Annals of Operations Research (2022) 312:65–85
https://doi.org/10.1007/s10479-021-04258-y

S.I. : STATISTICAL RELIABILITY MODELING AND OPTIMIZATION

Reliability and optimal release time analysis for multi up-gradation software with imperfect debugging and varied testing coverage under the effect of random field environments

Subhashis Chatterjee[1] · Deepjyoti Saha[1] · Akhilesh Sharma[2] · Yogesh Verma[2]

Accepted: 30 August 2021 / Published online: 30 October 2021

Abstract
Due to change requests for up-gradation of adding new features, software organizations always develop new versions of the software by adding new features and improving the existing software. Various software reliability growth models have been proposed considering realistic issue which affects the reliability growth of software. Testing coverage is a crucial realistic issue that influences the fault detection and correction process. The difficulty level for removing different faults is different, same kind of testing coverage function can't capture the fault detection process for all types of faults. Also, there exist random effects in the field environment due to the change between the testing environment and the operational environment. This randomness also affects the reliability growth of software. In this paper, a software reliability growth model has been proposed considering imperfect debugging, faults removal proportionality, two types of testing coverage function in the presence of random effect of the testing environment. Here different categories of faults have been considered. Though reliability is an important issue for software professionals, they are worried about the optimal release of software at an optimal cost. Considering the testing cost and debugging cost random, a cost model has been proposed for release time analysis.

Keywords Software reliability · Non-homogeneous poisson process (NHPP) · Random field environment (RFE) · Testing coverage function · Imperfect debugging · Fault removal proportionality · Optimal release time · Mean value function (MVF)

✉ Deepjyoti Saha
sahadeepjyoti01@gmail.com

[1] Department of Mathematics & Computing, IIT(ISM) Dhanbad, Dhanbad, Jharkhand, India

[2] GSQAD/SRG/SAC-ISRO, Ahmedabad, India

1 Introduction

People are acquainted with a well-behaved software system where works are carried out in a fast, efficient manner by the software. The traditional method of doing a job is time taking; the software system saves time and makes life more comfortable with its best use. Software system has gained the most prominent place in the scientific, technological, and R&D field. Due to the growing importance of software in safety–critical areas like: banking sector, medical science, communication, defence, Air traffic control system, nuclear power plant, etc. (Pham, 2007), software professionals are always in search for developing more efficient and reliable software. Due to multiple functionalities and complex development process fault detection and correction process has become very difficult as well as the failure of software causes a huge burden on the economy. Therefore, people are more concerned about the reliability issues of software. As a result, software developers carry out rigorous testing with the objective of producing a highly reliable software system. Software reliability growth model (SRGM) plays a very important role in measuring and analysing the reliability of software system (Yamada & Osaki, 1985). Since the last five decades, many SRGMs have been proposed to predict the remaining faults, reliability growth of software, failure rate, etc. SRGMs are also used for optimal release time analysis. SRGMs based on NHPP are very popular (Kapur et al., 2011; Pachauri et al., 2015; Pham, 2001; Pham & Zhang, 1997; Yamada et al., 1984) among the software professionals and software reliability research community. Every parametric SRGMs are formulated based on various assumptions such as: imperfect debugging, the effect of coverage factor, the effect of change point, effect of random field environment, dependency of faults, the effect of learning, etc., and these models are used to study the effects of these factors on fault detection and correction process during testing (Pham, 1995). Debugging can't be perfect due to partial removal of the cause of existing faults and the generation of new faults during fault detection and correction process. Also, change in development and testing conditions as well as variations in the performance of developers, faults can't be removed perfectly. Hence, many SRGMs have been proposed based on imperfect debugging and fault removal efficiency (Chatterjee et al., 2012; Pham et al., 1999; Yamada et al., 1993; Zhang et al., 2003).

Parametric SRGMs are mainly developed using failure data gathered during testing of software. In house testing of software is carried out in a regulated environment. Therefore, the factors which are used for the development of parametric SRGMs are also measured in the same controlled environment. Practically there is a difference between the testing environment and the field environment (Chang et al., 2014; Teng & Pham, 2006). After release, standalone software may be used in varied locations for divergent applications and purposes. Thus, operating environment for standalone software is separate from regulated testing environment, and the failure behaviour of the software will be different in the field environment due to the presence of randomness. Ultimately the randomness in the field environment affects the cumulative failure, different metrics used to measure reliability, and different factors that are used to develop SRGMs.

Testing coverage (Chang et al., 2014; Chatterjee & Shukla, 2016; Chatterjee & Singh, 2014) is very important from the developer's point of view and also from the customer's point of view. To developers, testing coverage is an instrument for monitoring the improvement in fault prediction and introduction process, whereas to the customer, testing coverage provides the trust about the software product they are going to purchase and use. Testing coverage helps the developers to get an idea about additional efforts required to improve the reliability of software. Many researchers use different testing coverage functions to develop SRGMs.

The only way to deliver improved and reliable software to the customers is by adding new features, correcting residuals bugs, and upgrading the existing version. Hence, up-gradation of software takes place as when required depending on the demand from the customer side or from in house development team. Software up-gradation may generate many faults (Kapur et al., 2014). There are few SRGMs available in the literature related to multi-release and multi up-gradation of software system (Yang et al., 2016; Kapur et al., 2010a, b) based on different realistic issues. None of these SRGMs has taken care of two very important issues: the effect of random environment and testing coverage factor. The random field environment is a very important issue as it influences fault removal efficiency, faulty introduction rate, failure rate, change point, testing coverage, etc. Considering this factor in this paper, a SRGM has been proposed based on two types of testing coverage functions, presence of randomness in field environment of testing, imperfect debugging, and percentage of faults removal efficiency to estimate the reliability of upgraded software system. Also, in the proposed work presence of two types of faults has been considered: soft faults that are easy to detect and remove from software, and hard faults which are not easy to remove and detect from software. This implies that the testing coverage should be different for two different types of faults to quantify the effect of these two types of faults. This fact has also been incorporated in the proposed SRGM. The effect of random field environments with different fault types along with different test coverage factors on the reliability growth of multi up-gradation software has not been studied earlier.

Though reliability is an important issue for software companies, there is also another important and challenging issue to the software developers and organizations which is: when to release the software in the market with optimal cost. In order to achieve higher reliability, software is normally tested for a longer time of period before releasing. The testing phase is the costliest phase as half of the amount of software development resources is consumed in testing phase. Many optimal release time models have been found for single release problem in literature (Ho & Ruhe, 2013; Li et al., 2014; Sailu & Ruhe, 2005; Zhu & Pham, 2018a, b; Li et al., 2011). Software companies always try to estimate the optimal release time such that the cost of the product will be optimal (Chatterjee et al., 2018; Kapur et al., 2014; Pham et al., 2003). Hence, a new optimal release time model has also been proposed considering the random debugging cost. In most of the existing literature about optimal release time analysis, the debugging cost is considered as constant. Practically due to imperfect debugging, variation in fault removal efficiency, the effect of random field environment in debugging process, the debugging cost cannot be constant. Rather, it will change with time, i.e., debugging costs will be random. Hence, in this article, the debugging cost is considered as a function of time to incorporate the randomness.

The paper is organized as follows: Sect. 2 presents the proposed model, Sect. 3 describes parameter estimation, model validation and the comparative study with the existing model, Sect. 4 gives the details of optimal release time, and Sect. 5 concludes the paper.

2 Notation

The following notations are used for model formulation:

a_k: Total initial faults present in the k^{th} release of software.
s_{1k}: Proportion of soft faults in the k^{th} release of software.
s_{2k}: Proportion of hard faults in the k^{th} release of software where $s_{2k} = \left(1 - s_{1k}\right)$.
p_{1k}: Probability that a soft type faults will be successfully removed from the k^{th} release of software.
p_{2k}: Probability that a hard type faults will be successfully removed from the *kth* release of software.
q_{1k}: Error introduction rate of soft type faults from the *kth* release of software.
q_{2k}: Error introduction rate of hard type faults from the *kth* release of software.
λ_k: Scale parameter of Gamma distribution for *kth* release.
η: Random environment effect.
$f(\eta)$: Probability density function of η.
b_{1k}, ϕ_{1k}: Parameters of Weibull distribution of k^{th} release of software.
b_{2k}: Parameter of delayed S-shaped distribution of k^{th} release of software.
m_k: Shape parameter of Gamma distribution of k^{th} release of software.

3 Software reliability modelling

This section depicts a glimpse about the details of software reliability terminologies used for the proposed model.

3.1 Assumptions

The following assumptions are made for the proposed model:

1. Generation of fault follows an NHPP.
2. Fault detection rate is proportional to the remaining faults in software.
3. Removal of faults take place immediately with a debugging effort having probability p.
4. Debugging process is imperfect.
5. Presence of testing coverage function.
6. Effect of random environment on testing coverage function.
7. Existence of two types of faults: soft fault and hard fault.

3.2 SRGM formulation with fault removal efficiency and imperfect debugging

When failure occurs in software, the review board assigns developers to look into the different aspects of the failure and causes behind the failures. As a result, the testing process gets initiated. Most of the time, bugs can't be eliminated from the software perfectly by review, inspections, and tests, because of poor fault removal efficiency (Chatterjee et al., 2012; Pham et al., 1999; Yamada et al., 1993). Fault removal efficiency (Zhang et al.,

2003) is the proportion of bugs removed by developers. Imperfect debugging (Yamada, 2014) also plays a crucial role as new faults occur in software during the development period. Based on the aforementioned discussions, the MVF of the proposed model will be as follows:

$$\begin{aligned} m'(t) &= h(t)(a(t) - p.m(t)) \\ a'(t) &= q.m'(t) \\ &\text{with a(0) = a and m(0) = 0} \end{aligned} \tag{1}$$

3.3 SRGM with testing coverage and effect of random environment

During testing time, the software failure rate also depends on the environmental effects (Chang et al., 2014; Teng & Pham, 2006; Zhu & Pham, 2018a, 2018b(b)). The impact of environment on software fault generation can't be constant; practically, the environment is uncertain. To capture the uncertainty of the environment, a time-independent non-negative random variable η has been considered here. The fault detection in an uncertain field environment is captured by multiplying the constant η with unit failure detection rate $h(t)$, Therefore, the proposed SRGM in the presence of environmental effect will be as follows:

$$\begin{aligned} m'(t) &= \eta h(t)(a(t) - p.m(t)) \\ a'(t) &= q.m'(t) \\ &\text{with a(0) = a and m(0) = 0} \end{aligned} \tag{2}$$

It is known that testing coverage plays a very important role in the fault detection and correction process (Chang et al., 2014; Chatterjee & Shukla, 2016; Chatterjee & Singh, 2014)). Also, the importance of testing coverage from the developer's point of view as well as from the user's point of view has been presented in the introduction of this paper. Generally testing coverage factor is denoted by $c(t)$. The derivative of testing coverage factor $c'(t)$ gives coverage rate and $(1 - c(t))$ denotes the proportion of codes which has not been tested yet by test cases during the time t. Hence, the unit failure detection rate function in terms of testing coverage function will be $h(t) = \frac{c'(t)}{(1-c(t))}$.

Therefore, the proposed model will be as follows:

$$\begin{aligned} m'(t) &= \eta \frac{c'(t)}{(1 - c(t))}(a(t) - p.m(t)) \\ a'(t) &= q.m'(t) \\ &\text{with a(0) = a and m(0) = 0} \end{aligned} \tag{3}$$

The closed form of MVF derived from Eq. (2) with initial condition will be

$$\begin{aligned} m_\eta(t) &= a\eta \int_0^t b(u).exp(-\int_0^u \eta(p - q)b(\tau)d\tau)du \\ &\text{where } b(\tau) = \frac{c'(\tau)}{1 - c(\tau)} \end{aligned} \tag{4}$$

3.4 Gamma RFE reliability model

Previously two types of non-negative probability density function have been proposed for η, one is Gamma distribution, and the other is Beta distribution (Pahm, 2007). The random field environment factor η can follow any kind of positive distribution, but Gamma distribution can take care of all conditions for fault detection in testing as well as the operational environment. η less than 1 implies that the condition for fault detection in operational environment is less favorable than testing environment, whereas when $\eta = 1$ the condition for fault detection are same in two different environments and when $\eta > 1$ the condition for detection of faults in operational environment will be more favorable than testing environment. On the other hand, in case of Beta distribution η can take value only between 0 and 1. Hence, Gamma distribution is more suitable for describing random field environment and due to this reason Gamma distribution has been considered here.

Also, Fig. 1 shows the density curve of Gamma probability distribution, and it also seems to be reasonable to describe the software failure phenomenon in uncertain environments.

The Gamma distribution for η is as follows:

$$f_\lambda(\eta) = \frac{\lambda^m \eta^{(m-1)} e^{-\lambda\eta}}{(m-1)!} \text{where parameters } \lambda, \eta, m \geq 0 \tag{5}$$

The Laplace transformation of $f_\lambda(\eta)$ given in (4) is

$$F^*(s) = \left[\frac{\lambda}{\lambda + s}\right]^m \tag{6}$$

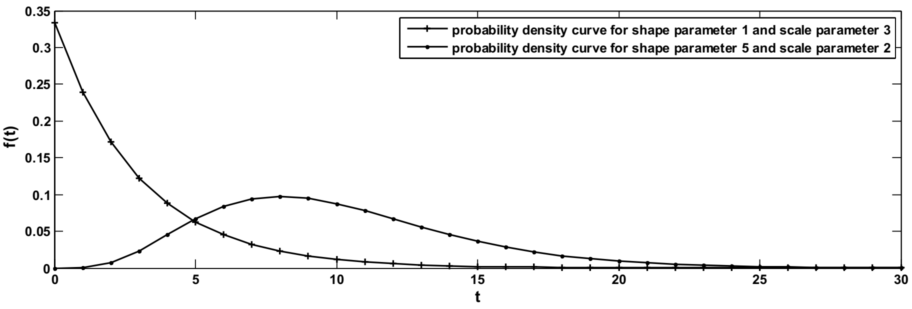

Fig. 1 Gamma distribution function with two different shape parameters

Finally the MVF of the proposed SRGM using Gamma distribution is as follows:

$$\begin{aligned} m(t) &= \int_0^{\infty} m_{\eta}(t) f(\eta) d\eta \\ &= \int_0^{\infty} a\left(\eta \int_0^{t} b(u).exp\left(-\int_0^{u} \eta(p-q) b(\tau) d\tau\right) du f(\eta)\right) d\eta \\ &= \int_0^{t} b(u)\left(\int_0^{\infty} a.exp(-\int_0^{u} \eta(p-q) b(\tau) d\tau) \eta f(\eta) d\eta\right) du \end{aligned}$$

Using the Laplace transformation, the compact form of the solution of MVF $m(t)$ is as follows:

$$m(t) = \frac{a}{(p-q)}\left(1-\left(\frac{\lambda}{\lambda+(p-q)\int_0^{t} b(u) du}\right)^{m}\right) \tag{7}$$

$$\text{where } b(u) = \frac{c'(u)}{1-c(u)}$$

3.5 SRGM based on testing coverage function

Due to changes in codes, environmental effects, imperfect debugging, etc., new faults occur in software. Faults are not the same kind; some faults are very easy to detect and remove; on the other hand, some faults can't be easily detected and removed. To remove the causes of the second category of faults extra effort is required. The first categories of faults are classified as soft faults, and second category of faults are classified as hard faults. Grambaaki et al. (Grambaki et al., 2011) has quantified the soft and hard fault using a logistic function in the mean value function $m(t)$. Testing coverage function is more appropriate for quantifying the effect of soft and hard fault. As detection and removal of soft fault is easy, hence the time taken to detect and removal of soft fault is negligible. This phenomenon can be better represented using exponential testing coverage function. As exponential function is a special case of Weibull function, hence Weibull function has been used here to represent the testing coverage function for soft fault, and this gives a generalized effect. On the other hand, hard faults are difficult to detect and remove. Hence, more time and effort will be required for detection and removal of hard faults. Also, the knowledge and skill of developers play an important role in detection and removal of hard faults. The knowledge and skill of developer's increases with learning. Therefore, with the passage of time, knowledge and skill of developers increases from low to high with. This phenomenon is better captured by S-shaped testing coverage function. Hence, S-shaped coverage function has been used here to represent the testing coverage function of hard faults. It is observed that the Weibull curve and delayed S-shaped curve captures the fault detection phenomenon better for soft and hard faults respectively. Based on this argument, two testing coverage function is considered in the proposed SRGM. Out of these two functions, one is Weibull type testing coverage function for soft faults, and the other is delayed S-shaped testing coverage function for hard faults. Therefore, the testing coverage functions are as follows:

$c_1(t) = \left(1 - e^{-bt^{\phi}}\right)$ and $c_2(t) = 1 - ((1 + bt).exp(-bt))$

The MVF for Weibull type testing coverage function will be

$$m(t) = \frac{a}{(p-q)}\left(1 - \left(\frac{\lambda}{\lambda + (p-q)bt^{\phi}}\right)^{m}\right) \tag{8}$$

where the constant b and ϕ reflects the quality of testing,

The MVF with delayed S-shaped testing coverage is as follows:

$$m(t) = \frac{a}{(p-q)}.\left(1 - \left(\frac{\lambda}{\lambda - (p-q).ln\left((1+bt)e^{-bt}\right)}\right)^{m}\right) \tag{9}$$

The combined MVF in presence of two types of testing coverage is as follows:

$$\begin{aligned} m(t) = {} & \frac{as_1}{(p_1-q_1)}.\left(1 - \left(\frac{\lambda}{\lambda + (p_1-q_1)b_1t^{\phi_1}}\right)^{m}\right) \\ & + \frac{as_2}{(p_2-q_2)}.\left(1 - \left(\frac{\lambda}{\lambda - (p_2-q_2).ln\left((1+b_2t)e^{-b_2t}\right)}\right)^{m}\right) \end{aligned} \tag{10}$$

The details of the proposed model for different releases are discussed in the following subsection,

3.6 Proposed SRGM for different releases

The mean value function can be derived for different releases using Eq. (10).

Hence, MVF for 1st release of the software will be as follows:

$$\begin{aligned} m_1(t) = {} & \frac{a_1s_{11}}{(p_{11}-q_{11})}.\left(1 - \left(\frac{\lambda_1}{\lambda_1 + (p_{11}-q_{11})b_{11}t^{\phi_1}}\right)^{m_1}\right) \\ & + \frac{a_1s_{21}}{(p_{21}-q_{21})}.\left(1 - \left(\frac{\lambda_1}{\lambda_1 - (p_{21}-q_{21}).ln\left((1+b_{21}t)e^{-b_{21}t}\right)}\right)^{m_1}\right) \text{ for } 0 \le t \le t_1 \end{aligned} \tag{11}$$

Hence, MVF can be written for 1st release is as follows: $m_1(t) = a_1 \times F_1(t)$ for $0 \le t \le t_1$ where.

$F_1(t) = \frac{s_{11}}{(p_{11}-q_{11})}.\left(1 - \left(\frac{\lambda_1}{\lambda_1+(p_{11}-q_{11})b_{11}t^{\phi_1}}\right)^{m_1}\right) + \frac{s_{21}}{(p_{21}-q_{21})}.\left(1 - \left(\frac{\lambda_1}{\lambda_1-(p_{21}-q_{21}).ln\left((1+b_{21}t)e^{-b_{21}t}\right)}\right)^{m_1}\right)$

Now, MVF for other releases are as follows: $m_k(t) = \{a_k + a_{k-1}.(1 - F_{k-1}(t_{k-1}))\} \times F_k(t)$ for $t_{k-1} \le t \le t_k$ and where

$$\begin{aligned} F_k(t) = {} & \frac{s_{1k}}{(p_{1k}-q_{1k})}.\left(1 - \left(\frac{\lambda_k}{\lambda_k + (p_{1k}-q_{1k})b_{1k}(t-t_{(k-1)})^{\phi_k}}\right)^{m_k}\right) \\ & + \frac{s_{2k}}{(p_{2k}-q_{2k})}.\left(1 - \left(\frac{\lambda_k}{\lambda_k - (p_{2k}-q_{2k}).ln\left((1+b_{2k}(t-t_{(k-1)}))e^{-b_{2k}(t-t_{(k-1)})}\right)}\right)^{m_k}\right) \text{ for } k = 2,3,4,\ldots \end{aligned} \tag{12}$$

4 Parameter estimation, model validation & performance analysis

This section presents the parameter estimation and validation of the proposed model using some real software failure data for different releases. Also, the performance analysis of the proposed model has been carried out comparing with some existing SRGMs (Garmabaki et al., 2011; Kapur et al., 2010a, b).

4.1 Parameter estimation

The parameters of the proposed model have been estimated using non- linear least square estimation method with the help of SPSS.

The failure data which has been collected during testing phase of software is in the form of $\left(t_i, y_i\right)$ where t_i represents the time and y_i represents actual cumulative data of i^{th} observation. Here, residuals $r_{i,}$ for $i = 1, 2, 3, \ldots$ can be described by $r_i = y_i - f(t_i, x)_,$ where x is a vector of m unknown parameters. Let S is defined as $S = \sum_i r_i^2$. The unknown parameters can be found by equating the gradient of S with 0, *i.e*, $\frac{\delta S}{\partial x_i} = 0$ for i = 1,2,... The estimated parameters are obtained solving $\frac{\delta S}{\partial x_i} = 0$ using SPSS.

4.2 Model validation and performance analysis

Tandem computer failure data (Wood, 1996) and online bug tracking system Mozilla fire-fox failure data sets (Yang et al., 2016) are used for different releases to estimate unknown parameters of proposed model (Pham & Zhang, 2003).

4.2.1 Data Set 1

First data set is the Tandem computer failure data (Wood, 1996). The detected faults are collected for four releases. The estimated values of the parameters for the proposed model using this data set have been tabulated in Table 1.

Table 1 Estimated parameters of the proposed model

Parameters	Release 1 (k = 1)	Release 2 (k = 2)	Release 3 (k = 3)	Release 4 (k = 4)
a_k	104.396	125.504	61.6	46.572
s_{1k}	0.140	0.134	0.444	0.957
p_{1k}	0.944	1	1	0.972
q_{1k}	0.122	0	0.001	0.004
λ_k	0.016	16,576.88	10.076	92.687
b_{1k}	0.014	0.044	0.001	0.512
φ_k	4.091	0.010	4.586	1.75
p_{2k}	1	1	0.999	0.603
q_{2k}	0.001	0	0	0.602
b_{2k}	0.011	0.033	0.999	0.001
m_k	3.482	435,570.431	2.639	3.775

Table 2 Estimated parameters of the proposed model

Parameters	Release 1 (k = 1)	Release 2 (k = 2)	Release 3 (k = 3)
a_k	65.58	51.03	33.27
s_{1k}	0.393	0.969	0.001
p_{1k}	0.999	1	0.9
q_{1k}	0.287	0.003	0.899
λ_k	8.498	0.001	0.387
b_{1k}	0.0013	0.001	0.001
φ_k	1.046	7.762	2.915
p_{2k}	1	6.7	1
q_{2k}	001	6.667	0
b_{2k}	0.002	0.164	0.103
m_k	1799.649	0.022	0.626

4.2.2 Data Set 2

This data set is collected from Mozilla Firefox failure data (Yang et al., 2016). The detected faults are collected for three releases. The estimated values of parameters for the proposed model using this data set have been tabulated in Table 2.

The plots of cumulative number of faults of different releases for the Tandem computer failure data sets are given in Figs. 2, 3, 4, and 5. And Figs. 6, 7, and 8 have given the cumulative faults for different releases of the Mozilla fire-fox data sets. It is observed

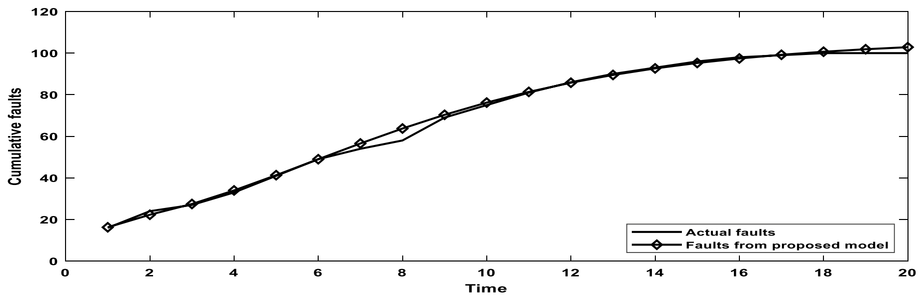

Fig. 2 Predicted cumulative faults via proposed models for Release 1

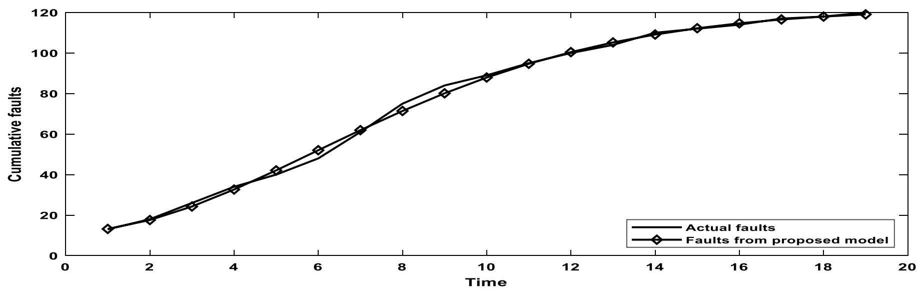

Fig. 3 Predicted cumulative faults via proposed models for Release 2

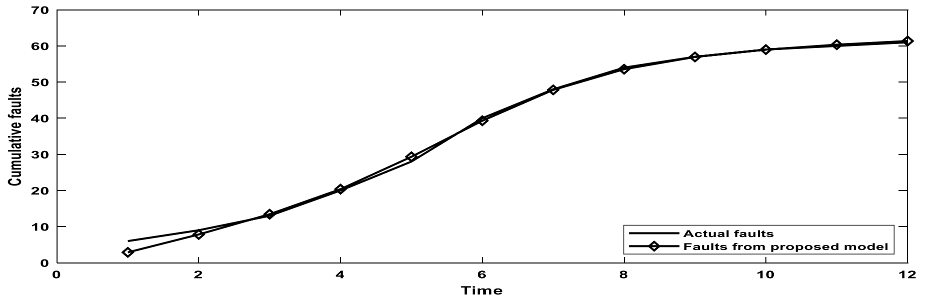

Fig. 4 Predicted cumulative faults via proposed models for Release 3

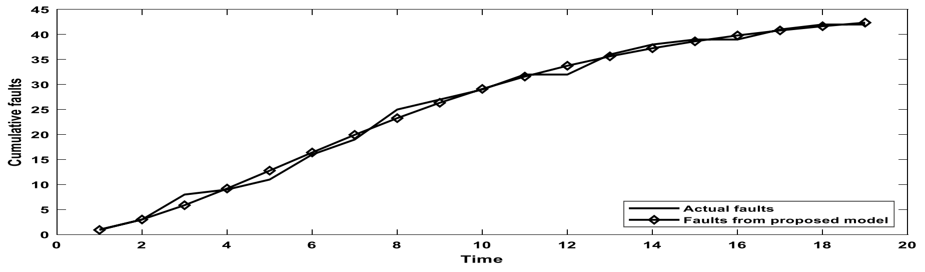

Fig. 5 Predicted cumulative faults via proposed models for Release 4

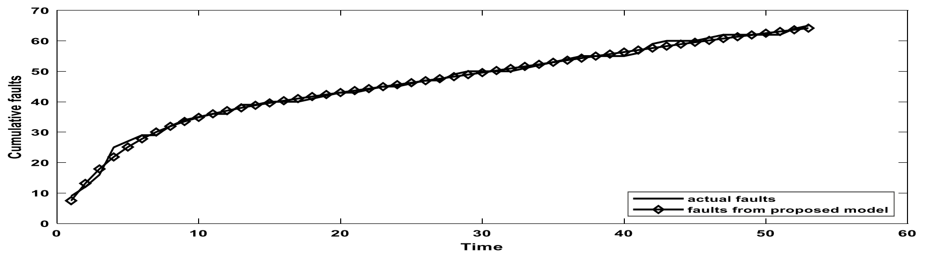

Fig. 6 Predicted cumulative faults via proposed models for Release 1

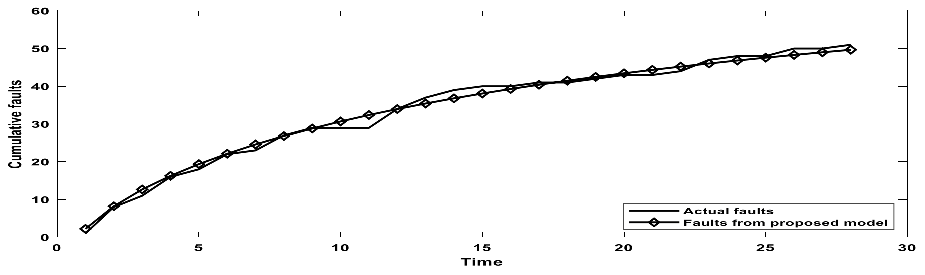

Fig. 7 Predicted cumulative faults via proposed models for Release 2

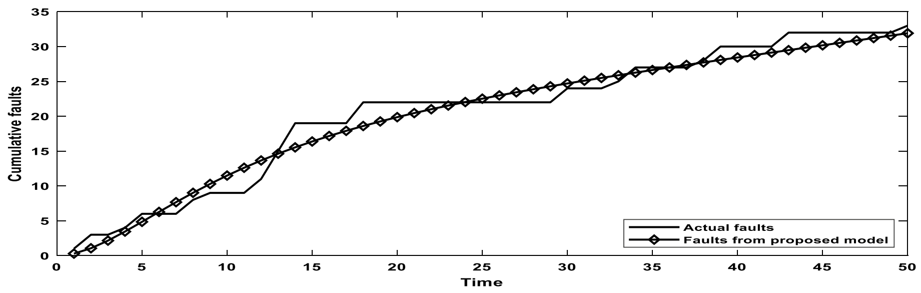

Fig. 8 Predicted cumulative faults via proposed models for Release 3

that the predicted faults are very close to the actual faults exist in the different release. The plots validate the different assumptions made to develop the proposed SRGM.

4.3 Comparison criterion

The proposed model has been compared with existing models for multi release problem (Kapur et al., 2010a, 2010b, December; Garmabaki et al., 2011, December; Kapur et al., 1999; Kapur et al., 2010a, 2010b, December). Different comparison measure like: Sum of Square of Errors (SSE), Mean Square Error (MSE) and Adjusted R^2, Akaike Information Criteria (AIC), Bayesian Information Criteria (BIC), and Mean Absolute Error have been used for model comparison (Kapur et al., 2011; Pham, 2007). The following tables give the values of these criteria for the used data sets. The comparative study for the proposed model with the existing models based on these criteria has been tabulated in Tables 3 and 4.

As shown in Tables 3 and 4, the values of SSE and MSE and AIC and mean absolute error are less for the proposed model than the existing model, and Adjusted R^2 values are high than the existing model. Hence, it can be said from the tabulated values that the proposed SRGM performs better. Also, the comparative study establishes the validation of the assumptions made for the proposed model.

5 Optimal release time modeling

This section provides the details of optimal release time for different releases of software. As development process is a continuous process, software companies can't place software at one time in the market (Li et al., 2011; Naciri et al., 2015; Szoke et al., 2011). Instead, software developers release a new version of software within a shorter span of time. Developers also try to enhance reliability of new versions than the existing ones. If developers stop testing too early, bugs can't be identified and removed properly. As a result, failures occur in the system. On the other hand, if too much testing carried out, it will consume extra effort and cost increases. Due to business competition and commitment to clients, organizations always in search for suitable optimal release time with affordable cost of the software (Chatterjee & Shukla, 2019; Chatterjee et al., 1997, 2012, 2019). Considering this a cost model has been carried out to find optimal releasing time of software with affordable cost for releasing.

During the up-gradation and release of new versions of software, new faults may generate in successive versions as well as some faults will also survive from the previous versions,

Table 3 Comparative study of the proposed model with existing models for Tandem computer data

Models		SSE	MSE	Adjusted R^2	AIC	BIC	Mean absolute Error
Release 1	Proposed model	16.156	0.8078	0.997	26.036	36.989	0.8805
	Garmabaki's faults in severity model	38.836	1.942	0.992	48.569	53.548	3.6789
	Kapur's imperfect debugging model	151.662	7.581	0.989	54.699	59.678	4.1785
	Logistic multi-release model	179.4	8.97	0.989	52.378	57.21	6.6712
	Goel-Okumoto model	469.533	23.483	0.976	53.061	55.052	7.0712
	Delayed S-shaped model	505.128	25.286	0.969	68.611	70.603	12.4121
Release 2	Proposed model	58.348	3.071	0.998	33.563	43.952	1.0017
	Garmabaki's faults in severity model	64.7938	3.4102	0.997	37.309	46.864	2.8915
	Kapur's imperfect debugging model	144.685	7.615	0.995	48.572	53.294	4.6987
	Logistic multi-release model	160.55	8.45	0.994	61.342	57.143	6.1341
	Goel-Okumoto model	445.689	22.28445	0.971	86.364	88.253	7.7266
	Delayed S-shaped model	249.777	12.489	0.978	53.788	55.677	2.5578
Release 3	Proposed model	14.339	1.024	0.995	27.617	32.951	1.0661
	Garmabaki's faults in severity model	7.702	0.642	0.997	4.6786	7.103	0.8967
	Kapur's imperfect debugging model	27.708	2.309	0.995	28.039	32.987	1.9723
	Logistic multi-release model	85.68	7.14	0.983	28.782	34.164	2.2751
	Goel-Okumoto model	243.172	20.264	0.982	49.713	50.683	6.0354
	Delayed S-shaped model	12.159	1.013	0.990	30.456	38.880	2.5312
Release 4	Proposed model	15.7508	0.8289894	0.996	18.437	28.825	0.6987
	Garmabaki's faults in severity model	16.140	0.850	0.995	20.456	32.671	0.8927
	Kapur's imperfect debugging model	16.065	0.846	0.995	19.454	31.678	0.8909
	Logistic multi-release model	31.35	1.65	0.991	27.175	37,169	1.978
	Goel-Okumoto model	86.703	4.335	0.952	47.893	49.782	2.6557
	Delayed S-shaped model	18.618	0.930	0.981	41.106	42.995	2.3279

Table 4 Comparative study of the proposed model with existing models for Mozilla data set

Models		SSE	MSE	Adjusted R^2	AIC	BIC	Mean absolute error
Release 1	Proposed model	43.177	0.81466	0.998	57.796	79.469	1.004548
	Kapur's faults in severity model	48.273	0.911	0.989	48.273	105.98	1.38232
	Garmabaki's imperfect debugging model	822.035	15.510	0.987	822.035	170.11	3.660802
	Logistic multi-release model	831.804	16.63	0.912	152.84	164.214	3.4134
	Goel-Okumoto model	820.962	15.490	0.908	149.26	153.20	3.46076
	Delayed S-shaped model	2073.222	39.117	0.778	198.33	202.27	5.4045
Release 2	Proposed model	37.136	1.3263	0.995	27.99	12.65	1.056977
	Garmabaki's faults in severity model	46.782	1.671	0.988	31.78	32.44	1.06943
	Kapur's imperfect debugging model	50.657	1.80917	0.986	53.49	23.50	1.3677
	Logistic multi-release model	51.142	1.8265	0.990	54.009	24.216	1.472
	Goel-Okumoto model	51.057	1.823	0.986	31.41	28.50	1.07737
	Delayed S-shaped model	187.451	6.694	0.964	72.13	74.79	2.721342
Release 3	Proposed model	121.095	2.4219	0.972	74.66	93.69	1.36607
	Garmabaki's faults in severity model	143.721	2.874	0.964	85.80	95.36	1.6979
	Kapur's imperfect debugging model	144.589	2.894	0.962	86.04	95.60	1.7838
	Logistic multi-release model	159.225	3.18	0.966	87.298	97.126	1.4671
	Goel-Okumoto model	149.192	2.983	0.952	91.36	34.48	1.3989
	Delayed S-shaped model	213.544	4.271	0.955	97.86	99.1	1.8054

which will be carried out in the successive version of that particular release (Kapur et al., 2014; Pham, 2007). Hence, debugging cost for faults migrated from the previous version has been considered in the cost model. The notation, which is used to formulate the cost model is as follows:

$c_{0j}(t)$	cost of testing in testing period in *jth* release.
$c_{1j}(t)$	Debugging cost of faults in testing period of *jth* release.
$c_{2j}(t)$	Debugging cost of faults in operational phase of *jth* release.
$c_j(t)$	Cost for debugging of faults from the previous version.
c_s	Set up cost for software release.

Hence, the cost model for different releases will be as follows

$$\begin{aligned} C_{Total} &= C_{set\,up\,cost} + C_{per\,unit\,testing\,cost} + C_{debuggingcost\,in\,testing\,phase} \\ &\quad + C_{cost\,for\,operational\,period} + C_{debugging\,cost\,for\,remaining\,faults\,from\,previos\,version} \end{aligned} \tag{13}$$

Here, $c_{0j}(t) = c_{0j}$ is taken as a constant since the testing is performed in “in-housing” condition. As mentioned earlier, due to the effect of different factors, the debugging costs in each release are time-dependent. With the passage of time, more testing is required which results in increasing manpower requirements. Hence, debugging cost increases.

Therefore, the cost for debugging in testing period is taken as follows: $c_{1j}(t) = c_{1j} + r_{1j}(t - t_{j-1})$ for $j \geq 1$ and t_0 is the starting time for first release. Here $\mathrm{r_{1j}}$-unit charges will be increased for per unit debugging due to increase of unit time in testing period. Also, the debugging cost from previous remaining faults increases is as debugging cost increases for removing faults in testing period of the present release. The debugging cost in the operational phase will again increase due to the detection and removal of faults during operational period. Also, more skilled personnel are required for identification and removal of faults in this phase. Hence, the cost model for operational phase will be as follows:$c_{2j}(t) = c_{2j} + r_{2j}.(t_{lc} - t)$. Here $\mathrm{r_{2j}}$-unit charges will be increased for per unit debugging due to increase of unit time in operational phase. Here t_{LC} means total life cycle of software.

Cost model for 1st release As there is no previous version before this version, there is no fault to debug in this present version. Hence, there is no debugging cost required for testing previous version. The cost model will be as follows:

$$\begin{aligned} &C(t) = c_s + c_{01}(t)t + c_{11}(t)a_1F_1(t) + c_{12}(t)\big(1 - F_1(t)\big) \\ &\text{for } 0 \leq t \leq t_1 \text{ where } m_1(t) = a_1.F_1(t) \text{ and } m_1(t) \text{ is the mean value} \\ &\text{function of cumulative faults of 1st release} \end{aligned} \tag{14}$$

Cost model for 2nd release New faults will occur in the software during testing and some faults will survive from the previous version. Hence, extra cost will be incurred. Finally proposed cost model for 2nd release will be

$$\begin{aligned} C(t) = c_s &+ c_{02}(t)t + c_{12}(t).a_2.F_2(t - t_1) + c_1(t).a_1.(1 - F_1(t_1)).F_2(t - t_1) \\ &+ c_{22}(t)[\{a_2 + a_1.((1 - F_1(t_1)).(1 - F_2(t - t_1)))\} \\ &- a_2.F_2(t - t_1)] \text{ where } m_2(t) = a_2.F_2(t) \text{ and } m_1(t) \\ &= a_1.F_1(t) \text{ in the time interval } t_1 \leq t \leq t_2. \end{aligned} \tag{15}$$

Cost model for 3rd version As thoroughly testing is going on before releasing any version of software, there is a less chance of migration of any faults from 1st version. In 3rd version there are some faults surviving from 2nd version. Hence, cost model for 3rd version will be

$$\begin{aligned} C(t) = c_s + c_{03}(t)t + c_{13}(t).a_3.F_3(t-t_2) + c_2(t).a_2.(1-F_2(t_2)).F_3(t-t_2) \\ + c_{23}(t)[\{a_3 + a_2.((1-F_2(t_2)).(1-F_3(t-t_2)))\} \\ - a_3.F_3(t-t_2)] \text{ where } m_3(t) = a_3.F_3(t) \text{ and } m_2(t) \\ = a_2.F_2(t) \text{ in the time interval } t_2 \le t \le t_3. \end{aligned} \tag{16}$$

Cost model for 4th version Due to abovementioned reason the cost model for 4th version is as follows:

$$\begin{aligned} C(t) = c_s + c_{04}(t)t + c_{14}(t).a_4.F_4(t-t_3) + c_3(t).a_3.(1-F_3(t_3)).F_4(t-t_3) \\ + c_{24}(t)[\{a_4 + a_3.((1-F_3(t_3)).(1-F_4(t-t_3)))\} \\ - a_4.F_4(t-t_3)] \text{ Where } m_4(t) = a_4.F_4(t) \text{ and } m_3(t) \\ = a_3.F_3(t) \text{ in the time interval } t \ge t_3. \end{aligned} \tag{17}$$

Let c_s=200 dollars, c_{0i} = 10 dollars, c_{1_i} = 15 dollars, c_{2i} = 20 dollars and c_1, c_2, c_3 are taken 15 dollars for reported faults from 1st, 2nd and 3rd release of the software. Due to absence of software experts, testing tools, and effect of field environment in operational phase, debugging process is very much difficult in operational phase than debugging in testing phase. As a result, cost of debugging in operational phase increases with increasing of time more than in testing phase. As a result, r_{1j} and r_{2j} have been considered as 0.1 and 0,15 here. These values are considered as an example to find the cost and optimal release time of the software. Based on the data sets, the release time for different releases is given in Figs. 9, 10, 11 and 12 respectively for the Tandem failure data set and Figs. 13,14, and 15 have been shown the optimal time of release for Mozilla failure data sets. The optimal release time and optimal cost has been tabulated in Tables 5 and 6 for these two data sets.

The arrows in the plots (Figs. 5, 6, 7, and 8) show the optimal release time and corresponding cost for a particular release for the Tandem software failure data.

The arrows in the plots (Figs. 13, 14, and 15) show the optimal release time and corresponding cost for three different releases for the Mozilla data.

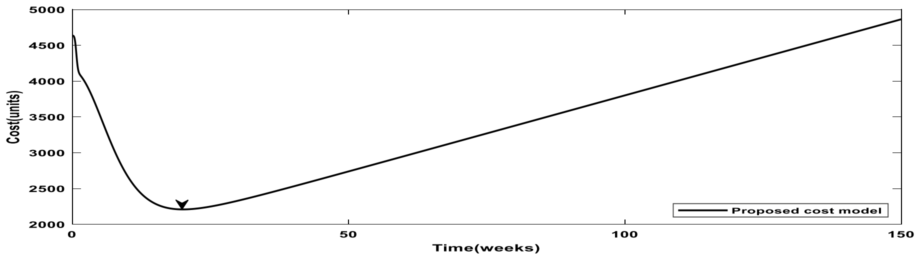

Fig. 9 Predicted optimal release time Release 1 for Tandem data set

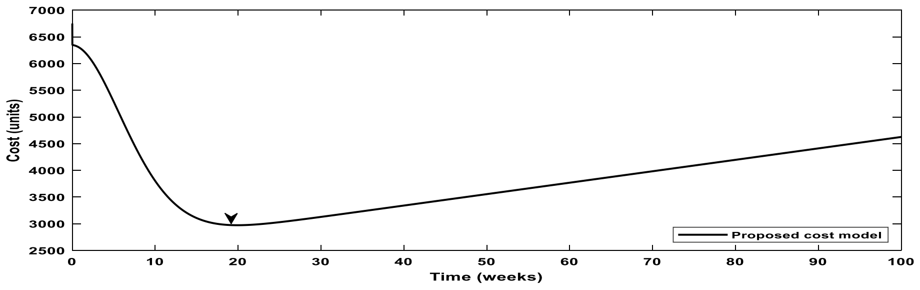

Fig. 10 Predicted optimal release time Release 2 for Tandem data set

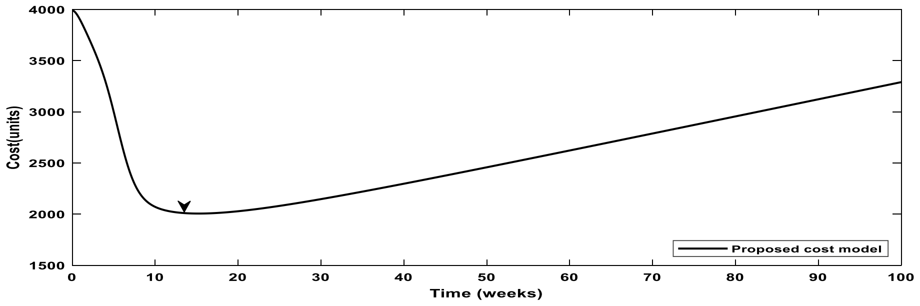

Fig. 11 Predicted optimal release time Release 3 for Tandem data set

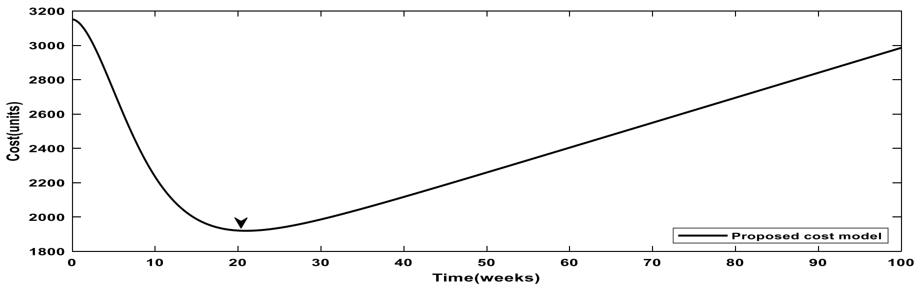

Fig. 12 Predicted optimal release time Release 4 for Tandem data set

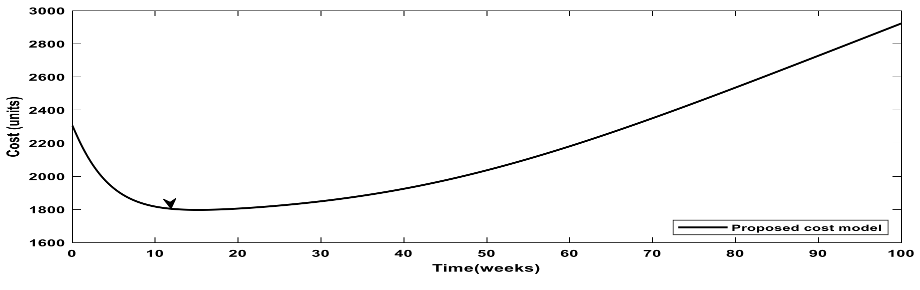

Fig. 13 Predicted optimal release time Release 1 for Mozilla data set

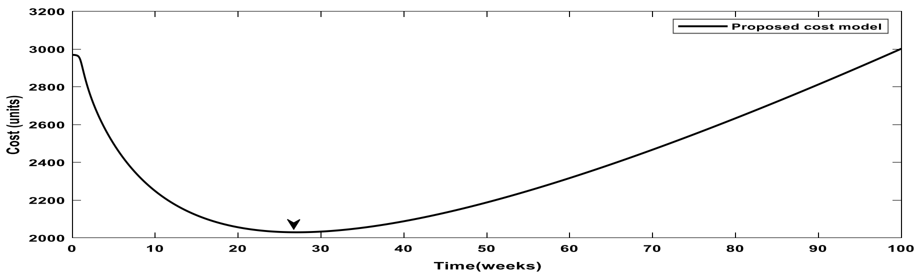

Fig. 14 Predicted optimal release time Release 1 for Mozilla data set

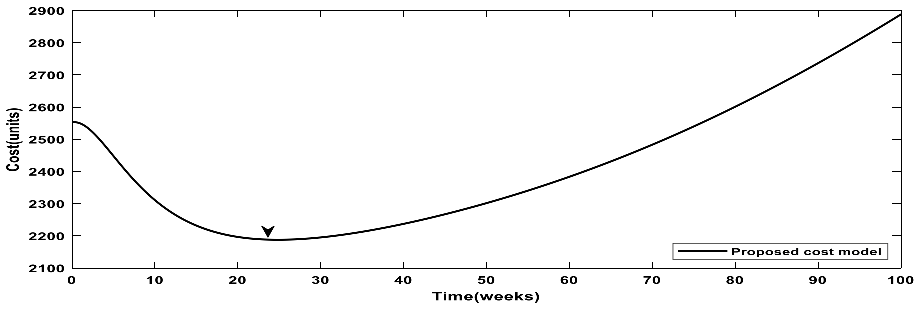

Fig. 15 Predicted optimal release time Release 1 for Mozilla data set

Table 5 Optimal release time and corresponding cost for Tandem data set

Releases	Optimal Time (weeks)	Optimal cost (units)
1st release	20.12	2209
2nd release	19.44	2973
3rd release	14.79	2006
4th release	20.05	1908

Table 6 Optimal release time and corresponding cost for Mozilla data set

Releases	Optimal Time (weeks)	Optimal cost (units)
1st release	13.34	1799
2nd release	27.51	2030
3rd release	24.03	2188

6 Conclusion

To enhance reliability and to estimate faults of software, many SRGMs have been constructed based on various realistic issues. But there is no such SRGM based on two very important realistic issues like: testing coverage and effect of testing environment on growth models for multi-release problem are available. Fault detection rate can change due to the presence of different kinds of faults. In this article, a NHPP based SRGM has been proposed considering imperfect debugging, faults removal efficiency, two types of testing coverage, and effects of uncertain testing environment on SRGMs for multi-release problems. Also, the presence of two different types of faults: soft fault and hard fault has been considered here. In formulating the cost model, the concept of random debugging cost has been incorporated. To the software organizations, reliability is not only main issue; determination of optimal release time of the software is also an important issue. Due to this reason, a cost model has also been carried out to find the optimal time of release at an affordable cost. In brief, the major contribution of the proposed work is: consideration of the effect of testing coverage and random field environment in the reliability growth of multi-release software in the presence of two different types of faults and consideration of random debugging cost for cost modelling. Performance analysis and comparison with existing SRGMs show that the proposed model is better fitted than existing models for multi release problem. Therefore, it can be concluded that the proposed model is more realistic, and it will be more useful for reliability analysis of multi up-gradation problem.

Acknowledgements Authors are thankful to Indian Space Research Organization, funding the project. Also, the authors are thankful to Indian Institute of Technology (ISM), Dhanbad, India for providing facility.

Funding The author(s) received funding from Indian Space Research Organization for the research.

Declarations

Conflict of interest The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

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

Chatterjee, S., Singh, J. B., Roy, A., & Shukla, A. (2018). NHPP-Based Software Reliability Growth Modeling and Optimal Release Policy for N-Version Programming System with Increasing Fault Detection Rate under Imperfect Debugging. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, 1–16.

Chatterjee, S., Chaudhuri, B., & Bhar, C. (2019). Optimal release time determination in intuitionistic fuzzy environment involving randomized cost budget for SDE-based software reliability growth model. *Arabian Journal for Science and Engineering*, 1–21.

Chatterjee, S., Misra, R. B., & Alam, S. S. (1997). Joint effect of test effort and learning factor on software reliability and optimal release policy. *International Journal of Systems Science, 28*(4), 391–396.

Chatterjee, S., Nigam, S., Singh, J. B., & Upadhyaya, L. N. (2012). Effect of change point and imperfect debugging in software reliability and its optimal release policy. *Mathematical and Computer Modelling of Dynamical Systems, 18*(5), 539–551.

Chatterjee, S., & Shukla, A. (2016). Effect of Test Coverage and Change Point on Software Reliability Growth Based on Time Variable Fault Detection Probability. *JSW, 11*(1), 110–117.

Chatterjee, S., & Shukla, A. (2019). A unified approach of testing coverage-based software reliability growth modelling with fault detection probability, imperfect debugging, and change point. *Journal of Software: Evolution and Process, 31*(3), e2150.

Chatterjee, S., & Singh, J. B. (2014). 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), 399–406.

Garmabaki, A. H., Aggarwal, A. G., & Kapur, P. K. (2011). Multi up-gradation software reliability growth model with faults of different severity. In *2011 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 1539–1543). IEEE.

Ho, J., & Ruhe, G. (2013). Releasing sooner or later: An optimization approach and its case study evaluation. *In Proceedings of the 1st International Workshop on Release Engineering* (pp. 21–24). IEEE Press

Kapur, P. K., Kumar, S., & Garg, R. B. (1999). *Contributions to hardware and software reliability* (Vol. 3). World Scientific.

Kapur, P. K., Tandon, A., & Kaur, G. (2010). Multi up-gradation software reliability model. In *2010 2nd International Conference on Reliability, Safety and Hazard-Risk-Based Technologies and Physics-of-Failure Methods (ICRESH)* (pp. 468–474). IEEE.

Kapur, P. K., Pham, H., Gupta, A., & Jha, P. C. (2011). *Software reliability assessment with OR applications* (p. 364). Springer.

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

Kapur, P. K., Singh, O., Garmabaki, A. S., & Singh, J. (2010b). Multi up-gradation software reliability growth model with imperfect debugging. *International Journal of System Assurance Engineering and Management, 1*(4), 299–306.

Li, L., Harman, M., Letier, E., & Zhang, Y. (2014). Robust next release problem: Handling uncertainty during optimization. In *Proceedings of the ACM 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 1247–1254).

Li, Q., & Pham, H. (2017). NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage. *Applied Mathematical Modelling, 51*, 68–85.

Li, X., Li, Y. F., Xie, M., & Ng, S. H. (2011). Reliability analysis and optimal version-updating for open-source software. *Information and Software Technology, 53*(9), 929–936.

Naciri, S., Idrissi, M. A. J., & Kerzazi, N. (2015). A strategic release planning model from TPM point of view. *In 2015 10th international conference on IEEE intelligent systems: Theories and applications (SITA)* (pp. 1–9).

Pachauri, B., Dhar, J., & Kumar, A. (2015). Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth. *Applied Mathematical Modelling, 39*(5), 1463–1469.

Pham, H. (2001). Software reliability. *Wiley Encyclopedia of Electrical and Electronics Engineering*.

Pham, H. (1995). *Software reliability and testing*. IEEE Computer Society Press.

Pham, H. (2007). *System software reliability*. Springer Science & Business Media.

Pham, H., Nordmann, L., & Zhang, Z. (1999). A general imperfect-software-debugging model with S-shaped fault-detection rate. *IEEE Transactions on Reliability, 48*(2), 169–175.

Pham, H., & Zhang, X. (1997). An NHPP software reliability model and its comparison. *International Journal of Reliability, Quality and Safety Engineering, 4*(03), 269–282.

Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *European Journal of Operational Research, 145*(2), 443–454.

Saliu, O., & Ruhe, G. (2005). Software release planning for evolving systems. *Innovations in Systems and Software Engineering, 1*(2), 189–204.

Szöke, Á. (2011). Conceptual scheduling model and optimized release scheduling for agile environments. *Information and Software Technology, 53*(6), 574–591.

Teng, X., & Pham, H. (2006). A new methodology for predicting software reliability in the random field environments. *IEEE Transactions on Reliability, 55*(3), 458–468.

Wood, A. (1996). Predicting software reliability. *Computer, 29*(11), 69–77.

Yamada, S. (2014). *Software reliability modeling: Fundamentals and applications* (Vol. 5). Springer.

Yamada, S., Hishitani, J., & Osaki, S. (1993). Software-reliability growth with a Weibull test-effort: A model and application. *IEEE Transactions on Reliability, 42*(1), 100–106.

Yamada, S., Ohba, M., & Osaki, S. (1984). S-shaped software reliability growth models and their applications. *IEEE Transactions on Reliability, 33*(4), 289–292.

Yamada, S., & Osaki, S. (1985). Software reliability growth modeling: Models and applications. *IEEE Transactions on Software Engineering, 12*, 1431–1437.

Yang, J., Liu, Y., Xie, M., & Zhao, M. (2016). Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes. *Journal of Systems and Software, 115*, 102–110.

Zhang, X., Teng, X., & Pham, H. (2003). Considering fault removal efficiency in software reliability assessment. *IEEE Transactions on Systems, Man, and Cybernetics-Part a: Systems and Humans, 33*(1), 114–120.

Zhu, M., & Pham, H. (2018a). A multi-release software reliability modeling for open source software incorporating dependent fault detection process. *Annals of Operations Research, 269*(1–2), 773–790.

Zhu, M., & Pham, H. (2018b). A software reliability model incorporating martingale process with gamma-distributed environmental factors. *Annals of Operations Research*. https://doi.org/10.1007/s10479-018-2951-7