

$$\text{PRV}(t) = \sqrt{\frac{\sum_{i=1}^{K} (\text{PE}_i - \text{Bias})^2}{K-1}}. \tag{17c}$$

The lower the values of Bias, PRV, and RMSPE, the better the goodness of fit.

3.3. Model Parameter Estimation. There are six input parameters for our PDE method: the total number of initial faults a, the fault detection rate b, the shape factor d, the subsequent presentation k, the noise strength σ_r^2, and correlation time τ. Given a data set, we used SPSS to estimate the parameters a, b, and d, and the methods for estimating three parameters include method of moments, least square method, maximum likelihood, and Bayesian methods [47–49]. For the subsequent presentation k, we use the empirical value of [30, 39, 40]. About the noise parameters, we adopt earlier works on white noise [24] for the strength of our correlated noise σ_r^2, which yields the optimal solution as discussed in the sensitivity analysis Section 3.5. Regarding the noise correlation time τ, we take the fact that a software failure is typically repaired within days [50] and hence set $\tau = 0.1$ (month) or (week) for later analysis. A stationary Gaussian process is prescribed to the fluctuation term, $r' \sim \mathcal{N}(0, \sigma_r^2)$, with an exponential correlation $\rho(t, s; \tau) = \exp(-|t - s|/\tau)$.

3.4. Validation Results. In this section, we validate our PDE model (12) by computing the mean fault detection $\langle M \rangle$ and compare it with historical data and two other methods, namely, the classic deterministic SRGMs (generalised NHPP model) and the popular stochastic SRGMs (white-noise model). Figure 1 shows expected number of faults $\langle M \rangle$ from our PDE model, the G-O model, and the white-noise model against four data sets over the testing time, respectively. Parameters of all three models are adjusted for their best fit. As demonstrated in Figure 1, though our PDE model overestimates the number of faults for DS1 and leads to underestimation for DS2, DS3, and DS4, overall it provides the overall best match to the four data sets, compared with the G-O model and the white-noise model.

Figure 2 shows the PRE results for our PDE model, G-O model, and white-noise model over the testing time. Similar trends are exhibited for all three models: large estimation error (deviation from zero) at the beginning but gradual convergence to observation at later time. This is expected, since few data can be utilised at earlier time to gauge the models' parameters; as time elapses and more data become available, the models' accuracy improve and hence their PRE approaches zero. Overall, the PDE model yields the smallest PRE.

We tabulated the aforementioned evaluation results of our PDE model, G-O model, and white-noise model for DS1, DS2, DS3, and DS4 in Table 3. Despite a slightly higher value of PRV for DS1, the PDE model presents best goodness-to-fit with all four data sets in all performance indicators.

3.5. Sensitivity Analysis. Having validated our PDE model with four observation real data sets, we now conduct the sensitivity analysis to study the impact of correlation time τ, strength σ_r^2, and function $\rho(t, s; \tau)$ on our fault prediction.

We first investigate the impact of the correlation time τ. Figure 3 and Table 4 present the PRE, MSE, R^2, Bias, PRV, and RMSPE results on DS2 for our PDE model with parameters $a = 156$, $b = 0.5446$, $d = 0.026$, $\sigma_r^2 = 0.0002228$, and $k = 0.5092$, G-O model with parameters $a = 94$ and $b = 0.146$, and white-noise model with parameters $a = 97$, $b = 0.16246$, $d = 0.026$, $\sigma_r^2 = 0.0002228$, and $k = 0.5092$. It demonstrates that the PDE model fits best the real data and the repair time is $\tau = 0.1$ week. Hence a correlation time $\tau = 0.1$ (month or week) is a good approximation of fault repair time in reality.

Next we study the impact of correlation strength. Figure 4 and Table 5 show the results with DS2 for our PDE model with parameters $a = 156$, $b = 0.5446$, $d = 0.026$, and $\tau = 0.1$, G-O model with parameters $a = 94$ and $b = 0.146$, and white-noise model with parameters $a = 97$, $b = 0.16246$, $d = 0.026$, and $k = 0.5092$. It demonstrates that σ_r^2 has a large effect on the white-noise model, while its influence on the PDE model is very limited.

Lastly we conduct the sensitivity analysis of the correlation function. Five types of correlation function, exponential, besselj, normal, cosine, and cubic, are studied. Figure 5 and Table 6 show the results on DS2 for our PDE model with parameters $a = 156$, $b = 0.5446$, $d = 0.026$, $\sigma_r^2 = 0.0002228$, $k = 0.5092$, and $\tau = 0.1$, G-O model with parameters $a = 94$ and $b = 0.146$, and white-noise model with parameters $a = 97$, $b = 0.16246$, $d = 0.026$, $k = 0.5092$, and $\sigma_r^2 = 0.0002228$. It is clear that the correlation type significantly affect our model estimation. It also demonstrates that the PDE model with exponential correlation function fits best.

4. Reliability Assessment

Having validated our model with historical data, we now apply our framework first to study the detection process behaviour through its full statistical information in Section 4.1 and later to assess software reliability in Section 4.2.

4.1. Temporal Evolution of PDF. In addition to conventional focus on the mean value of fault detection, $\langle M \rangle$, we now

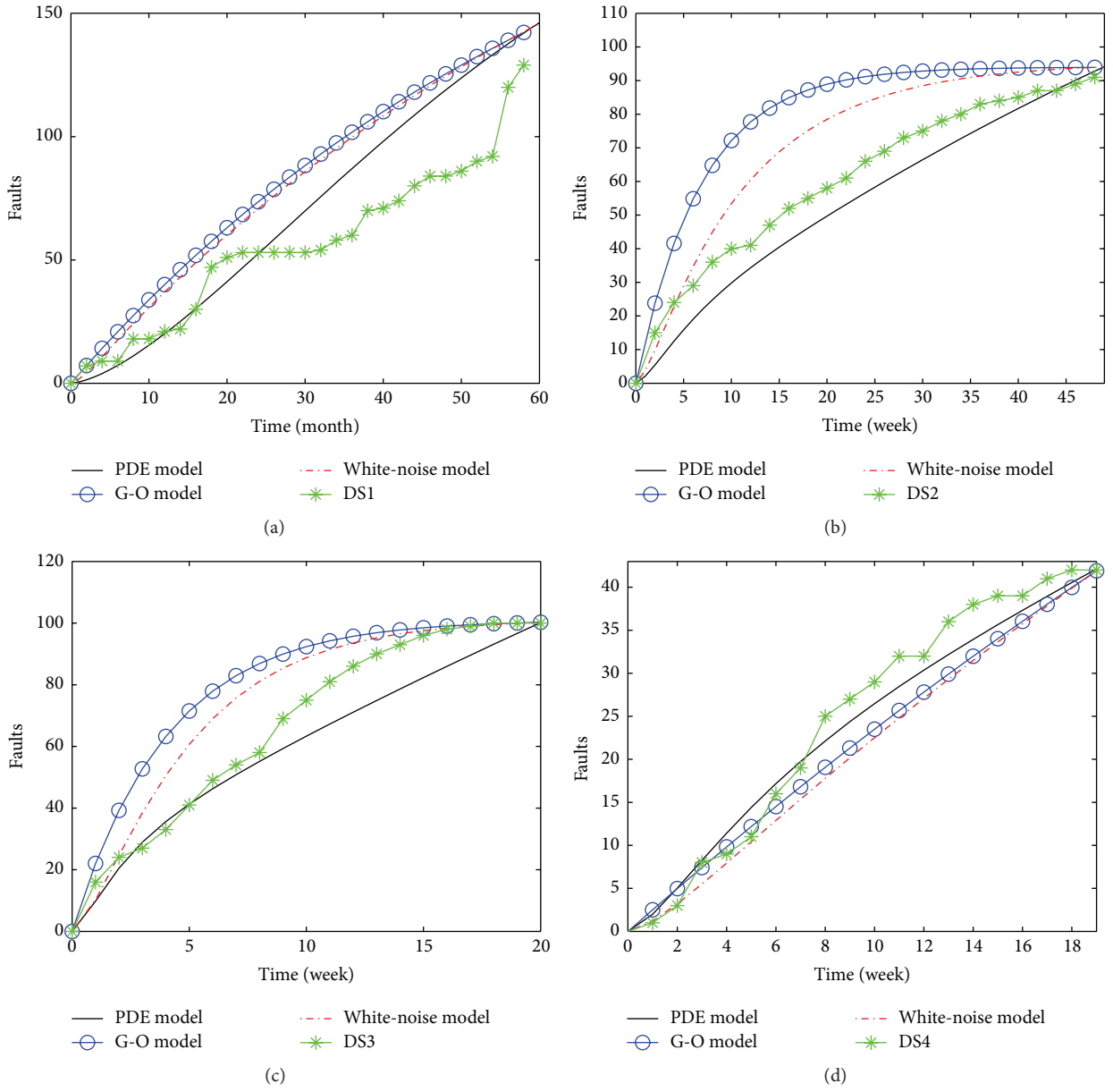

FIGURE 1: Expected number of faults, $\langle M \rangle$, computed from the PDE model, G-O model, and white-noise model, are compared with four data sets: (a) DS1; (b) DS2; (c) DS3; and (d) DS4.

include full statistical information, that is, the probabilistic density function (PDF) and the cumulative density function (CDF), to study system behaviour.

Three snapshots of the PDF $f_M(m;t)$ and CDF $F_M(m;t)$ at time $t = 2$, 10 and 50 weeks are plotted in Figure 6, respectively, using $a = 156$, $b = 0.005446$, $d = 0.026$, $\sigma_r^2 = 0.0002228$, $k = 0.5092$, and $\tau = 0.1$. The uncertainty of our prediction, encapsulated in the width of the PDF and the slope of the CDF, initially rises from 1 to 10 weeks; it later falls at 50 weeks as most of the faults are detected. Here we propose employing the CDF as a criteria to ensure certain levels of reliability. For example, in Figure 6, for a 90% software reliability in 2, 10, and 50 weeks, the minimum number of faults detected must be 3, 10, and 44, respectively.

4.2. Software Reliability and Reliability Growth Rate. Software reliability $R(t;T)$ is broadly defined as the probability that no software fault is detected in the time interval $[T, T+t]$, given that the last fault occurred at time T. The software reliability can be expressed as

$$R(t;T) = e^{-[\langle M(T+t)\rangle - \langle M(T)\rangle]}, \tag{18}$$

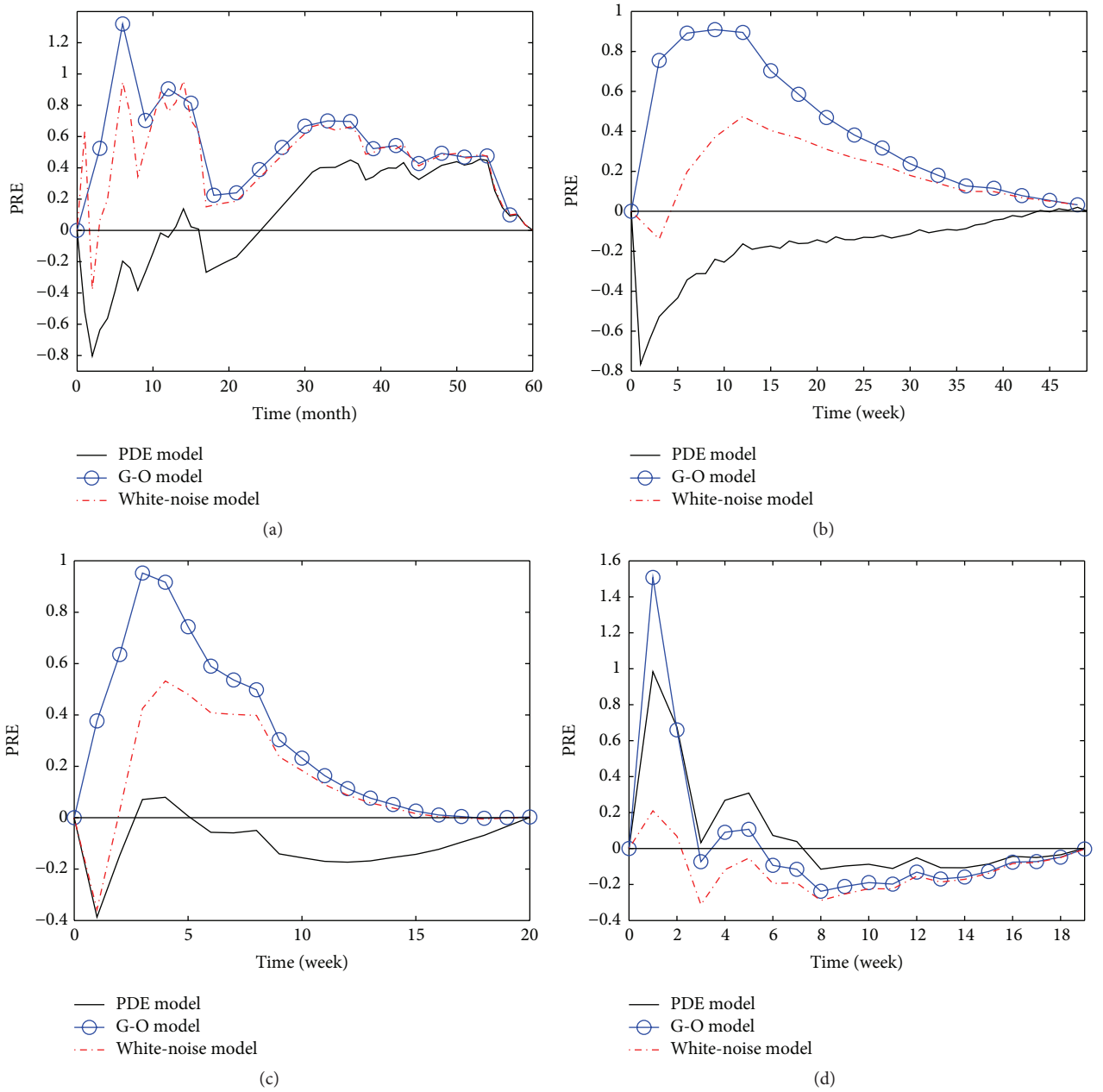

FIGURE 2: Predicted relative error (PRE) for the PDE model, G-O model, and white-noise model are compared with four data sets in (a) DS1, (b) DS2, (c) DS3, and (d) DS4, respectively.

Thus, the software reliability growth rate, $\mathrm{RGR}(t;T)$, a measure of reliability change rate, can be computed as

$$\begin{aligned}\mathrm{RGR}(t;T) &= \frac{d}{dt}R(t;T)\\ &= aR\int_{-\infty}^{\infty}\left[\frac{d\sigma_I}{dt}\frac{\sigma_I^2 - I_{r'}^2}{\sigma_I^3 k}\left(1-e^{-\beta}\right)\right.\\ &\quad \left. - b(T+t)^d e^{-\beta}\right] f_{I_{r'}}\,dI_{r'},\end{aligned} \tag{19a}$$

where

$$\beta = k\left(I_{r'} + \frac{bt^{d+1}}{d+1}\right). \tag{19b}$$

The temporal derivative of the variance $d\sigma_I/dt$ is listed in Table 2.

Three snapshots of software reliability, $R(t;T)$, at time T = 2nd, 10th, 50th weeks, are presented in Figure 7, using a = 156, b = 0.005446, d = 0.026, σ_r^2 = 0.0002228, k = 0.5092, and τ = 0.1. Reliability at all three timeframes exhibits initial drop but such deterioration gradually slows down and $R(t;T)$ eventually approaches zero as $t \to \infty$. This is expected,

Table 3: The MSE, R^2, Bias PRV, and RMSPE results of PDE model, G-O model, and white-noise model for DS1, DS2, DS3, and DS4.

Models	Data set	Setting	MSE	R-square	Bias	PRV	RMSPE
PDE model	DS1	$a = 432$, $b = 0.04047$, $d = 0.028$, $\sigma_r^2 = 0.0002236$, $k = 0.9536$, and $\tau = 0.1$	419.6179	0.6466	12.5213	16.3490	13.1506
	DS2	$a = 156$, $b = 0.5446$, $d = 0.026$, $\sigma_r^2 = 0.0002228$, $k = 0.5092$, and $\tau = 0.1$	61.1382	0.8853	−6.9274	3.6637	6.8133
	DS3	$a = 605$, $b = 0.846$, $d = 0.025$, $\sigma_r^2 = 0.0002219$, $k = 0.9824$, and $\tau = 0.1$	82.1784	0.8989	−6.9221	6.0056	9.1642
	DS4	$a = 269$, $b = 0.3046$, $d = 0.024$, $\sigma_r^2 = 0.0002207$, $k = 1$, and $\tau = 0.1$	5.9284	0.9682	−0.9743	2.2926	2.4910
G-O model	DS1	$a = 249$ and $b = 0.0146$	847.9599	0.2858	25.6179	13.9619	29.1755
	DS2	$a = 94$ and $b = 0.146$	495.6554	0.0701	19.3977	11.0395	22.3191
	DS3	$a = 101$ and $b = 0.246$	318.5772	0.6081	13.5753	11.8893	12.2783
	DS4	$a = 173$ and $b = 0.0146$	15.3147	0.9178	−2.7146	2.8961	3.9694
white-noise model	DS1	$a = 510$, $b = 0.0126$, $d = 0.028$, $k = 0.9536$, and $\sigma_r^2 = 0.0002236$	760.9538	0.3591	23.6059	14.3934	27.6479
	DS2	$a = 97$, $b = 0.16246$, $d = 0.026$, $k = 0.5092$, and $\sigma_r^2 = 0.0002228$	171.8446	0.6776	10.9225	7.3238	13.1506
	DS3	$a = 199$, $b = 0.226$, $d = 0.025$, $k = 0.9824$, and $\sigma_r^2 = 0.0002219$	146.6972	0.8195	8.8329	9.1706	12.7326
	DS4	$a = 377$, $b = 0.0126$, $d = 0.024$, $k = 1.0000$, and $\sigma_r^2 = 0.0002207$	20.4094	0.8905	−3.6981	2.6661	4.5589

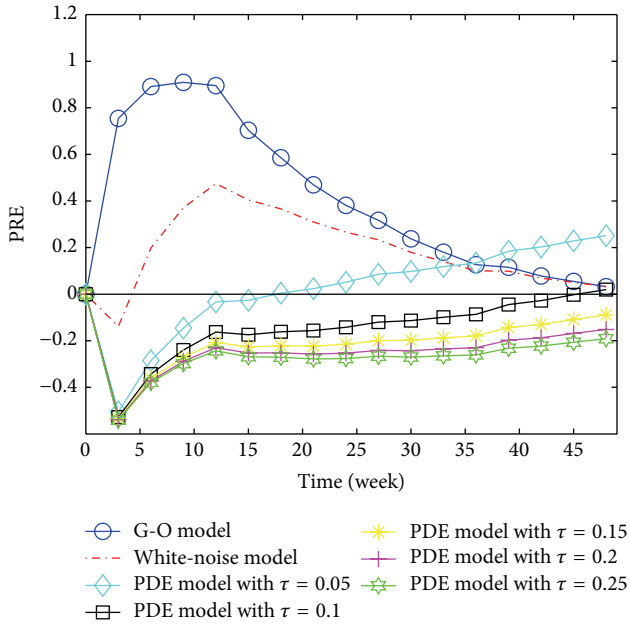

Figure 3: Sensitivity analysis of PRE for PDE model, G-O model, and white-noise model with various configurations of τ.

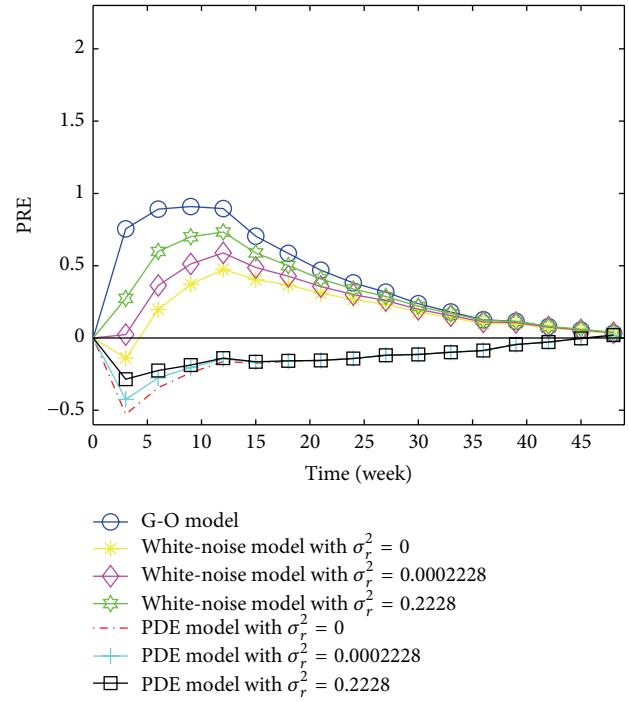

Figure 4: Sensitivity analysis of PRE for PDE model, G-O model, and white-noise model with various configurations of σ_r^2.

since, at earlier time, most of the faults are yet removed; with elapsing time, more are debugged and this leads to a fall in software reliability; when the system is cleared of all faults, $R(t;T)$ reaches a stationary state. Therefore, later timeframe (T = 50th week) also presents a sharper drop to zero.

For a closer examination, we now look at the software reliability growth rate, RGR$(t;T)$, at T = 2nd, 10th, 50th weeks. As shown in Figure 7, RGR in all three timeframes demonstrates an initial rise to maximum and late fall to stationarity of zero, albeit earlier timeframe (T = 2nd week) exhibits higher zenith (RGR = 600), since fewer faults are removed at earlier stage.

Table 4: Various configurations of τ in obtaining MSE, R^2, Bias, PRV, and RMSPE values of PDE model, G-O model, and white-noise model.

Model	MSE	R^2	Bias	PRV	RMSPE
G-O model	495.6554	0.0701	19.3977	11.0395	22.3191
White-noise model	171.8446	0.6776	10.9225	7.3238	13.1506
PDE model with $\tau = 0.05$	126.5667	0.7625	5.1685	10.0962	11.3423
PDE model with $\tau = 0.10$	61.1382	0.8853	−6.9274	3.6637	7.8366
PDE model with $\tau = 0.15$	146.7881	0.7244	−11.9221	2.1791	12.1196
PDE model with $\tau = 0.20$	222.5672	0.5824	−14.6230	2.9862	14.9248
PDE model with $\tau = 0.25$	280.6167	0.4735	−16.3072	3.8729	16.7608

Table 5: Various configurations of σ_r^2 in obtaining MSE, R^2, Bias, PRV, and RMSPE values of PDE model, G-O model, and white-noise model.

Model	MSE	R^2	Bias	PRV	RMSPE
G-O model	495.6554	0.0701	19.3977	11.0395	22.3191
White noise model with $\sigma_r^2 = 0$	171.6587	0.6779	10.9154	7.3217	13.1435
White noise model with $\sigma_r^2 = 0.0002228$	171.8446	0.6776	10.9225	7.3238	13.1506
White noise model with $\sigma_r^2 = 0.2228$	343.6834	0.3552	16.1015	9.2835	18.5861
PDE model with $\sigma_r^2 = 0$	61.1445	0.8853	−6.9277	3.6640	7.8370
PDE model with $\sigma_r^2 = 0.0002228$	61.1382	0.8853	−6.9274	3.6637	7.8366
PDE model with $\sigma_r^2 = 0.2228$	56.5464	0.8939	−6.6843	3.4804	7.5362

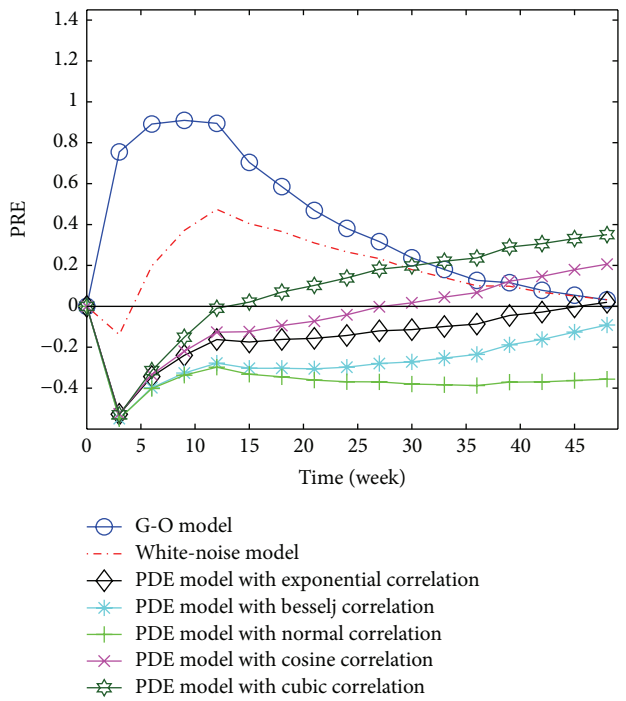

Figure 5: Sensitivity analysis of PRE for PDE model, G-O model, and white-noise model with various correlation functions.

5. Conclusion

We presented a novel model to quantify uncertainties associated with the perfect or imperfect debugging process. Treating the random environmental impacts collectively as a noise with arbitrary correlation, an equation of the distribution of fault detection, that is, its probabilistic density function (PDF), is derived. Our analysis leads to the following major conclusions.

(i) Under the conventional evaluation criteria on mean value of detected fault in the perfect or imperfect debugging process, our model provides best fit to observation data comparing to the classic G-O model and white-noise model.

(ii) The proposed approach also goes beyond uncertainty quantification based on mean and variance of system states by computing its PDF and cumulative density function (CDF). This enables one to evaluate probabilities of rare events, which are necessary for probabilistic risk assessment.

(iii) Predictive uncertainty of fault detection initially increases but gradually diminishes as time elapses; hence more maintenance is needed at the initial stage.

(iv) Software developers could also use CDF as a new criteria to assess system reliability, for example, by computing the minimum number of faults detected for an prescribed confidence level.

(v) The proposed correlated model describes an imperfect debugging process for constant k, which enables us to obtain an analytical solution $\langle M \rangle$. The approach could also be extended to the imperfect debugging process with time-dependent $k(t)$, to help us drive a numerical solution by computing its PDF. This would be the focus of our future work.

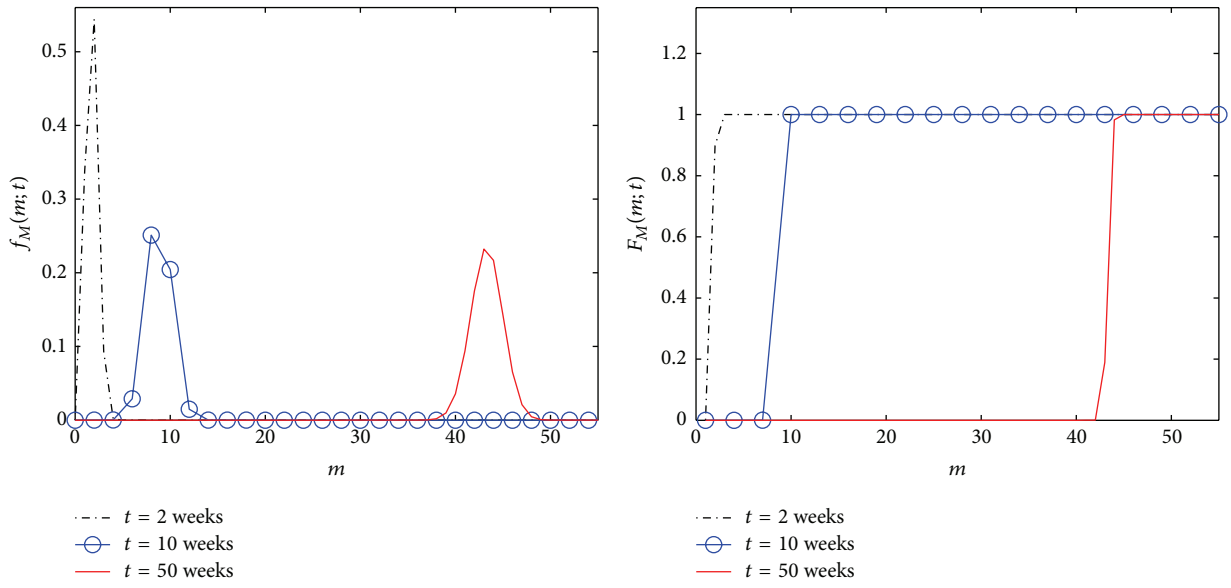

FIGURE 6: Snapshots of the probabilistic density function (PDF), $f_M(m;t)$, and the cumulative density function (CDF), $F_M(m;t)$, at time t = 2, 10 and 50 weeks, using a = 156, b = 0.005446, d = 0.026, σ_r^2 = 0.0002228, k = 0.5092, and τ = 0.1.

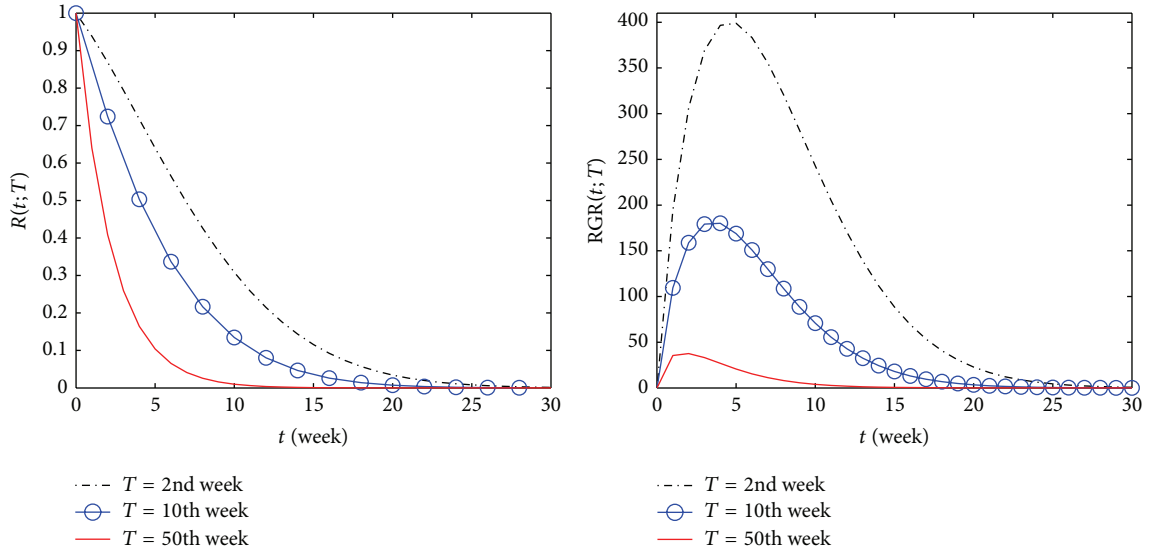

FIGURE 7: Temporal Evolutions of software reliability, $R(t;T)$, and software reliability growth rate, RGR$(t;T)$, at time T = 2nd, 10th, 50th week, using a = 156, b = 0.005446, d = 0.026, σ_r^2 = 0.0002228, k = 0.5092, and τ = 0.1.

TABLE 6: Various correlation functions in obtaining MSE, R^2, Bias, PRV, and RMSPE values of PDE model, G-O model, and white-noise model.

Model	MSE	R^2	Bias	PRV	RMSPE
G-O model	495.6554	0.0701	19.3977	11.0395	22.3191
White-noise model	171.8446	0.6776	10.9225	7.3238	13.1506
PDE model with exponential correlation	61.1382	0.8853	−6.9274	3.6637	7.8366
PDE model with besselj correlation	246.4608	0.5374	−15.2180	3.8966	15.7089
PDE model with normal correlation	602.0697	0.1296	−23.1279	8.2806	24.5656
PDE model with cosine correlation	84.3103	0.8418	0.7364	9.2473	9.2766
PDE model with cubic correlation	287.8605	0.4599	10.2405	13.6676	17.0784

Table 7: The failure data of web-based and integrated accounting ERP system (DS1) from August 2003 to July 2008 [37, 38].

Time unit (month)	Detected faults	Corrected faults	Time unit (month)	Detected faults	Corrected faults	Time unit (month)	Detected faults	Corrected faults
1	1	1	21	53	50	41	72	65
2	7	7	22	53	50	42	74	67
3	7	7	23	53	50	43	74	67
4	9	9	24	53	50	44	80	67
5	9	9	25	53	53	45	84	69
6	9	9	26	53	53	46	84	76
7	12	11	27	53	53	47	84	79
8	18	15	28	53	53	48	84	82
9	18	15	29	53	53	49	85	83
10	18	18	30	53	53	50	86	84
11	18	18	31	53	53	51	89	87
12	21	21	32	54	54	52	90	90
13	22	22	33	56	54	53	90	90
14	22	22	34	58	56	54	92	91
15	27	26	35	59	56	55	108	106
16	30	28	36	60	56	56	120	119
17	45	39	37	63	56	57	128	126
18	47	44	38	70	60	58	129	128
19	49	47	39	71	61	59	139	136
20	51	47	40	71	64	60	146	136

Table 8: The failure data of open source project management software (DS2) from August 2007 to July 2008 [37, 38].

Time unit (week)	Detected faults	Corrected faults	Time unit (week)	Detected faults	Corrected faults	Time unit (week)	Detected faults	Corrected faults
1	9	0	18	55	31	35	82	41
2	15	2	19	57	31	36	83	41
3	19	11	20	58	31	37	83	44
4	24	12	21	61	31	38	84	44
5	28	12	22	61	31	39	84	44
6	29	13	23	64	33	40	85	44
7	32	14	24	66	33	41	85	45
8	36	15	25	67	33	42	87	45
9	36	20	26	69	33	43	87	46
10	40	21	27	70	33	44	87	46
11	41	22	28	73	33	45	89	47
12	41	22	29	74	33	46	89	62
13	45	24	30	75	36	47	91	62
14	47	25	31	75	38	48	91	65
15	49	30	32	78	38	49	94	65
16	52	30	33	79	40	—	—	—
17	52	31	34	80	41	—	—	—

Appendix

Data Sets

In the Appendix, four data sets used are presented. The detailed information about two data sets DS1 and DS2 can be referenced for further study at the SourceForge website. Originally, these two data sets had belonged to time-between-failures data, as tabulated in Tables 7 and 8. The other software testing data were collected from two major releases of the software products at Tandem Computers [4], as summarized in Table 9.

Table 9: The failure data of tandem software (DS3 and DS4) from the first and fourth release [32].

Time unit (week)	Detected faults	Time unit (week)	Detected faults	Time unit (week)	Detected faults	Time unit (week)	Detected faults
Release 1							
1	16	11	81	6	49	16	98
2	24	12	86	7	54	17	99
3	27	13	90	8	58	18	100
4	33	14	93	9	69	19	100
5	41	15	96	10	75	20	100
Release 4							
1	1	11	32	6	16	16	39
2	3	12	32	7	19	17	41
3	8	13	36	8	25	18	42
4	9	14	38	9	27	19	42
5	11	15	39	10	29	—	—

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] IEEE, "Standard glossary of software engineering terminology," IEEE Std. STD-729-1991, 1991.

[2] V. Almering, M. van Genuchten, G. Cloudt, and P. J. M. Sonnemans, "Using software reliability growth models in practice," *IEEE Software*, vol. 24, no. 6, pp. 82–88, 2007.

[3] M. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.

[4] H. Pham, *System Software Reliability*, Springer, London, UK, 2006.

[5] N. Schneidewind and T. Keller, "Applying reliability models to the space shuttle," *IEEE Software*, vol. 9, no. 4, pp. 28–33, 1992.

[6] C.-Y. Huang and M. R. Lyu, "Estimation and analysis of some generalized multiple change-point software reliability models," *IEEE Transactions on Reliability*, vol. 60, no. 2, pp. 498–514, 2011.

[7] J. Zhao, H.-W. Liu, G. Cui, and X.-Z. Yang, "Software reliability growth model with change-point and environmental function," *Journal of Systems and Software*, vol. 79, no. 11, pp. 1578–1587, 2006.

[8] Z. Jelinski and P. Moranda, "Software reliability research," in *Proceedings of the Statistical Methods for the Evaluation of Computer System Performance*, pp. 465–484, Academic Press, New York, NY, USA, 1972.

[9] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206–211, 1979.

[10] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on Reliability*, vol. 32, no. 5, pp. 475–484, 1983.

[11] M. Ohba, *Inflection S-Shaped Software Reliability Growth Models*, Springer, Berlin, Germany, 1984.

[12] S. Yamada, J. Hishitani, and S. Osaki, "Software-reliability growth with a Weibull test-effort: a model and application," *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 100–106, 1993.

[13] C.-Y. Huang, M. R. Lyu, and S. Osaki, "Estimation and analysis of some generalized multiple change-point software reliability models," *IEEE Transactions on Reliability*, vol. 60, no. 2, pp. 498–514, 2011.

[14] P. K. Kapur and R. B. Garg, "Software reliability growth model for an error removal phenomenon," *Software Engineering Journal*, vol. 7, no. 4, pp. 291–294, 1992.

[15] A. L. Goel, "Software reliability models: assumptions, limitations, and applicability," *IEEE Transactions on Software Engineering*, vol. 11, no. 12, pp. 1411–1423, 1985.

[16] M. Ohba and X.-M. Chou, "Does imperfect debugging affect software reliability growth?" in *Proceedings of the 11th International Conference on Software Engineering (ICSE '89)*, pp. 237–244, ACM, Pittsburgh, Pa, USA, May 1989.

[17] H. Pham, L. Nordmann, and X. Zhang, "A general imperfect software debugging model with s-shaped fault detection rate," *IEEE Transactions on Reliability*, vol. 48, no. 2, pp. 169–175, 1999.

[18] H. Pham, "A software cost model with imperfect debugging, random life cycle and penalty cost," *International Journal of Systems Science*, vol. 27, no. 5, pp. 455–463, 1996.

[19] X. Zhang, X. Teng, and H. Pham, "Considering fault removal efficiency in software reliability assessment," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 1, pp. 114–120, 2003.

[20] P. K. Kapur, D. Kumar, A. Gupta, and P. C. Jha, "On how to model software reliability growth in the presence of imperfect debugging and error generation," in *Proceedings of the 2nd International Conference on Reliability and Safety Engineering*, pp. 515–523, Chennai, India, December 2006.

[21] S. Yamada, K. Okumoto, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment," *International Journal of Systems Science*, vol. 23, no. 12, pp. 2253–2264, 1992.

[22] J. Wang, Z. Wu, Y. Shu, and Z. Zhang, "An imperfect software debugging model considering log-logistic distribution fault content function," *Journal of Systems and Software*, vol. 100, pp. 167–181, 2015.

[23] X. Zhang and H. Pham, "Analysis of factors affecting software reliability," *Journal of Systems and Software*, vol. 50, no. 1, pp. 43–56, 2000.

[24] Y. Tamura and S. Yamada, "A flexible stochastic differential equation model in distributed development environment," *European Journal of Operational Research*, vol. 168, no. 1, pp. 143–152, 2006.

[25] P. K. Kapur, S. Anand, S. Yamada, and V. S. S. Yadavalli, "Stochastic differential equation-based flexible software reliability growth model," *Mathematical Problems in Engineering*, vol. 2009, Article ID 581383, 15 pages, 2009.

[26] P. K. Kapur, H. Pham, A. Gupta, and P. C. Jha, *SRGM Using SDE*, Springer, London, UK, 2011.

[27] C. Fang and C. Yeh, "Effective confidence interval estimation of fault-detection process of software reliability growth models," *International Journal of Systems Science*, 2015.

[28] N. Kumar and S. Banerjee, *Measuring Software Reliability: A Trend Using Machine Learning Techniques*, Springer, New York, NY, USA, 2015.

[29] M. D. Pacer and T. L. Griffiths, "Upsetting the contingency table: causal induction over sequences of point events," in *Proceedings of the 37th Annual Conference of the Cognitive Science Society (CogSci '15)*, Pasadena, Calif, USA, July 2015.

[30] M. Ohba, "Software reliability analysis models," *IBM Journal Research & Development*, vol. 4, no. 28, pp. 328–443, 1984.

[31] H. Pham and X. Zhang, "NHPP software reliability and cost models with testing coverage," *European Journal of Operational Research*, vol. 145, no. 2, pp. 443–454, 2003.

[32] A. Wood, "Predicting software reliability," *Computer*, vol. 29, no. 11, pp. 69–77, 1996.

[33] T. S. Lundgren, "Distribution functions in the statistical theory of turbulence," *Physics of Fluids*, vol. 10, no. 5, pp. 969–975, 1967.

[34] P. Wang, A. M. Tartakovsky, and D. M. Tartakovsky, "Probability density function method for langevin equations with colored noise," *Physical Review Letters*, vol. 110, no. 14, Article ID 140602, 2013.

[35] R. H. Kraichnan, "Eddy viscosity and diffusivity: exact formulas and approximations," *Complex Systems*, vol. 1, pp. 805–820, 1987.

[36] S. P. Neuman, "Eulerian-lagrangian theory of transport in space-time nonstationary velocity fields: exact nonlocal formalism by conditional moments and weak approximation," *Water Resources Research*, vol. 29, no. 3, pp. 633–645, 1993.

[37] SourceForge.net, "An open source software website," 2008, http://sourceforge.net.

[38] C.-J. Hsu, C.-Y. Huang, and J.-R. Chang, "Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor," *Applied Mathematical Modelling*, vol. 35, no. 1, pp. 506–521, 2011.

[39] W.-C. Huang and C.-Y. Huang, "Software reliability analysis and measurement using finite and infinite server queueing models," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 192–203, 2008.

[40] W.-C. Huang, C.-Y. Huang, and C.-C. Sue, "Software reliability prediction and assessment using both finite and infinite server queueing approaches," in *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC '06)*, pp. 194–201, IEEE, Riverside, Calif, USA, December 2006.

[41] M. Xie, *Software Reliability Modeling*, World Scientific Publishing, Singapore, 1991.

[42] C.-Y. Huang and W.-C. Huang, "Software reliability analysis and measurement using finite and infinite server queueing models," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 192–203, 2008.

[43] J. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York, NY, USA, 1989.

[44] K. Pillai and V. S. S. Nair, "A model for software development effort and cost estimation," *IEEE Transactions on Software Engineering*, vol. 23, no. 8, pp. 485–497, 1997.

[45] C.-Y. Huang and C.-T. Lin, "Software reliability analysis by considering fault dependency and debugging time lag," *IEEE Transactions on Reliability*, vol. 55, no. 3, pp. 436–450, 2006.

[46] K. Pillai and V. S. Nair, "A model for software development effort and cost estimation," *IEEE Transactions on Software Engineering*, vol. 23, no. 8, pp. 485–497, 1997.

[47] W. A. Massey, G. A. Parker, and W. Whitt, "Estimating the parameters of a nonhomogeneous Poisson process with linear rate," *Telecommunication Systems*, vol. 5, no. 4, pp. 361–388, 1996.

[48] J. G. Leite, J. Rodrigues, and L. A. Milan, "A bayesian analysis for estimating the number of species in a population using nonhomogeneous poisson process," *Statistics & Probability Letters*, vol. 48, no. 2, pp. 153–161, 2000.

[49] Z. Yang and C. Liu, "Estimating parameters of Ohba three-parameters NHPP Models," in *Proceedings of the International Conference on Computer Science and Service System (CSSS '11)*, pp. 1259–1261, IEEE, Nanjing , China, June 2011.

[50] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug," in *Proceedings of the 4th International Workshop on Mining Software Repositories (ICSE Workshops MSR '07)*, p. 1, IEEE, Minneapolis, Minn, USA, May 2007.

[51] P. Wang, D. M. Tartakovsky, K. D. Jarman, and A. M. Tartakovsky, "Cdf solutions of buckley-leverett equation with uncertain parameters," *Multiscale Modeling and Simulation*, vol. 11, no. 1, pp. 118–133, 2013.