Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2009, Article ID 581383, 15 pages
doi:10.1155/2009/581383

Research Article

Stochastic Differential Equation-Based Flexible Software Reliability Growth Model

P. K. Kapur,[1] Sameer Anand,[2] Shigeru Yamada,[3] and Venkata S. S. Yadavalli[4]

[1] *Department of Operational Research, University of Delhi, Delhi 110007, India*
[2] *S.S. College of Business Studies, University of Delhi, Delhi 110095, India*
[3] *Department of Social Management Engineering, Graduate School of Engineering, Tottori University, 4-101, Minnami, Koyama, Tottori 680-8552, Japan*
[4] *Department of Industrial and System Engineering, University of Pretoria, Pretoria 0002, South Africa*

Correspondence should be addressed to P. K. Kapur, pkkapur1@gmail.com

Received 28 November 2008; Accepted 14 April 2009

Recommended by Sergio Preidikman

Several software reliability growth models (SRGMs) have been developed by software developers in tracking and measuring the growth of reliability. As the size of software system is large and the number of faults detected during the testing phase becomes large, so the change of the number of faults that are detected and removed through each debugging becomes sufficiently small compared with the initial fault content at the beginning of the testing phase. In such a situation, we can model the software fault detection process as a stochastic process with continuous state space. In this paper, we propose a new software reliability growth model based on Itô type of stochastic differential equation. We consider an SDE-based generalized Erlang model with logistic error detection function. The model is estimated and validated on real-life data sets cited in literature to show its flexibility. The proposed model integrated with the concept of stochastic differential equation performs comparatively better than the existing NHPP-based models.

1. Introduction

Software reliability engineering is a fast growing field. More than 60% of critical applications are dependent on software. The complexity of business software application is also increasing.

Customers need products with high performance that can be sustained over time. Due to high cost of fixing failures, safety concerns, and legal liabilities organizations need to produce software that is reliable. There are several methodologies to develop software but questions that need to be addressed are how many times will the software fail and when, how to estimate testing effort, when to stop testing, and when to release the software. Also, for a software product we need to predict/estimate the maintenance effort; for example, how long must the warranty period must be, once the software is released, how many defects can be expected at what severity levels, how many engineers are required to support the product, for how long, and so forth. Software reliability engineering (SRE) addresses all these issues, from design to testing to maintenance phases.

The Software Reliability Growth Model (SRGM) is a tool of SRE that can be used to evaluate the software quantitatively, develop test status, schedule status, and monitor the changes in reliability performance [1]. In the last two decades several Software Reliability models have been developed in the literature showing that the relationship between the testing time and the corresponding number of faults removed is either Exponential or S-shaped or a mix of the two [1–7]. The software includes different types of faults, and each fault requires different strategies and different amounts of testing effort to remove it.

Ohba [6] refined the Goel-Okumoto model by assuming that the fault detection/removal rate increases with time and that there are two types of faults in the software. SRGM proposed by Bittanti et al. [2] and Kapur and Garg [5] has similar forms as that of Ohba [6] but is developed under different set of assumptions. Bittanti et al. [2] proposed an SRGM exploiting the fault removal (exposure) rate during the initial and final time epochs of testing. Whereas, Kapur and Garg [5] describe a fault removal phenomenon, where they assume that during a removal process of a fault some of the additional faults might be removed without these faults causing any failure. These models can describe both exponential and S-shaped growth curves and therefore are termed as flexible models [2, 5, 6].

The systems with distributed computing improve performance of a computing system and individual users through parallel execution of programs, load balancing and sharing, and replication of programs and data. Ohba [6] proposed the Hyper-exponential SRGM, assuming that software consists of different modules. Each module has its characteristics and thus the faults detected in a particular module have their own peculiarities. Therefore, the Fault Removal Rate for each module is not the same. He suggested that the fault removal process for each module is modeled separately and that the total fault removal phenomenon is the addition of the fault removal process of all the modules. Kapur et al. [1] proposed an SRGM with three types of fault. The first type is modeled by an Exponential model of Goel and Okumoto [4]. The second type is modeled by Delayed S-shaped model of Yamada et al. [7]. The third type is modeled by a three-stage Erlang model proposed by Kapur et al. [1]. The total removal phenomenon is again modeled by the superposition of the three SRGMs [1, 8]. Later they extended their model to cater for more types of faults [9] by incorporating logistic rate during the removal process. We have used different forms of FDR used in Kapur et al. [9] while modeling our proposed SRGM.

A number of faults are detected and removed during the long-testing period before the system is released to the market. However, the users then find number of faults and the software company then releases an updated version of the system. Thus in this case the number of faults that remain in the system can be considered to be a stochastic process with continuous state space [10]. Yamada et al. [11] proposed a simple software reliability growth model to describe the fault detection process during the testing phase by applying Itô type Stochastic Differential Equation (SDE) and obtain several software reliability measures using the probability distribution of the stochastic process. Later on, they proposed a flexible Stochastic Differential Equation Model describing a fault-detection process during the system-testing phase of the distributed development environment [12]. Lee et al. [13] used SDEs to represent a per-fault detection rate that incorporate an irregular fluctuation instead of an NHPP, and consider a per-fault detection rate that depends on the testing time t.

In this paper, we will use SDEs to represent fault-detection rate that incorporate an irregular fluctuation. We consider a composite model called generalized SRGM that includes three different types of faults, for example, simple, hard, and complex. Fault detection rates for hard and complex faults are assumed to be time dependent that can incorporate learning as the testing progresses. In practice, it is more realistic to describe different rates for three different types of faults. This model can further be extended to n-type of faults.

For the estimation of the parameters of the proposed model, Statistical Package for Social Sciences (SPSS) is used. The goodness-of-fit of the proposed model is compared with NHPP-based Generalised Erlang Model [1, 8]. The proposed model provides significant improved goodness-of-fit results. The paper is organized as follows. Section 2 presents the model formulation for the proposed model. Sections 3 and 4 give the method used for parameter estimation and criteria used for validation and evaluation of the proposed model. We conclude the paper in Section 5.

2. Framework for Modeling

2.1. Notations for the Proposed SRGM using SDE

$(N(t))$: The number of faults detected during the testing time t and is a random variable.

$E(N(t))$: Expected number of faults detected in the time interval $(0, t]$ during testing phase.

a: Total fault content.

a_1, a_2, a_3: Initial fault content for simple, hard, and complex types of faults.

b_1, b_2, b_3: Fault detection rates for simple, hard, and complex faults.

$E(N_1(t))$, $E(N_2(t))$, $E(N_3(t))$: Mean number of fault for simple, hard, and complex faults.

σ_1, σ_2, σ_3: Positive constant that represents the magnitude of the irregular fluctuations for simple, hard, and complex faults.

$\gamma_1(t)$, $\gamma_2(t)$, $\gamma_3(t)$: Standardized Gaussian White Noise for simple, hard, and complex faults.

P_1, P_2, P_3: Proportion of simple, hard, and complex faults in total fault content of the software.

β: Constant parameter representing a learning phenomenon in the Fault Removal Rate function.

2.2. Assumptions for the Proposed SRGM using SDE

(1) The Software fault-detection process is modeled as a stochastic process with a continuous state space.

(2) The number of faults remaining in the software system gradually decreases as the testing procedure goes on.

(3) Software is subject to failures during execution caused by faults remaining in the software.

(4) The faults existing in the software are of three types: simple, hard, and complex. They are distinguished by the amount of testing effort needed to remove them.

(5) During the fault isolation/removal, no new fault is introduced into the system and the faults are debugged perfectly.

2.3. SDE Modeling for Different Categories of Faults

2.3.1. Framework for Modeling for Proposed SRGM

Several SRGMs are based on the assumption of NHPP, treating the fault detection process during the testing phase as a discrete counting process. Recently Yamada et al. [11] asserted that if the size of the software system is large then the number of the faults detected during the testing phase is also large and change in the number of faults, which are corrected and removed through each debugging, becomes small compared with the initial faults content at the beginning of the testing phase. So, in order to describe the stochastic behavior of the fault detection process, we can use a Stochastic Model with continuous state space. Since the latent faults in the software system are detected and eliminated during the testing phase, the number of faults remaining in the software system gradually decreases as the testing progresses. Therefore, it is reasonable to assume the following differential equation:

$$\frac{dN(t)}{dt} = r(t)[a - N(t)], \tag{2.1}$$

where $r(t)$ is a fault-detection rate per remaining fault at testing time t.

However, the behavior of $r(t)$ is not completely known since it is subject to random effects such as the testing effort expenditure, the skill level of the testers, and the testing tools and thus might have irregular fluctuation. Thus, we have

$$r(t) = b(t) + \text{noise}. \tag{2.2}$$

Let $\gamma(t)$ be a standard Gaussian white noise and σ a positive constant representing a magnitude of the irregular fluctuations. So (2.2) can be written as

$$r(t) = b(t) + \sigma\ \gamma(t). \tag{2.3}$$

Hence, (2.1) becomes

$$\frac{dN(t)}{dt} = \left[b(t) + \sigma\gamma(t)\right][a - N(t)]. \tag{2.4}$$

Equation (2.4) can be extended to the following stochastic differential equation of an Itô Type [10, 11]:

$$dN(t) = \left[b(t) - \frac{1}{2}\sigma^2\right][a - N(t)]dt + \sigma[a - N(t)]dw(t), \tag{2.5}$$

where $W(t)$ is a one-dimensional Wiener process, which is formally defined as an integration of the white noise $\gamma(t)$ with respect to time t. Use Itô formula solution to (2.5); and use initial condition $N(0) = 0$ as follows [10, 11]:

$$N(t) = a\left[1 - \exp\left\{-\int_0^t b(x)dx - \sigma W(t)\right\}\right]. \tag{2.6}$$

The Wiener process $W(t)$ is a Gaussian process and it has the following properties:

$$\begin{gathered}\Pr[w(0) = 0] = 1, \\ E[w(t)] = 0, \\ E\left[w(t)w(t')\right] = \min\left[t, t'\right].\end{gathered} \tag{2.7}$$

In this paper, we consider three different fault detection rates, that is, constant for simple and time dependent for hard and complex faults. In practical situation it has been observed that a large number of simple (trivial) faults are easily detected at the early stages of testing while fault removal may become extremely difficult in the later stages.

We now briefly describe the Generalised Erlang model with logistic error detection function. The proposed model is based on Generalised Erlang model with logistic error detection function and SDE as described below.

Generalized Erlang Model with Logistic Error Detection Function [9, 14–16]

The model assumes that the testing phase consists of three processes, namely, failure, observation, fault detection, and fault removal. The software faults are categorized into three types according to the amount of testing effort needed to remove them. The time delay between the failure observation and the subsequent fault removal is assumed to represent the testing effort. The faults are classified as simple if the time delay between the failure observation, fault detection and removal is negligible. For the simple faults, the fault removal phenomenon is modeled by the exponential model of Goel and Okumoto [4], that is,

$$m_1(t) = a_1\left(1 - e^{-b_1 t}\right). \tag{2.8}$$

It is assumed that the hard faults consume more testing effort when compared with simple faults. This means that the testing team will have to spend more time to analyze the cause of the failure and therefore requires greater efforts to remove them. Hence the removal process for such faults is modeled as a two-stage process. The first stage describes the failure observation process. The second stage of the two-stage process describes the delayed fault removal process. During this stage the fault removal rate is assumed to be time dependent. The reason for this assumption is to incorporate the effect of learning on the removal process. With each fault removal insight is gained into the nature of faults present and function described, called logistic function, can account for that. So its mean value function will be given by [9, 14–16]

$$m_2(t) = \frac{a_2\left[1 - \{1 + b_2 t\} e^{-b_2 t}\right]}{1 + \beta e^{-b_2 t}}. \tag{2.9}$$

There can be components still having harder faults or complex faults. These faults can require more effort for removal after isolation. Hence they need to be modeled with greater time lag between failure observation and removal. The first stage describes the failure observation process, the second stage describes the fault isolation process, and the third stage describes the fault removal process. During this stage the fault removal rate is assumed to be time dependent. Logistic learning function is used again to represent the knowledge gained by the removal team. Hence its mean value function will be given by [9, 14–16]

$$m_3(t) = \frac{a_3\left[1 - \left(1 + b_3 t + b_3^2 t^2/2\right) e^{-b_3 t}\right]}{1 + \beta e^{-b_3 t}}. \tag{2.10}$$

The total removal phenomenon is modeled by the superposition of the three NHPP, that is,

$$\begin{gathered} m(t) = m_1(t) + m_2(t) + m_3(t), \\ m(t) = a_1\left(1 - e^{-b_1 t}\right) + \frac{a_2\left[1 - (1 + b_2 t) e^{-b_2 . t}\right]}{1 + \beta e^{-b_2 t}} + \frac{a_3\left[1 - \left(1 + b_3 t + b_3^2 t^2/2\right) e^{-b_3 . t}\right]}{1 + \beta e^{-b_3 t}}, \end{gathered} \tag{2.11}$$

where $a_1 = ap_1$, $a_2 = ap_2$, and $a_3 = ap_3$, where $p_3 = (1 - p_1 - p_2)$.

From (2.8), (2.9), and (2.10), it has been observed that the removal rate per fault for simple faults is a constant b_1, whereas for hard and complex faults, these rates are function of time t and are given, respectively, by

$$\begin{gathered} b_2(t) = \frac{m_2'(t)}{a_2 - m_2(t)} = \frac{b_2(1 + \beta + b_2 t) - b_2\left(1 + \beta e^{-b_2 t}\right)}{(1 + \beta + b_2 t)\left(1 + \beta e^{-b_2 t}\right)}, \\ b_3(t) = \frac{m_3'(t)}{a_3 - m_3(t)} = \frac{b_3\left(1 + \beta + b_3 t + {b_3}^2 t^2/2\right) - b_3\left(1 + \beta e^{-b_3 t}\right)(1 + b_3 t)}{\left(1 + \beta + b_3 t + {b_3}^2 t^2/2\right)\left(1 + \beta e^{-b_3 t}\right)}. \end{gathered} \tag{2.12}$$

Note that $b_2(t)$ and $b_3(t)$ increases monotonically with time and tend to constants b_2 and b_3, respectively, as $t \to \infty$.

Proposed SRGM

Now in the proposed model considering the three forms of $b(t)$, that is, for simple represented by a constant FDR, hard and complex faults represented by time dependent FDR's, respectively, we have

$$\begin{gathered}
b_1(t) = b_1, \\
b_2(t) = \frac{b_2(1+\beta+b_2t) - b_2(1+\beta e^{-b_2t})}{(1+\beta+b_2t)(1+\beta e^{-b_2t})}, \\
b_3(t) = \frac{b_3\left(1+\beta+b_3t+{b_3}^2t^2/2\right) - b_3(1+\beta e^{-b_3t})(1+b_3t)}{\left(1+\beta+b_3t+{b_3}^2t^2/2\right)(1+\beta e^{-b_3t})}.
\end{gathered} \tag{2.13}$$

Now considering (2.6) and using the above form of $b(t)$ for different type of faults, we have the number of faults detected at testing time t given by the following expression for three types of faults:

$$\begin{gathered}
N_1(t) = a_1\left[1 - \left\{e^{-b_1t-\sigma_1W_1(t)}\right\}\right], \\
N_2(t) = a_2\left[1 - \frac{(1+\beta+b_2t)\{e^{-b_2t-\sigma_2W_2(t)}\}}{1+\beta\ e^{-b_2t}}\right], \\
N_3(t) = a_3\left[1 - \frac{(1+\beta+b_3t+b_3^2t^2/2)\{e^{-b_3t-\sigma_3W_3(t)}\}}{1+\beta\ e^{-b_3t}}\right].
\end{gathered} \tag{2.14}$$

Taking Expectation of $N_1(t)$, $N_2(t)$, and $N_3(t)$, respectively, we have

$$\begin{gathered}
E(N_1(t)) = a_1\left[1 - \left\{e^{-b_1t+{\sigma_1}^2t/2}\right\}\right], \\
E(N_2(t)) = a_2\left[1 - \frac{(1+\beta+b_2t)\left\{e^{-b_2t+{\sigma_2}^2t/2}\right\}}{1+\beta\ e^{-b_2t}}\right], \\
E(N_3(t)) = a_3\left[1 - \frac{(1+\beta+b_3t+b_3^2t^2/2)\left\{e^{-b_3t+{\sigma_3}^2t/2}\right\}}{1+\beta\ e^{-b_3t}}\right].
\end{gathered} \tag{2.15}$$

2.4. Modeling Total Fault Removal Phenomenon

Total fault removal phenomenon of the proposed model is the sum of mean removal phenomenon for simple, hard, and complex faults, that is,

$$E(N(t)) = E(N_1(t)) + E(N_2(t)) + E(N_3(t)). \tag{2.16}$$

This is the mean value function of superimposed removal phenomenon of simple, hard, and complex faults, respectively.

For proposed SRGM,

$$\begin{aligned} E(N(t)) = a_1\left[1-\left\{e^{-b_1t+\sigma_1^2t/2}\right\}\right] \\ + a_2\left[1-\frac{(1+\beta+b_2t)\left\{e^{-b_2t+\sigma_2^2t/2}\right\}}{1+\beta\ e^{-b_2t}}\right] \\ + a_3\left[1-\frac{(1+\beta+b_3t+b_3^2t^2/2)\left\{e^{-b_3t+\sigma_3^2t/2}\right\}}{1+\beta\ e^{-b_3t}}\right], \end{aligned} \tag{2.17}$$

where $a_1 = ap_1$, $a_2 = ap_2$, and $a_3 = ap_3$, where $p_3 = (1-p_1-p_2)$.

2.5. Software Reliability Measures

In this section, we present expression for various software reliability measures. Information on the current number of detected faults in the system is important to estimate the situation of the progress on the software testing procedures. Since it is a random variable in our models, so its expected value can be useful measures. We have already calculated the expected value for our models in (2.15).

Instantaneous MTBF for Proposed SRGM

The instantaneous MTBF (denoted by MTBF_I) is Average Time Between Failure in an interval dt. The instantaneous mean time between software failures is useful to measure the property of the frequency of software failure occurrence. The instantaneous MTBF for the proposed models is given by the following.

For simple faults,

$$(\text{MTBF})_\text{I} = \frac{1}{a_1(b_1-(1/2)\sigma^2)e^{-(b_1-(1/2)\sigma^2)t}}. \tag{2.18}$$

For hard faults,

$$(\text{MTBF})_\text{I} = \frac{1}{a_2\left[(1+\beta+b_2t)/(1+\beta e^{-b_2t})\right][\mathfrak{A}]e^{-(b_2-(1/2)\sigma_2{}^2)t}}, \tag{2.19}$$

where $\mathfrak{A} = (b_2(1+\beta+b_2t)-b_2(1+\beta e^{-b_2t}))/((1+\beta+b_2t)(1+\beta e^{-b_2t}))-(1/2)\sigma_2{}^2$.

For complex faults,

$$(\mathrm{MTBF})_{\mathrm{I}} = \frac{1}{a_3\left[\left(1+\beta+b_3t+{b_3}^2t^2/2\right)/\left(1+\beta e^{-b_3t}\right)\right]\left[\mathfrak{B}-(1/2){\sigma_3}^2\right]e^{-\left(b_3-(1/2){\sigma_3}^2\right)t}}, \tag{2.20}$$

where $\mathfrak{B}$ denotes $(b_3(1+\beta+b_3t+{b_3}^2t^2/2) - b_3(1+\beta e^{-b_3t})(1+b_3t))/((1+\beta+b_3t+{b_3}^2t^2/2)$ $(1+\beta e^{-b_3t}))$

Cumulative MTBF for Proposed SRGM

The cumulative MTBF is the Average Time Between Failure from the beginning of the test (i.e., $t = 0$) up to time t. We have the following cumulative mean time between software failures (denoted by $\mathrm{MTBF}_{\mathrm{C}}$) for the proposed models:

$$(\mathrm{MTBF})_{\mathrm{C}} = \frac{t}{E(N(t))}. \tag{2.21}$$

The cumulative MTBF of the model is given as follows.
Simple faults:

$$(\mathrm{MTBF})_{\mathrm{C}} = \frac{t}{a_1\left[1-\left\{e^{-b_1t+{\sigma_1}^2t/2}\right\}\right]}. \tag{2.22}$$

Hard faults:

$$(\mathrm{MTBF})_{\mathrm{C}} = \frac{t}{a_2\left[1-\left(1+\beta+b_2t\right)\left\{e^{-b_2t+{\sigma_2}^2t/2}\right\}/\left(1+\beta\ e^{-b_2t}\right)\right]}. \tag{2.23}$$

Complex faults:

$$(\mathrm{MTBF})_{\mathrm{C}} = \frac{t}{a_3\left[1-\left(1+\beta+b_3t+b_3^2t^2/2\right)\left\{e^{-b_3t+{\sigma_3}^2t/2}\right\}/\left(1+\beta\ e^{-b_3t}\right)\right]}. \tag{2.24}$$

3. Parameter Estimation

Parameter estimation and model validation are important aspects of modeling. The mathematical equations of the proposed SRGM are nonlinear. Technically, it is more difficult to find the solution for non-linear models using Least Square method and requires numerical algorithms to solve it. Statistical software packages such as SPSS help to overcome this problem. SPSS is a Statistical Package for Social Sciences. For the estimation of the parameters of the proposed model, Method of Least Square (Nonlinear Regression method) has been used. Nonlinear Regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables.

4. Comparison Criteria for SRGM

The performance of SRGM is judged by its ability to fit the past software fault data (goodness of fit).

4.1. Goodness of Fit Criteria

The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of "How good does a mathematical model (e.g., a linear regression model) fit to the data"?

(a) The Mean Square Fitting Error (MSE)

The model under comparison is used to simulate the fault data, the difference between the expected values, $\widehat{m}(t_i)$, and the observed data y_i is measured by MSE [1] as follows. MSE = $\sum_{i=1}^{k}((\widehat{m}(t_i) - y_i)^2/k)$, where k is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit.

(b) Coefficient of Multiple 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 subtracted from 1 [1], that is, $R^2 = 1 -$ residual SS/corrected SS. R^2 measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger R^2 is, the better the model explains the variation in the data.

(c) Prediction Error (PE)

The difference between the observation and prediction of number of failures at any instant of time i is known as PE_i. Lower the value of Prediction Error, better the goodness of fit [17].

(d) Bias

The average of PEs is known as bias. Lower the value of Bias, better the goodness of fit [17].

(e) Variation

The standard deviation of PE is known as variation. Variation = $\sqrt{(1/(N-1))\sum(\mathrm{PE}_i - \mathrm{Bias})^2}$. Lower the value of Variation, better the goodness of fit [17].

(f) Root Mean Square Prediction Error

It is a measure of closeness with which a model predicts the observation. RMSPE = $\sqrt{(\mathrm{Bias}^2 + \mathrm{Variation}^2)}$. Lower the value of Root Mean Square Prediction Error, better the goodness of fit [17].

Table 1: Schedule of release candidate version in Fedora core 7.

Date	Event
1 February 2007	Test 1 release
29 February 2007	Test 2 release
27 March 2007	Test 3 release
24 April 2007	Test 4 release
31 May 2007	Fedora 7 general availability

5. Model Validation

To check the validity of the proposed model and to find out its software reliability growth, it has been tested on three Data Sets. The Proposed Model has been compared with NHPP-based Generalised Erlang Model [1, 8]. For the proposed SRGM, the results are better for given data sets.

DS-I

This data is cited from Brooks and Motley (1980) [18]. The fault data set is for a radar system of size 124 KLOC (Kilo Lines of Code) tested for 35 weeks in which 1301 faults were removed. Parameters of SRGM (2.17) were estimated using SPSS software tool. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 2. The goodness of fit curve for DS-1 is given in Figure 1.

DS-II

This data is cited from Misra [19]. The software was tested for 38 weeks during which 2456.4 computer hours were used and 231 faults were removed. Parameters of SRGM (2.17) were estimated using SPSS software tool. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 3. The goodness of fit curve for DS-II is given in Figure 2. Values of p_1, p_2, and p_3 are computed from the actual data set since data was available separately for each type of fault.

DS-III

This data is cited from Fedora Core Linux [20, 21], which is one of the operating systems developed under an open source project. The Fedora project is made up of many small-sized projects. Fedora is a set of projects, sponsored by Red Hat and guided by the Fedora project board. These projects are developed by a large community of people, who strive to provide and maintain the very best in free, open source software and standards. The fault count data collected in this paper are collected in the bug tracking system on the website of Fedora project in May, 2007. The schedule of release candidate version in Fedora core 7 is shown as in Table 1.

In this paper, the test data for the end of Test 3 Release version is considered, where 164 faults were detected. The Parameter Estimation result and the goodness of fit results for the proposed SRGM are given in Table 4. The goodness of fit curve for DS-III is given in Figure 3.

The values of initial fault contents a_1, a_2, a_3 can be calculated from Tables 2, 3, and 4 for the given datasets, that is, DS-I, DS-II, and DS-III using $a_i = ap_i$; $i = 1, 2, 3$.

Table 2

(a) Parameter for DS-I (Brooks DS-2 1301 faults)

Models under comparisons	Parameter estimation										
	a	b_1	b_2	b_3	β	p_1	p_2	p_3	σ_1	σ_2	σ_3
Proposed SRGM	1339	.089	.248	.251	48	.264	.669	.067	.194	.001	.111
Generalized Erlang model [1, 8]	1453	.376	.000	.165	—	.011	.000	.989	—	—	—

(b) Parameter for DS-I

Models under comparison	Comparison criteria				
	R^2	MSE	Bias	Variation	RMSPE
Proposed SRGM	1.00	81.3734	−0.06975	9.20469	9.204957
Generalised Erlang model [1, 8]	.994	1200.522	0.939148	35.14148	35.15403

Table 3

(a) Parameter for DS-II (Misra 231 faults)

Models under comparisons	Parameter estimation										
	a	b_1	b_2	b_3	β	p_1	p_2	p_3	σ_1	σ_2	σ_3
Proposed SRGM	420	.059	.104	.378	66.593	.64	.342	.018	.048	.185	.599
Generalised Erlang model [1, 8]	561	.022	.012	.041	—	.64	.342	.018	—	—	—

(b) Parameter for DS-II

Models under Comparison	Comparison criteria				
	R^2	MSE	Bias	Variation	RMSPE
Proposed SRGM	.998	7.22	−0.7104	2.626231	2.720631
Generalised Erlang model [1, 8]	.995	22.09	0.6943	4.711796	4.762687

Table 4

(a) Parameter for DS-III

Models under comparisons	Parameter estimation										
	a	b_1	b_2	b_3	β	p_1	p_2	p_3	σ_1	σ_2	σ_3
Proposed SRGM	215	.189	.135	.113	8	.220	.640	.140	.328	.072	.346
Generalised Erlang model [1, 8]	195	.063	.011	.075	—	.212	.004	.784	—	—	—

(b) Parameter for DS-III

Models under comparison	Comparison criteria				
	R^2	MSE	Bias	Variation	RMSPE
Proposed SRGM	.998	5.88010	0.070583	2.541294	2.54227
Generalised Erlang model [1, 8]	.997	7.95905	0.149816	2.84224	2.84618

Description of Tables

Tables 2(a), 3(a), and 4(a) show the parameter estimates of proposed model and generalized Erlang model for data sets DS-I, DS-II, and DS-III, respectively. For data set DS-II, the proportions of different types of faults are given in the data set and for other data sets proportions of different types of faults are estimated. With the prior knowledge of proportion of different types of faults, programmer can act with better strategy for removing these faults.

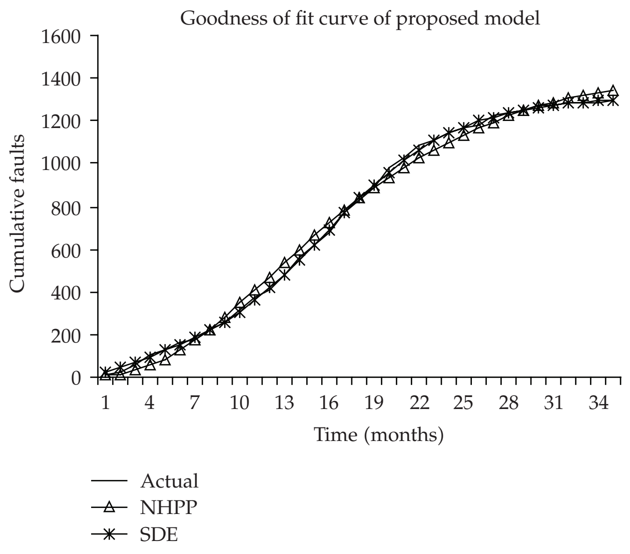

Figure 1: Goodness of Fit Curve for DS-I.

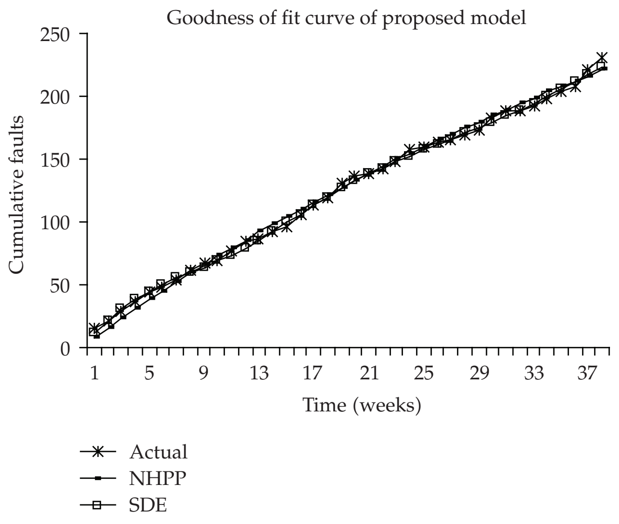

Figure 2: Goodness of Fit Curve for DS-II.

Tables 2(b), 3(b), and 4(b) describe the comparison criteria results for proposed model and generalized Erlang model. It is clear from the table that proposed model results are better in comparison with generalized Erlang model for different comparison criteria parameters.

Goodness of Fit Curves for DS-I, DS-II, and DS-III

The curves given in Figures 1 and 2 reflect the initial learning curve at the beginning, as test members become familiar with the software, followed by growth and then leveling off as the residual faults become more difficult to uncover.

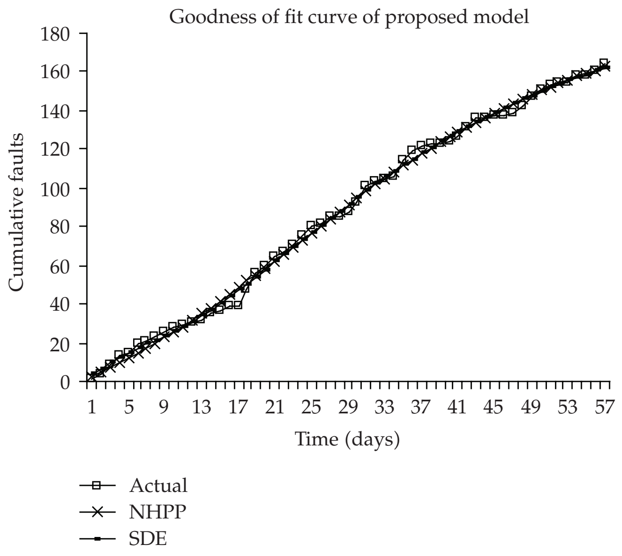

Figure 3: Goodness of Fit Curve for DS-III.

6. Conclusion

This paper presents an SRGM for different categories of faults based on Itô type Stochastic Differential Equations. In this paper, we have extended the SDE approach adopted by Yamada et al. [12] to the case where the faults are simple, hard, and complex in nature. The goodness of the fit analysis has been done on three real software failure datasets. The goodness-of-fit of the proposed Model is compared with NHPP-based Generalized Erlang model [1, 8]. The results obtained show better fit and wider applicability of the model to different types of failure datasets. From the numerical illustrations, we see that the Proposed Model provides improved results with better predictability because of lower MSE, Variation, RMSPE, Bias and higher R^2. The usability of SDE is not only restricted to the model described in this paper but it can also be extended to improve the results of any other SRGM. The Proposed Model can also be used by incorporating error generation and various Testing Effort functions.

Acronyms

MLE:	Maximum likelihood estimate
DS:	Data set
R^2:	Coefficient of multiple determination
SPSS:	Statistical package for social sciences
MSE:	Mean square error
PE:	Prediction error
RMSPE:	Root mean square prediction error
FDR:	Fault detection rate.

Acknowledgment

The first author acknowledges the financial support provided by the Defence Research and Development Organization, Ministry of Defence, Government of India under Project no. ERIP/ER/0703635/M/01/977.

References

[1] P. K. Kapur, R. B. Garg, and S. Kumar, *Contributions to Hardware and Software Reliability*, World Scientific, Singapore, 1999.

[2] S. Bittanti, P. Bolzern, E. Pedrotti, and R. Scattolini, "A flexible modeling approach for software reliability growth," in *Software Reliability Modelling and Identification*, G. Goos and J. Harmanis, Eds., pp. 101–140, Springer, Berlin, Germany, 1998.

[3] T. Downs and A. Scott, "Evaluating the performance of software-reliability models," *IEEE Transactions on Reliability*, vol. 41, no. 4, pp. 533–538, 1992.

[4] 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.

[5] 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.

[6] M. Ohba, "Software reliability analysis models," *IBM Journal of Research and Development*, vol. 28, no. 4, pp. 428–443, 1984.

[7] S. Yamada, M. Ohba, and S. Osaki, "S-shaped software reliability growth models and their applications," *IEEE Transactions on Reliability*, vol. 33, no. 4, pp. 289–292, 1984.

[8] P. K. Kapur, S. Younes, and S. Agarwala, "Generalised Erlang model with n types of faults," *ASOR Bulletin*, vol. 14, no. 1, pp. 5–11, 1995.

[9] P. K. Kapur, V. B. Singh, and B. Yang, "Software reliability growth model for determining fault types," in *Proceedings of the 3rd International Conference on Reliability and Safety Engineering (INCRESE '07)*, pp. 334–349, Reliability Center, Kharagpur, India, December 2007.

[10] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*, Universitext, Springer, Berlin, Germany, 6th edition, 2003.

[11] S. Yamada, A. Nishigaki, and M. Kimura, "A stochastic differential equation model for software reliability assessment and its goodness of fit," *International Journal of Reliability and Applications*, vol. 4, no. 1, pp. 1–11, 2003.

[12] 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, 2005.

[13] C. H. Lee, Y. T. Kim, and D. H. Park, "S-shaped software reliability growth models derived from stochastic differential equations," *IIE Transactions*, vol. 36, no. 12, pp. 1193–1199, 2004.

[14] P. K. Kapur, A. Gupta, A. Kumar, and S. Yamada, "Flexible software reliability growth models for distributed systems," *OPSEARCH*, vol. 42, no. 4, pp. 378–398, 2005.

[15] P. K. Kapur, O. Singh, A. Kumar, and S. Yamada, "Discrete software reliability growth models for distributed systems," *IEEE Transactions on Software Engineering*, communicated.

[16] P. K. Kapur, D. N. Goswami, A. Bardhan, and O. Singh, "Flexible software reliability growth model with testing effort dependent learning process," *Applied Mathematical Modelling*, vol. 32, no. 7, pp. 1298–1307, 2008.

[17] 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.

[18] W. D. Brooks and R. W. Motley, "Analysis of discrete software reliability models," Tech. Rep. RADC-TR-80-84, Room Air Development Center, New York, NY, USA, 1980.

[19] P. N. Misra, "Software reliability analysis," *IBM Systems Journal*, vol. 32, no. 3, pp. 262–270, 1983.

[20] Fedora Project, sponsored by Red Hat, http://fedoraproject.org.

[21] Y. Tamura and S. Yamada, "Optimal version-upgrade problem based on stochastic differential equations for open source software," in *Proceedings of the 5th International Conference on Quality and Reliability (ICQR '07)*, pp. 186–191, Chiang Mai, Thailand, 2007.