MethodsX 10 (2023) 102076

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Method Article

Multi-release software model based on testing coverage incorporating random effect (SDE)

Ritu Bibyan[a], Sameer Anand[b,*], Anu G. Aggarwal[a], Gurjeet Kaur[a]

[a] *Department of Operational Research, University of Delhi, New Delhi, India*
[b] *Shaheed Sukhdev College of Business Studies, University of Delhi, New Delhi, India*

ARTICLE INFO

Method name:
Testing Coverage incorporating SDE with Multi Release SRGM

Keywords:
Testing coverage
Software reliability growth models
Stochastic Differential Equation
Multi-release software
Imperfect debugging
Non-homogeneous poisson process (NHPP)
Method name
Testing coverage incorporating sde with multi release SRGM

ABSTRACT

In the past, various Software Reliability Growth Models (SRGMs) have been proposed using different parameters to improve software worthiness. Testing Coverage is one such parameter that has been studied in numerous models of software in the past and it has proved its influence on the reliability models. To sustain themselves in the market, software firms keep upgrading their software with new features or enhancements by rectifying previously reported faults. Also, there is an impact of the random effect on testing coverage during both the testing and operational phase. In this paper, we have proposed a Software reliability growth model based on testing coverage with random effect along with imperfect debugging. Later, the multi-release problem is presented for the proposed model. The proposed model is validated on the dataset from Tandem Computers. The results for each release of the models have been discussed based on the different performance criteria. The numerical results illustrate that models fit the failure data significantly.

- The random effect in the testing coverage rate is handled using Stochastic Differential Equations (SDE).
- Three testing coverage functions used are Exponential, Weibull, and S-shaped.
- Four Releases of the software model has been presented.

Specifications table

Subject area:	*Computer Science*
More specific subject area:	*Software Reliability*
Name of your method:	*Testing Coverage incorporating SDE with Multi Release SRGM*
Name and reference of the original method:	*None*
Resource availability:	*None*

Introduction

In today's world, the majority of people are well acquainted with software products as they play an important role in day-to-day life activities. The work is done efficiently by the software and at a very fast pace. The traditional methods consume lots of time whereas software products save time and make life smoother. Software products and systems have gained a significant place in various sectors including healthcare, science, information technology, and R&D field. Software tends to have a great impression on the safety-critical

* Corresponding author.
E-mail addresses: ritubibyan.du.or.17@gmail.com (R. Bibyan), sananddu@gmail.com (S. Anand), anuagg17@gmail.com (A.G. Aggarwal), gkaur@du.or.ac.in (G. Kaur).

https://doi.org/10.1016/j.mex.2023.102076
Received 25 November 2022; Accepted 9 February 2023
Available online 15 February 2023

spectrum like medical systems, biometric scans, communication, banking, air traffic control system, defense, nuclear plants, and so on (Pham 2007). Moreover, instant growth and specializations have made the working of software more complex. Software developers are always seeking to explore and develop more capable, stable, and efficient software. Due to this developing the software timely has become a considerable challenge for software-developing firms. Software needs a variety of features, which necessitates development procedures including fault detection and correction. And software failure also impacts the economy, consequently, users worry about software products' reliability. To handle this issue of the users, software engineers do rigorous testing before launching the software with the motive of providing the most reliable software to the market. The SRGMs (Software Reliability Growth Models) measure and analyzes the reliability of the software and hence play a vital role [43,44].

In history, various SRGMs have been studied which measure the number of remaining faults, reliability of the software, failure rate, failure intensity, etc. The most famous SRGMs in the research community are based on the NHPP (Non-Homogeneous Poisson Process [23,30,33,43]. The various traditional model based on time delay, correction process, the severity of faults, change point, and perfect debugging has been studied under a set of assumption [7,15,23,42]. These models were based on a single release i.e., there was no further changes were done once they were released into the software market. And, they also incorporated the concept of perfect debugging which means when the software fails, the testing team eliminates the faults with the assurance that no new faults were introduced during the process. It can't be possibly true because during the removal process, new faults are introduced and the testing team might have no knowledge about that. Many researchers have proposed studies based on imperfect debugging [16,20]. There are two scenarios of imperfect debugging namely, i) imperfect fault removal and ii) error generation. In an imperfect fault removal scenario, the fault amount remains the same which means the originally detected faults are eliminated without the generation of new faults. While in the error generation scenario, the total fault content rises during the testing process as the new faults are generated in the system when the original faults are removed. In 1985, the idea of imperfect debugging was first brought in [10]. Later, the concept of error generation was brought up which removed the assumption of conventional models that faults can be perfectly removed when detected [28]. Imperfect debugging models were introduced along with an exponential function of fault content or a linear function of testing time, an exponential function of error generation, and a time-dependent fault content function [32,33,45]. The fault removal efficiency model was discussed clearly by considering the existence of imperfect debugging along with error generation [22].

Moreover, researchers recommended that the efficiency of the growth models can be enhanced by taking practical issues under consideration while testing [3,14]. In the past, various testing coverage functions based on time have been introduced like Rayleigh [40], Weibull and logistic [13], log-exponential [26], s-shaped [34], beta & hyper exponential [3] and lognormal [31]. Testing coverage is thought to be one of the valuable attributes to identify the effectiveness of the software ([4]; Subhashis [5]; S [6]). None of these SRGMs have considered the testing coverage with random effects. The random effect is an important factor that impacts the failure rate, fault removal efficiency, testing coverage, change point, and so on. The random effect can be studied by the application of Itô type of Stochastic Differential equation [17–19,36,37]. The present study attempts to introduce a model by not only incorporating error generation but also testing coverage that is subjected to random effects. We have considered three cases of testing coverage functions. In case-1, we have considered the Exponential function in which the fault detection rate is constant which takes care of the case when the code is uniform in nature. The faults in the code are simple and as soon as they are detected they are isolated (Amrit L [11]). In the exponential testing coverage function, the failure intensity rate shows decreasing pattern while testing. It implies that as the testing progresses the quality of the software is enhanced. Moreover, in realistic situations, the failure intensity initially increases and later decreases. To handle this situation a generalized G-O model [8,9] was projected known as the Weibull model. Case-2, of our model, is the Weibull testing coverage function which inflates the time-varying detection rate. In case 3, we have considered the S-shaped testing coverage function. It identifies the authenticity of the time delay between fault detection and removal. The two phases of the testing process are fault detection and isolation [43].

One of the issues faced by software engineers is when to release the software in the market. Software firms need multiple releases of the software to sustain in the market because of various reasons explained in Fig. 1. The reliability of software can only be improved by improvising new features and upgrading the current version of the software. Hence, nowadays multi upgradation takes place whenever it is required based on customer demand by the developing department. This upgradation can also initiate many new faults [21,24,25]. The fault reduction factor (FRF) also plays a vital role in software growth modeling. A model with a time variable fault reduction factor was incorporated with the imperfect debugging to release multiple versions of the software in the market (Anu Gupta [1]). Later FRF was modeled using a delayed S-shaped model keeping factors like error generation constant [39]. The FRF and testing coverage-based SRGM was proposed along with the idea of the multi-release of an open-source software project [38]. A release time problem was suggested by using Fuzzy Analytical Hierarchy Process (FAHP) to find the weights of the modules. The two SRGMs were modeled by considering the effect of testing Coverage and FRF (Anu G [2]). A multi-release SRGM was addressed by assuming software failure distribution as generalized modified Weibull distribution [35]. Most of the SRGMs were given under the software development environment's preset parameters, which are quite predictable. However, when the program is released, the field operating environment is highly unpredictable and random. Considering the random field operating environment, two multi-upgrade SRGMs were introduced to seize the uncertainty of fault detection rate [27]. We have also incorporated multi-release problems in our proposed model that considers imperfect debugging along with testing coverage with random effects. The random effect causes irregular fluctuation which is modeled through SDE.

The paper is organized into different sections as follows: section 2 of the paper elaborates on the model framework. The sections 2.1 and 2.2, we discuss the notations used in our study and the assumptions of the model. In section 2.3, we deal with the development of the proposed model using three different testing coverage: Exponential. Weibull, S-shaped Testing Coverage. The section 3, we propose a multi-release framework for our proposed model. While section 4, provides a numerical example for pro-

Problem with present release
Requirement of new hardware to update software
Market competition
Different company policies
Requirement of different functionalities
Why need for different releases of software?

Fig. 1. Need for Different Releases of Software.

posed models on Tandem Computers dataset with parameter estimation and performance analysis. Finally, conclusions are drawn in Section 5.

Model framework

Notations

$N(t)$: a random variable that represents the number of faults disclosed during testing time t
$m(t)$: Mean value function giving an anticipated number of faults disclosed or removed by time t.
a : total number of expected faults existing in the system before testing,
$c(t)$: testing coverage representing the percentage of potential faults disclosed up to time t,
$c'(t)$: coverage rate i.e., the rate at which remaining possible faults are disclosed,
$r(t)$: fault removal rate,
$\lambda(t)$: failure intensity rate, b: failure occurrence rate per fault,
σ : represents the magnitude of random effects.
$\gamma(t)$: Standard Gaussian White Noise

Assumptions

The assumptions for the proposed SRGM are given as follows:

1. The fault detection process is modeled as a stochastic process.
2. The rate at which fault is detected is proportional to the number of faults remaining in the software at time t.
3. The software is exposed to failure during decapitation induced by remaining faults.
4. The faults detection rate can be expressed in the form of testing coverage as $\frac{c'(t)}{1-c(t)}$.
5. There is a presence of a random effect on testing coverage.
6. The new faults are introduced during the removal process with probability α.

Model formulation

The majority of the proposed software reliability model considers fault detection during both the phases of testing and the operational counting process. Furthermore, if the software system is large, the number of faults found in the course of testing increases, and the number of faults found and fixed by debugging decreases to a significant degree in comparison to the fault content at the start of the testing period. We may thus represent the software fault detection process in terms of testing coverage in such a scenario as a stochastic process with a continuous state space. [29]. Let $N(t)$ be a random variable that represents the number of faults disclosed during testing time t. The testing coverage is denoted as $c(t)$ and $(1-c(t))$ represents the proportion of code that has not been captured up to time t. The fault detection rate can be denoted as $\frac{c'(t)}{1-c(t)}$. The differential equation testing coverage and fault removal efficiency are given by:

$$\frac{dN(t)}{dt} = r(t)\ [a(t) - N(t)] \tag{1}$$

where $r(t)$ is the fault detection rate in terms of testing coverage and is subjected to some random effects, and $a(t)$ is the fault content function which is a linear function of expected faults detected up to time t so that we have

$$r(t) = \frac{c^{'}(t)}{1-c(t)} + \varepsilon noise\varepsilon = \frac{c^{'}(t)}{1-c(t)} + \sigma\gamma(t) \tag{2}$$

$$a(t) = a + \alpha\ N(t) \tag{3}$$

Hence, by incorporating Eq. (2) & (3) in (1), we have

$$\frac{dN(t)}{dt} = \left(\frac{c'(t)}{1-c(t)} + \sigma\gamma(t)\right)[a + \alpha\ N(t) - N(t)] \tag{4}$$

Now, expanding Eq. (4) to the Stochastic differential equation of an Itộ type [29,36], we have

$$dN(t) = \left[\frac{c'(t)}{1-c(t)} + \frac{1}{2}\sigma^2\right][a-(1-\alpha)N(t)]dt + \sigma[a-(1-\alpha)N(t)]dw(t) \tag{5}$$

where $W(t)$ is one - dimension Wiener process which is the integration of white noise $\gamma(t)$ with respect to time t. Also, wiener process $W(t)$ is a Gaussian process that has the following properties:

$$\Pr[w(0) = 0] = 1$$

$$E[w(t)] = 0$$

$$E[w(t)w(t')] = \min[t, t']$$

Here, by using the Itộ formula, we can achieve the solution process:

$$E(N(t)) = m(t) = \frac{a}{(1-\alpha)}\left[1 - \exp\left\{-\int_0^t \frac{c'(x)}{1-c(x)}dx + \frac{\sigma^2 t}{2}\right\}\right] \tag{6}$$

In this paper, we consider three different coverage functions [12], that is, Exponential Testing Coverage, Weibull Testing Coverage, and S-Shaped Testing Coverage Function as shown in Fig. 2. We now briefly describe three testing coverage functions based on the proposed model incorporating random effects in testing coverage and a linear function of the expected number of faults up to time t.

i) Model-1: Exponential testing Coverage

The traditional Goel Okumoto Model (Amrit L [11]), has had a substantial impact on the reliability modeling of software. It is one of the basic models with parameters having physical understanding. The model states that the failure process is modeled by NHPP, with MVF as

$$m(t) = a\left(1 - e^{-bt}\right)$$

$$\lambda(t) = abe^{-bt}$$

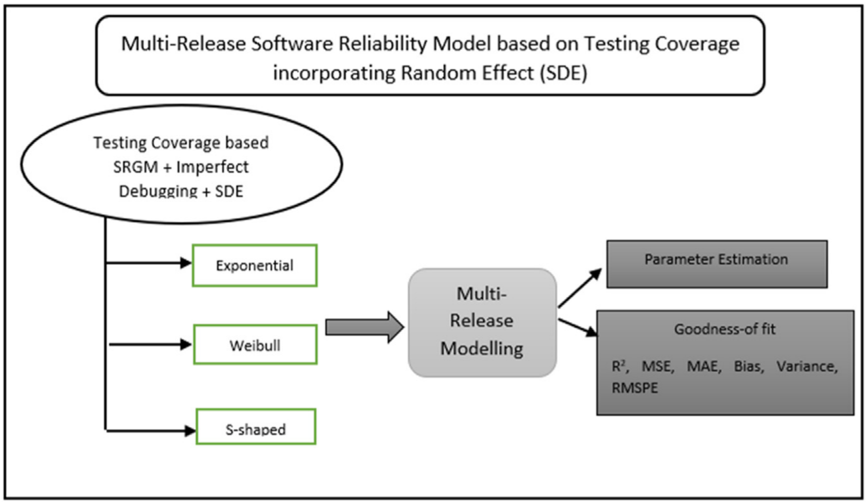
Multi-Release Software Reliability Model based on Testing Coverage incorporating Random Effect (SDE)
Testing Coverage based SRGM + Imperfect Debugging + SDE
Exponential
Weibull
S-shaped
Multi-Release Modelling
Parameter Estimation
Goodness-of fit
R², MSE, MAE, Bias, Variance, RMSPE

Fig.2. Structure of the Proposed Model.

The Coverage function and failure detection rate are given as

$$c(t) = 1 - e^{-bt}$$

$$r_*(t) = \frac{c'(t)}{1 - c(t)} = b \tag{7}$$

Now, we incorporate Eq. (7) in the proposed model Eq. (6), and we get

$$m_1(t) = \frac{a}{(1-\alpha)}\left[1 - exp\left(-\int_0^t bdx + \frac{\sigma^2 t}{2}\right)\right]$$

$$m_1(t) = \frac{a}{(1-\alpha)}\left[1 - exp\left(-bt + \frac{\sigma^2 t}{2}\right)\right] \tag{8}$$

ii) **Model-2: Weibull Testing Coverage**

In the exponential testing coverage function, the failure intensity rate shows decreasing pattern while testing. It implies that as the testing progresses the quality of the software is enhanced. Moreover, in realistic situations, the failure intensity initially increases and later decreases. To handle this situation a generalized G-O model [8,9] projected the mean value function as:

$$m(t) = a\left(1 - e^{-bt^{\gamma}}\right)$$

Here, b and γ are the quality parameters of testing

$$\lambda(t) = ab\gamma e^{-bt^{\gamma}} t^{\gamma-1}$$

The Coverage function and failure detection rate are given as

$$c(t) = 1 - e^{-bt^{\gamma}}$$

$$r_*(t) = \frac{c'(t)}{1 - c(t)} = b\gamma t^{\gamma-1} \tag{9}$$

Eq. (9) implies that the Weibull testing coverage function inflates the time-varying detection rate. The $r_*(t)$ is increasing function of time ‘t’ when $\gamma > 1$, decreasing when $\gamma < 1$, and constant when $\gamma = 1$.

Now, we incorporate Eq. (9) in the proposed model Eq. (6), and we get

$$m_2(t) = \frac{a}{1-\alpha}\left[1 - exp\left(-\int_0^t b\gamma x^{\gamma-1} dx + \frac{\sigma^2 t}{2}\right)\right]$$

$$m_2(t) = \frac{a}{1-\alpha}\left[1 - exp\left(-bt^{\gamma} + \frac{\sigma^2 t}{2}\right)\right] \tag{10}$$

iii) **Model-3: S-Shaped Testing Coverage**

The S-shaped SRGM identifies the authenticity of the time delay between fault detection and removal. The two phases of the testing process are fault detection and isolation. The mean value function is given as:

$$m(t) = a\left[1 - (1 + bt)e^{-bt}\right]$$

$$\lambda(t) = b^2 te^{-bt}$$

The Coverage function and failure detection rate are given as:

$$c(t) = 1 - (1 + bt)e^{-bt}$$

$$r_*(t) = \frac{c'(t)}{1 - c(t)} = \frac{b^2 t}{1 + bt} \tag{11}$$

Now, we incorporate Eq. (11) in the proposed model Eq. (6), and we get

$$m_3(t) = \frac{a}{1-\alpha}\left[1 - exp\left(-\int_0^t \frac{b^2 x}{1 + bx} dx + \frac{\sigma^2 t}{2}\right)\right]$$

$$m_3(t) = \frac{a}{1-\alpha}\left[1 - (1 + bt)\ exp\left(-bt + \frac{\sigma^2 t}{2}\right)\right] \tag{12}$$

A generalized way of representing the above models can be given as:

$$m(t) = \frac{a}{(1-\alpha)} F(t)$$

Therefore, the MVF given by Eqs. 8, 10, and 12 can be expressed as:

$$m_1(t) = \frac{a}{(1-\alpha)} F_1(t),\ where\ F_1(t) = \left[1 - exp\left(-bt + \frac{\sigma^2 t}{2}\right)\right] \tag{13}$$

$$m_2(t) = \frac{a}{1-\alpha} F_2(t),\ where\ F_2(t) = \left[1 - exp\left(-bt^{\gamma} + \frac{\sigma^2 t}{2}\right)\right] \tag{14}$$

$$m_3(t) = \frac{a}{1-\alpha} F_3(t),\ where F_3(t) = \left[1 - (1+bt)\ exp\left(-bt + \frac{\sigma^2 t}{2}\right)\right] \tag{15}$$

Multi-release modelling framework

Release 1:

The competition in the software market has driven the firms to release multiple variants of the software i.e., upgradation of software. The software is upgraded by adding new features for enhancement purpose to exist in the market. When the software is released for the first time in the market, the MVF is given by

$$m_{1j}(t) = \frac{a_{1j}}{1-\alpha_{1j}} F_{1j}(t),;\ 0 \le t \le t_1 \tag{16}$$

Where $m_{1j}(t)$ is the MVF of the first release for the j^{th} model and $F_{1j}(t)$ is the cumulative failure function, and is t_1 is the time when software is first released in the market.

Release 2:

The software is released for the second time due to feature enhancement made based on responses by the users on the previous release. The second release contains detected faults from the current release and some faults from the previous release that were not removed and are reported in the operational phase of Release 1. The testing is stopped at the time t_2 and the second version of the software is released.

$$m_{2j}(t) = \frac{a_{2j}}{1-\alpha_{2j}} F_{2j}\left(t-t_1\right) + \frac{a_{1j}}{1-\alpha_{1j}}(1 - F_{1j}\left(t_1\right))F_{2j}\left(t-t_1\right); t_1 < t \le t_2 \tag{17}$$

Release 3:

Similarly, release three contains detected faults from the present release and some faults from the previous release. The third version of the software is released at the time t_3 and MVF is given as:

$$m_{3j}(t) = \frac{a_{3j}}{1-\alpha_{3j}} F_{3j}\left(t-t_2\right) + \frac{a_{2j}}{1-\alpha_{2j}}(1 - F_{2j}\left(t_2-t_1\right))F_{3j}\left(t-t_2\right) + \frac{a_{1j}}{1-\alpha_{1j}}(1 - F_{1j}\left(t_1\right))(1 - F_{2j}\left(t_2-t_1\right))F_{3j}\left(t-t_2\right);\ t_2 < t \le t_3 \tag{18}$$

Release 4:

The fourth release contains detected faults from the present release and some faults from the previous release. The fourth version of the software is released at the time t_4 and MVF is given as:

$$\begin{aligned} m_{4j}(t) = {} & \frac{a_{4j}}{1-\alpha_{4j}} F_{4j}\left(t-t_3\right) + \frac{a_{3j}}{1-\alpha_{3j}}\left(1 - F_{3j}\left(t_3-t_2\right)\right)F_{4j}\left(t-t_3\right) + \frac{a_{2j}}{1-\alpha_{2j}}(1 - F_{2j}\left(t_2-t_1\right))\left(1 - F_{3j}\left(t_3-t_2\right)\right)F_{4j}\left(t-t_3\right) \\ & + \frac{a_{1j}}{1-\alpha_{1j}}(1 - F_{1j}\left(t_1\right))(1 - F_{2j}\left(t_2-t_1\right))\left(1 - F_{3j}\left(t_3-t_1\right)\right)F_{4j}\left(t-t_3\right); t_3 < t \le t_4 \end{aligned} \tag{19}$$

Numerical example

Parameter estimation, performance analysis

This section of the paper presents the estimation of the parameters of the proposed models using a tandem computer dataset [41] for four releases. In this paper, we estimate the unknown parameters using the Least Square Estimation method in SPSS. The data statistics for the tandem dataset have been provided in Table 1.Now, based on the data provided as given in Table 1, the efficiency of the model is validated using the goodness of fit criteria such as R^2 (Coefficient of determination), MSE, MAE, Bias, Variation, and RMSPE. The mathematical formulae for the criteria are provided in Table 2 along with their interpretation.

Table 3 provides the estimation results based on the three proposed models given by equations Eq. (13), (14) & (15) for four releases. The actual observed fault content for four releases of Tandem data sets is 100, 120, 60, and 42 respectively. On observing the estimated values of three models for four releases, we can interpret that the deviation between the predicted and actual value of fault content “a” is less. Thereby, it indicates that the proposed modeling framework fits the data well which is further supported by the Goodness of fit analysis.

Table 1
Tandem Computers dataset.

Tandem Dataset	Test duration in weeks	Cumulative number of faults
Release 1	20	100
Release 2	19	120
Release 3	12	60
Release 4	19	42

Table 2
Performance Criteria.

CRITERIA	DESCRIPTION
R^2	It measures the proportion of variation in the dependent variables. The value lies between 0 to 1and the larger the value the better the fit. $R^2 = 1 - \frac{\sum_{i=1}^{k} (m_i - \widehat{m}(t_i))^2}{\sum_{i=1}^{k} (m_i - \sum_{j=1}^{k} m_j/n)^2}$
MSE	The deviation between actual and estimated values is calculated using MSE. The lesser the value better is the goodness of fit. $MSE = \frac{\sum_{i=1}^{k} (m_i - \widehat{m}(t_i))^2}{k-p}$
MAE	The mean absolute error provides a deviation of absolute values. The lesser the value better is the goodness of fit $MAE = \frac{\sum_{i=1}^{k} \lvert(m_i - \widehat{m}(t_i))\rvert}{k-p}$
Bias	The Bias is given by the average of prediction error which is the difference between actual and estimated values at any time t. The lesser the value better is the goodness of fit $Bias = \frac{\sum_{i=1}^{k} \lvert\widehat{m}(t_i) - m_i\rvert}{k-p}$
Variance	The variance is given by the standard deviation of the prediction error. $Variance = \sqrt{\frac{\sum_{i=1}^{k} (m_i - \widehat{m}(t_i) - Bias)^2}{k-1}}$
Root Mean Square Percentage Error	The RMSPE measures the percentage error. It measures the closeness with which the model predicts the values. The lesser the value better is the goodness of fit. $RMSPE = \sqrt{Variance^2 + Bias^2}$

Table 3
Estimation results.

Release	Case	a	b	α	σ	γ
1	Exponential	101.496	0.391	0.22	0.785	-
	Weibull	101.847	0.077	0.141	0	1.11
	S-shaped	101.089	0.467	0.141	0.002	-
2	Exponential	128.09	0.061	0.19	0.003	-
	Weibull	120.217	0.042	0.024	2.5E-07	1.438
	S-shaped	122.446	0.305	0.121	0.00073	-
3	Exponential	63.727	0.37	0.362	0.002	-
	Weibull	61.957	0.022	0.001	2.4E-07	2.12
	S-shaped	57.082	0.109	0.003	9.2E-07	-
4	Exponential	41.256	0.052	0.063	0.0025	-
	Weibull	39.422	0.024	0.082	3.6E-07	1.656
	S-shaped	44.047	0.082	0.199	6.4E-08	-

The goodness of fit analysis

Table 4 provides the goodness of fit results for our proposed models. As mentioned in Table 2, the R^2 value lies between 0 to 1, and the larger the value the better the fit. The R^2 value of the S-shaped testing coverage-based SRGM model is close to 1 when compared to other models i.e., the S-shaped model fits the data well. The performance criteria values of the s-shaped testing coverage model also show better results for all releases. We plot the curves of the goodness-of-fit based on observed and estimated faults for the four releases of the tandem dataset (Fig.3–6). In the graph, the x-axis presents the testing time in weeks and the y-axis presents the cumulative number of faults. The boxplots for absolute residual have been shown in Fig. 7–10 to explain the spreading of the predicted and observed values of the dataset for all releases corresponding to the proposed 3 models.

Table 4
Goodness of fit results.

Release	Testing Coverage Cases	R^2	MSE	MAE	Bias	Variation	RMPSE
1	Exponential	0.986	11.65167	3.088	-0.2652	3.337131	3.347638
	Weibull	0.987	10.93034	2.699	-0.0751	3.229765	3.230636
	S-shaped	0.997	10.86576	2.6995	-0.0175	3.128690	3.043367
2	Exponential	0.997	23.47873	4.366316	0.10421	4.981716	4.982805
	Weibull	0.993	20.14181	4.168421	3.40211	7.610041	8.335894
	S-shaped	0.998	9.415642	2.210526	-0.4326	3.245222	3.273929
3	Exponential	0.991	3.066927	1.31425	-0.6159	2.241789	2.258940
	Weibull	0.991	4.166592	1.569167	-0.9575	2.744691	2.909061
	S-shaped	0.996	3.0952	1.4441	-0.4825	2.138581	2.289990
4	Exponential	0.993	4.472495	1.878947	0.4277	2.302029	2.341363
	Weibull	0.995	1.354211	0.947368	-0.6158	1.602949	1.711562
	S-shaped	0.998	0.8904	0.687368	-0.0168	0.969930	0.970077

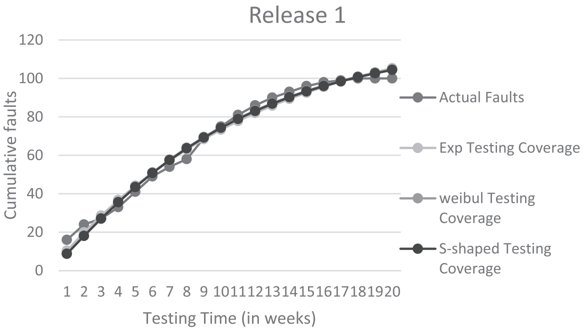

Fig. 3. Goodness-of-fit curve for Release 1.

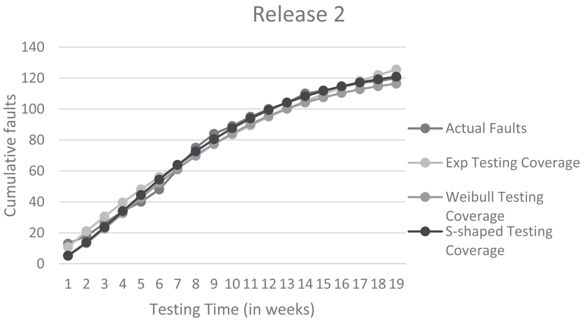

Fig. 4. Goodness-of-fit curve for Release 2.

The absolute residual for the given dataset is calculated as

$$Absolute\ Residual = \left|pred_j - obs_j\right|\ for\ j = 1, 2, \dots, n$$

where n represents the number of data values, $pred_j$ is the predicted value of the j^{th}data value and obs_j is the j^{th}observed or actual value of the dataset. It is said that a lesser absolute value provides a compact boxplot which signifies a better fit. The boxplot spread indicates that predicted faults have higher mismatches with the observed faults whereas the compact boxplot indicates the opposite. The data concentration at the lower or upper end is represented using skewness. The data points which are out of the range are called outliers and can be handled. Using a boxplot, we can easily recognize these points as most of the performance criteria don't acknowledge these points. In turn, it affects the efficiency of the results. The boxplots have been plotted for each release corresponding to each model in Fig. 5–8**.**

To describe the boxplot for our dataset briefly, consider the Release 3 boxplot (Fig. 7) for all the models. It can be seen that more area is enclosed by boxplot for the models with Exponential and s-shaped testing coverage functions when compared with the Weibull model. It implies that including random effects in fault detection rate based on testing coverage with imperfect debugging provides efficient accuracy to the model. There are no outliers in the boxplots and the median lies in the lower part. This signifies better performance as most of the residuals obtained are small. Similarly, the boxplots for other releases can be studied.

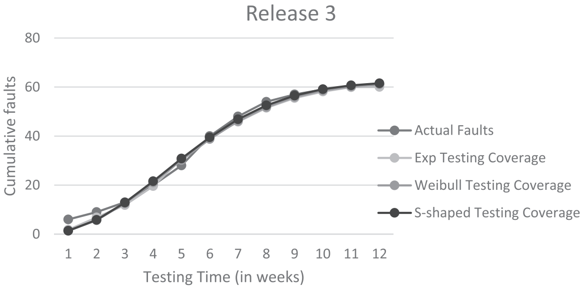

Fig.5. Goodness-of-fit Curve for Release 3.

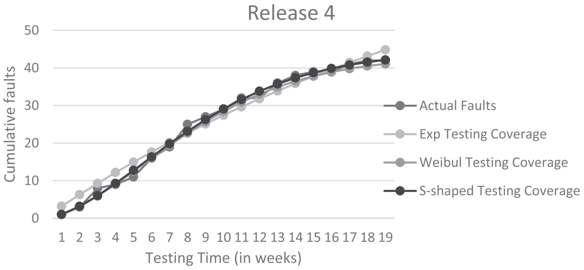

Fig.6. Goodness-of-fit Curve for Release 3.

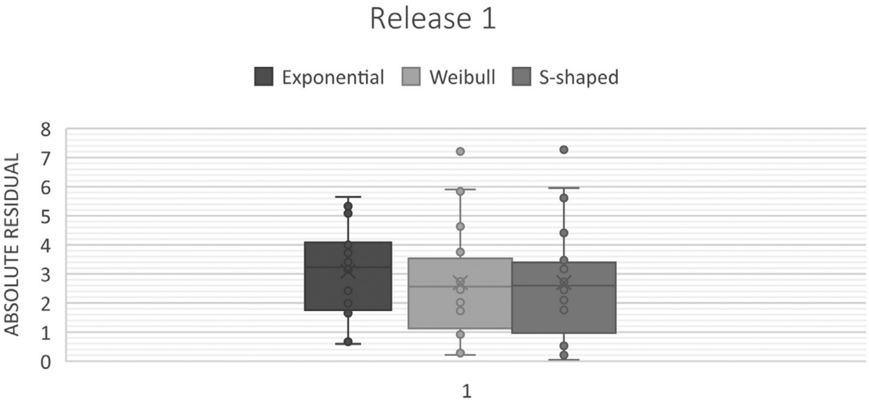

Fig. 7. Release 1 Boxplot for the proposed models.

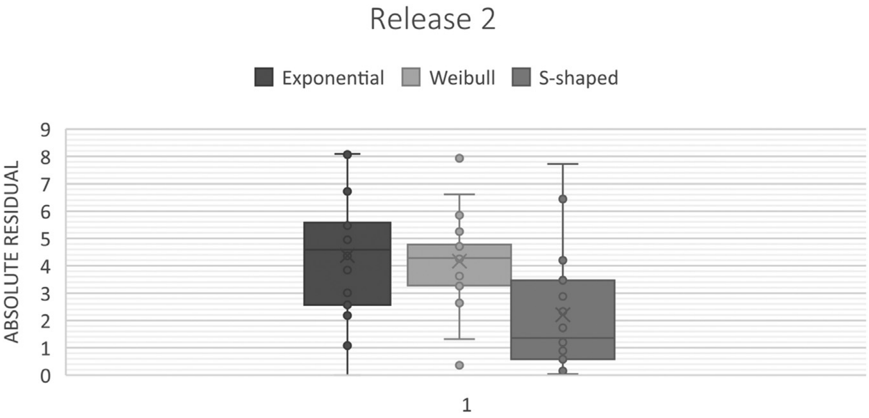

Fig. 8. Release 2 Boxplot for the proposed models.

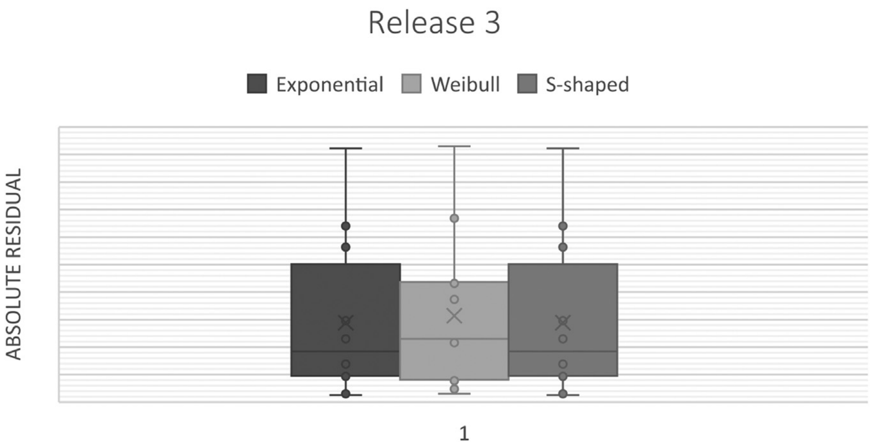

Fig. 9. Release 3 Boxplot for the proposed models.

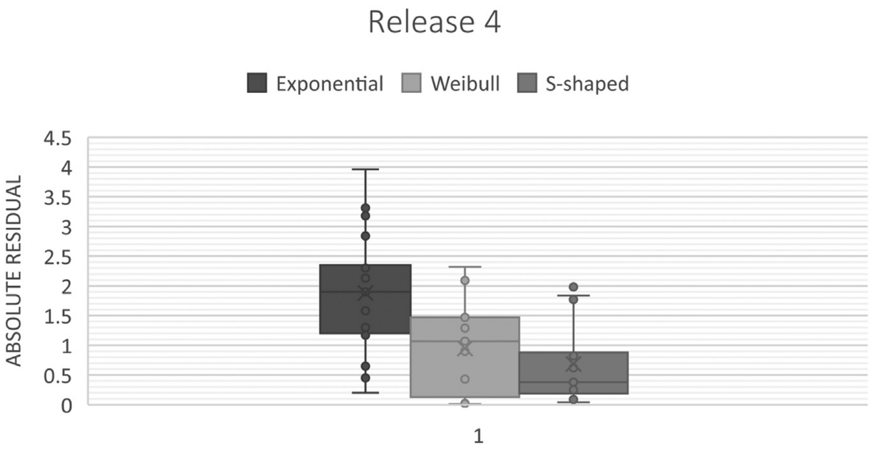

Fig. 10. Release 4 Boxplot for the proposed models.

Conclusion

In literature, various SRGMs have been proposed to improve the reliability and estimate the faults of the software system. The majority of the proposed software reliability model considers fault detection during both the phases of testing and the operational counting process. Furthermore, if the software system is large, the number of faults found in the course of testing increases, and the number of faults found and fixed by debugging decreases to a significant degree in comparison to the fault content at the start of the testing period. We may thus represent the software fault detection process in terms of testing coverage in such a scenario as a stochastic process with a continuous state space. The change in fault detection rate due to random effect causes irregular fluctuation which is studied using SDE in our models. We have proposed Testing Coverage based SRGM using SDE along with the multi-release. Also, we have considered three different testing coverage functions namely exponential, Weibull, and S-shaped. We have considered the Exponential function in which the fault detection rate is constant which takes care of the case when the code is uniform in nature. The faults in the code are simple and as soon as they are detected they are isolated. In the exponential testing coverage function, the failure intensity rate shows decreasing pattern while testing. It implies that as the testing progresses the quality of the software is enhanced. Moreover, in realistic situations, the failure intensity initially increases and later decreases. To handle this situation a generalized G-O model is projected known as the Weibull model. Weibull testing coverage function inflates the time-varying detection rate. We have also considered the S-shaped testing coverage function. It identifies the authenticity of the time delay between fault detection and removal. The two phases of the testing process are fault detection and isolation.

Later. Multi-release models have also been briefly discussed. We have analyzed our proposed models on the Tandem Computers failure dataset [41] from the literature. The results obtained from the numerical illustration in the paper show significant estimation potential of the models for each release. Also, the statistical performance criteria have been evaluated namely R^2, Mean Square Error, Mean Absolute Error, Bias, Variance, and Root Mean Square Percentage Error. The goodness-of-fit curves for the proposed models for four releases have been observed and they depict that model fits the data significantly. To obtain more insight, boxplots of residual values have been shown. We observe that all releases of the S-shaped model fit the data perfectly based on goodness of fit. More area is enclosed by boxplot for the models with Exponential and s-shaped testing coverage functions when compared with the Weibull model. It implies that including random effects in fault detection rate based on testing coverage with imperfect debugging provides efficient accuracy to the model.

The proposed models will give various aspects of extensions. In the future, a modeling framework can be generated by discussing the idea of change points and testing coverage rates with random effects. To study the imperfect debugging problem in our study, we have used the linear function for the fault content function. Later, research can be done using better forms to model imperfect debugging.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Have taken dataset from another research paper

References

[1] A.G. Aggarwal, N. Gandhi, V. Verma, A. Tandon, Multi-release software reliability growth assessment: an approach incorporating fault reduction factor and imperfect debugging, Int. J. Math. Oper. Res. 15 (4) (2019) 446–463.

[2] A.G. Aggarwal, A. Jaiswal, Multi-objective Release Time Problem for Modular Software using Fuzzy Analytical Hierarchy Process, in: Optimization Models in Software Reliability, Springer, 2022, pp. 159–191.

[3] Cai, X., & Lyu, M.R. (2007). *Software reliability modeling with test coverage: experimentation and measurement with a fault-tolerant software project.* Paper presented at the The 18th IEEE International Symposium on Software Reliability (ISSRE'07).

[4] I.H. Chang, H. Pham, S.W. Lee, K.Y. Song, A testing-coverage software reliability model with the uncertainty of operating environments, International Journal of Systems Science: Operations & Logistics 1 (4) (2014) 220–227.

[5] S. Chatterjee, A. Shukla, Effect of Test Coverage and Change Point on Software Reliability Growth Based on Time Variable Fault Detection Probability, J. Softw. 11 (1) (2016) 110–117.

[6] S. Chatterjee, J. Singh, A NHPP based software reliability model and optimal release policy with logistic–exponential test coverage under imperfect debugging, International Journal of System Assurance Engineering and Management 5 (3) (2014) 399–406.

[7] K.-.C. Chiu, Y.-.S. Huang, T.-.Z. Lee, A study of software reliability growth from the perspective of learning effects, Reliab. Eng. Syst. Saf. 93 (10) (2008) 1410–1421.

[8] Goel, A.L. (1982). *Software Reliability Modelling and Estimation Techniques*. Retrieved from

[9] Goel, A.L. (1983). *A Guidebook for Software Reliability Assessment*. Retrieved from

[10] A.L. Goel, Software reliability models: assumptions, limitations, and applicability, IEEE Trans. Software Eng. (12) (1985) 1411–1423.

[11] A.L. Goel, K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, IEEE Trans. Reliab. 28 (3) (1979) 206–211.

[12] Gokhale, S.S., Philip, T., Marinos, P.N., & Trivedi, K.S. (1996). *Unification of finite failure non-homogeneous Poisson process models through test coverage.* Paper presented at the Proceedings of ISSRE'96: 7th International Symposium on Software Reliability Engineering.

[13] S.S. Gokhale, K.S. Trivedi, A time/structure based software reliability model, Annals of Software Engineering 8 (1) (1999) 85–121.

[14] C.-.Y. Huang, S.-.Y. Kuo, M.R. Lyu, An assessment of testing-effort dependent software reliability growth models, IEEE Trans. Reliab. 56 (2) (2007) 198–211.

[15] S. INOUE, K. FUKUMA, S. YAMADA, Two-dimensional change-point modeling for software reliability assessment, Int. J. Reliab. Qual. Saf. Eng. 17 (06) (2010) 531–542.

[16] P. Kapur, A.G. Aggarwal, S. Anand, A new insight into software reliability growth modeling, International Journal of Performability Engineering 5 (3) (2009) 267.

[17] P. Kapur, S. Anand, K. Yadav, J. Singh, A unified scheme for developing software reliability growth models using stochastic differential equations, International Journal of Operational Research 15 (1) (2012) 48–63.

[18] P. Kapur, S. Anand, V. Yadavalli, F. Beichelt, A generalised software growth model using stochastic differential equation, Communication in Dependability and Quality Management Belgrade, Serbia (2007) 34.

[19] P. Kapur, S. Anand, S. Yamada, V.S. Yadavalli, Stochastic differential equation-based flexible software reliability growth model, Math. Probl. Eng. 2009 (2009).

[20] P. Kapur, D. Gupta, A. Gupta, P. Jha, Effect of introduction of fault and imperfect debugging on release time, Ratio Mathematica 18 (1) (2008) 62–90.

[21] P. Kapur, H. Pham, A.G. Aggarwal, G. Kaur, Two dimensional multi-release software reliability modeling and optimal release planning, IEEE Trans. Reliab. 61 (3) (2012) 758–768.

[22] P. Kapur, H. Pham, S. Anand, K. Yadav, A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation, IEEE Trans. Reliab. 60 (1) (2011) 331–340.

[23] Kapur, P., Pham, H., Gupta, A., & Jha, P. (2011). Software reliability assessment with OR applications.

[24] P. Kapur, N. Sachdeva, J.N. Singh, Optimal cost: a criterion to release multiple versions of software, International Journal of System Assurance Engineering and Management 5 (2) (2014) 174–180.

[25] Kapur, P., Tandon, A., & Kaur, G. (2010). *Multi up-gradation software reliability model.* Paper presented at the 2010 2nd International Conference on Reliability, Safety and Hazard-Risk-Based Technologies and Physics-of-Failure Methods (ICRESH).

[26] Y.K. Malaiya, M.N. Li, J.M. Bieman, R. Karcich, Software reliability growth with test coverage, IEEE Trans. Reliab. 51 (4) (2002) 420–426.

[27] G. Mishra, P. Kapur, A.G. Aggarwal, A generalized multi-upgradation SRGM considering uncertainty of random field operating environments, International Journal of System Assurance Engineering and Management (2023) 1–9.

[28] Ohba, M., & Chou, X.-.M. (1989). *Does imperfect debugging affect software reliability growth?* Paper presented at the Proceedings of the 11th international conference on Software engineering.

[29] B. Øksendal, Stochastic differential equations, in: Stochastic Differential Equations, Springer, 2003, pp. 65–84.

[30] B. Pachauri, J. Dhar, A. Kumar, Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth, Appl Math Model 39 (5–6) (2015) 1463–1469.

[31] J.-.Y. Park, G. Lee, J.H. Park, A class of coverage growth functions and its practical application, J Korean Stat Soc 37 (3) (2008) 241–247.

[32] H. Pham, L. Nordmann, Z. Zhang, A general imperfect-software-debugging model with S-shaped fault-detection rate, IEEE Trans. Reliab. 48 (2) (1999) 169–175.

[33] H. Pham, X. Zhang, An NHPP software reliability model and its comparison, Int. J. Reliab. Qual. Saf. Eng. 4 (03) (1997) 269–282.

[34] H. Pham, X. Zhang, NHPP software reliability and cost models with testing coverage, Eur J Oper Res 145 (2) (2003) 443–454.

[35] V. Pradhan, A. Kumar, J. Dhar, Modeling Multi-Release Open Source Software Reliability Growth Process with Generalized Modified Weibull Distribution, Evolving Software Processes: Trends and Future Directions (2022) 123–133.

[36] Y. Shigeru, N. Akio, A stochastic differential equation model for software reliability assessment and its goodness-of-fit, International Journal of Reliability and Applications 4 (1) (2003) 1–12.

[37] Y. Tamura, S. Yamada, A flexible stochastic differential equation model in distributed development environment, Eur J Oper Res 168 (1) (2006) 143–152.

[38] A. Tandon, A.G. Aggarwal, Testing coverage based reliability modeling for multi-release open-source software incorporating fault reduction factor, Life Cycle Reliability and Safety Engineering 9 (4) (2020) 425–435.

[39] V. Verma, S. Anand, A.G. Aggarwal, Reliability Assessment of Multi-release Software System Under Imperfect Fault Removal Phenomenon, in: Decision Analytics Applications in Industry, Springer, 2020, pp. 367–380.

[40] Vouk, M. (1992). *Using reliability models during testing with non-operational profiles.* Paper presented at the Proceedings of the 2nd Bellcore/Purdue workshop on issues in Software Reliability Estimation.

[41] A. Wood, Predicting software reliability, Computer (Long Beach Calif) 29 (11) (1996) 69–77.

[42] M. Xie, Q. Hu, Y. Wu, S.H. Ng, A study of the modeling and analysis of software fault-detection and fault-correction processes, Qual. Reliab. Eng. Int. 23 (4) (2007) 459–470.

[43] S. Yamada, M. Ohba, S. Osaki, S-shaped software reliability growth models and their applications, IEEE Trans. Reliab. 33 (4) (1984) 289–292.

[44] S. Yamada, S. Osaki, Software reliability growth modeling: models and applications, IEEE Trans. Software Eng. (12) (1985) 1431–1437.

[45] S. Yamada, K. Tokuno, S. Osaki, Imperfect debugging models with fault introduction rate for software reliability assessment, Int. J. Syst. Sci. 23 (12) (1992) 2241–2252.