International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

Testing Resource based Optimal Release Policy for Software System Incorporating Fault Reduction Factor and Change point

Rajat Arora, Abhishek Tandon, Anu G. Aggarwal, Vibha Verma

Abstract: ***In today's competitive scenario, it is essential for software developers to perform rigorous testing. It helps them to satisfy the demand for reliable software systems. Various external and internal factors like human expertise, fault dependency, code complexity, dynamic approaches etc. affect the process of fault removal. Hence, along the time-line fault removal rate may change. The point on time-line beyond which rates are altered is termed as the change point. Also in many practical situations, the number of failures experienced may not coincide with the number of faults removed from the system. This ratio is computed by Fault Reduction Factor (FRF). Here, we have proposed testing effort based model considering effort-dependent FRF with and without change point for gauging the failure pattern of a software system. The FRF has been modelled by logistic curve. The developed models have been verified using real-life software fault datasets. Model parameters are estimated and various performance criteria are employed to check the goodness of fit. Later, we have developed a software cost model to determine the optimal Release testing effort that minimizes total expected cost of fault removal during testing phase and operational phase of software life cycle subject to a reliability constraint. A numerical study has been also taken to demonstrate the results. Cost sensitivity analysis has been carried out to identify the crucial cost component and role of each cost component on optimal testing effort and overall cost of development.***

Keywords: Change Point, Cost Sensitivity Analysis, FRF, Logistic Function, Release Policy, SRGM, Testing Effort, Weibull Function.

I. INTRODUCTION

Rapid increase in demand for qualitative software systems has led to competitive market conditions and hence revolutionary change in the development process. Developers are more concerned about controlling the cost and effort employed in the development. In today's mechanized environment, we are increasingly becoming software dependent by using it in diverse fields including medicine, education, data-processing, military, forecasting, real-time control systems and the list is endless. The major challenge faced by IT firms is the inescapability of defects in the software which can occur during any phase of development.

Developers aim to develop reliable software. Reliability is defined as "probability that given software functions without failure under given environmental condition during specified interval of time" [1]. Researchers and software developers assess the reliability of software systems before delivering it into the market. Some of the study in this direction focused on reliability assessment through architectural models [2, 3]. On the other hand many others developed mathematical models that captured the failure phenomenon of software systems. The Non Homogenous Poisson Process (NHPP) based Software Reliability Growth Models (SRGMs) are the widely employed models for assessing reliability of software. Various NHPP models in literature evaluate the failure phenomenon of software systems through these mathematical models [4, 5]. Some of the recent work in this domain has been done by Aggarwal, Gandhi, Verma and Tandon [6],Chatterjee and Shukla [7], Anand, Verma and Aggarwal [8], Aggarwal, Dhaka and Nijhawan [9] and many more. SRGMs are used to quantify the growth in software reliability. These models depict relationship between the cumulative fault count and the testing time. These models did not incorporate the effect of some realistic factors encountered during testing and debugging process such as FRF, change point and testing effort into the single model to analyze the failure phenomenon.

Fault removal process is affected by various factors like changing strategies, tester's skill, time difference between detection of fault and its removal, fault dependency, relationship between fault and failure, testing tools used etc. The failure process is highly affected by testing conditions. These changes may lead to change in rate of fault detection over time. The time moment at which the fault detection rate changes is referred to as Change point. At this kind we find change in the direction of the failure curve (Fig. 1). Hence the failure phenomenon, before and after change point is modelled separately. Initially change point concept was studied by [10, 11]. It is highly important for every software firm to take appropriate regarding effort consumption during software development.

Manuscript published on November 30, 2019.
* Correspondence Author
Rajat Arora, Department of Operational Research, University of Delhi, Delhi, India. Email:arorarajt87@yahoo.com
***Abhishek Tandon**, Deputy Director, Indian Council of Social Sciences Research, Delhi, India. Email: abhishektandon86@gmail.com
Anu G. Aggarwal, Department of Operational Research, University of Delhi, Delhi, India. Email:anuagg17@gmail.com
Vibha Verma, Department of Operational Research, University of Delhi, Delhi, India. Email:vibhaverma.du.aor@gmail.com

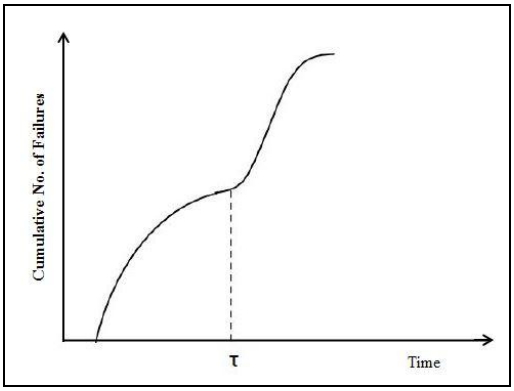


Fig. 1. Change Point (τ) Representation

Another aspect of software development that affects the software performance is amount of effort spent during testing. This is known as testing effort and it is expressed as the test cases considered for testing, testing time period, workforce employed etc. Testing effort required in the beginning of software development process is in small amount. With time effort requirement increases and reaches up to peak as more faults get detected and corrected. After reaching a high level the requirements starts declining and reaches at its low. Determining the right amount of effort spend on the testing process is very crucial for developing reliable software [12]. Therefore, a developer needs to control the consumption of resources. Testing effort consumed during the development process is has high correspondence with the reliability level of the software system. Time dependent testing effort cab be defined as the resources used during the whole time horizon [13]. They are defined in the form of functions known as Testing Effort Functions (TEFs). Huang and Lyu [14] introduced the change point into the testing effort dependent SRGM. Huang [15] also integrated change point with testing effort by considering logistic TEF and change point.

Musa [16] introduced a noteworthy factor that affects the fault removal process during testing of software system during practical situations. It is defined "as the percentage of net number of faults removed to the number of failures experienced". Initially researchers assumed FRF to be constant [17]. Later Hsu, Huang and Chang [18] proposed a study considering FRF to be increasing, decreasing and constant. Practically, FRF is affected by various factors viz, resource allocation, defect density, fault dependency, environment, time lag in removing faults, learning process, error generation etc. Pachauri, Dhar and Kumar [19] defined FRF as S-shaped function for software released in multiple versions. Several other definitions of FRF were coined by some researchers which did not become much popular in the research community Friedman, Tran and Goddard [20] defined FRF in terms of detectability, associability and fault growth and Malaiya, Von Mayrhauser and Srimani [21] defined FRF in terms of fault exposure ratio.

Study of software systems reliability is not limited to analyzing its growth and fault prediction at the end of testing period. Another significant area of the study is determination of the optimal testing time period or optimal effort before software release. The objective of problems can be either development cost minimization or reliability maximization subject to reliability and budget constraint respectively. Release time problems have been given more attention in literature while determination optimal effort is a less explored area. Some well-known models is this respect are Leung [22],Huang, Luo and Lyu [23],Inoue and Yamada [24] and Sachdeva, Kapur and Shrivastava [25]. High reliability aspiration requires more effort. If the set reliability goal is achieved with available efforts then the reliability constraint becomes redundant.

In this study, we incorporate the above discussed factors TEF, Change point and FRF into the modelling framework to develop better mathematical model that captures the practical failure trends of software system. The proposed model considers single change point, TEF modeled by Weibull curve and the logistic FRF. The results of the proposed model are validated on two real-life failure datasets. The performance results for with and without change point models are observed measures like R^2, Mean square error, predictive power, predictive ratio risk and root mean square prediction. The paper also develops release policy model to determine the optimal effort required during the software development process with the objective of minimizing the overall development cost under some reliability constraint. Further, sensitivity analysis of cost parameters is performed to identify the effect of cost parameters on required testing effort and software cost development. Also, we have performed sensitivity over the reliability goal to understand how increase and decrease in aspiration impacts the effort requirement.

Remainder of the paper is as follows; the next section presents the literature work corresponding to the concepts discussed in the paper. Section three discusses the proposed reliability growth model followed by its validation on two fault datasets in section four. In section five, software development cost model is proposed for determining optimal release policy. Section 5 presents the numerical illustration of proposed cost minimization problems. At the end the paper is concluded and future scope is presented.

II. LITERATURE REVIEW

Here in this section, we will discuss literature related to the basic concepts of the research study. Literature helps to gain insights regarding the importance and relevance of the study being conducted.

A. Testing Effort

Study of testing efforts is very important task of software development process. Effort can be seen as anything that is required to carry out the task of testing [26]. It can be man-power, tools and techniques, facilities etc. required to meet the testing goal. Several researchers have measured testing effort in terms of CPU hours, executed test cases or human effort in hours etc. [27]. Testing effort is influenced by factors like quality documentation with detailed information of testing progress, size of application software, type of software development life cycle selected for testing , tester's skill bonding among the team members etc. [26, 27]. In literature consumption of testing resources has been modelled using different functions as listed in Table I.

The notations used to define the testing effort functions are defined as follows:

N:Total amount of testing effort available

b, δ: Shape parameters
m: Scale parameter
A: Constant
ρ: Rate of testing-effort consumption

Table I: Testing-Effort Functions proposed in Literature

Expression	Testing-Effort Function	Reference
$N(1-e^{-bt})$	Exponential	Yamada, Hishitani and Osaki [28]
$N(1-e^{\left(-\frac{b}{2}\right)t^2})$	Rayleigh	Yamada, Hishitani and Osaki [28]
$N(1-e^{-bt^m})$	Weibull	Yamada, Hishitani and Osaki [28]
$\frac{N}{1+Ae^{-\alpha t}}$	Logistic	Huang, Kuo and Chen [29], Huang and Kuo [13], Huang, Kuo and Lyu [30]
$N\frac{(mt)^b}{1+(mt)^b}$	Log-Logistic	Bokhari and Ahmad [31], Gokhale and Trivedi [32]
$N(1-e^{-\alpha t})^b$	Generalized Exponential	Quadri, Ahmad, Peer and Kumar [33]
$N(1-e^{-mt^\delta})^b$	Exponentiated Weibull	Ahmad, Khan and Rafi [34]
$N\frac{1-e^{-\alpha t}}{1+Ae^{-\alpha t}}$	Inflexion S-Shaped	Li, Li, Lu and Wang [35]

Some of the recent testing effort dependent SRGMs are Inoue and Yamada [36], Khatri, John and Majumdar [37], Bokhari, Siddiqui and Ahmad [38], Gupta, Saxena and Achrya [39]. Inoue and Yamada [36] proposed lognormal testing effort basd SRGM that represents continuous state space stochastic process. The authors also compared the TEF with the Weibull TEF. Khatri, John and Majumdar [37] developed testing effort dependent SRGM under imperfect debugging environment. Bokhari, Siddiqui and Ahmad [38] proposed a model considering Weibull TEF under imperfect debugging environment considering time difference between fault detection and removal process.

B. Change Point

There are several factors that may influence the failure process of software system. To list, some are testing environment under which testing is performed, tools used, methodology/strategy adopted, skill/efficiency and constitution of testing team, effectiveness of test cases selected for testing and resource allocation [27, 40]. The failure process is assessed through SRGMs [8, 41-43]. Parameters of these models are considered to depict the testing and debugging conditions. It is also assumed that the model parameters will not change throughout the testing. But under practical conditions this is not the case because after some point during testing, the management may take decisions to add some member to the team or change the methods used for testing which will eventually result in change in rate by which faults are detected. The time point where rate of fault detection changes is termed as change point. It is dependent on tester's skill, software testability, code size of application software, code expansion factor, defect density and testing efforts in terms of CPU hours [34, 37, 40].

Incorporating change points into the reliability models helps to improve the prediction accuracy of SRGMs. Zou [10] proposes a change point estimator for analyzing reliability of software systems. Zhao and Wang [11] discussed the change point problems in modelling to capture the failure phenomenon for software and hardware systems. Aggarwal, Dhaka and Nijhawan [9] proposed a SRGM considering TEF and change point. They also performed sensitivity analysis on release time of a software system. Inoue and Yamada [45] compared the failure phenomenon of software system before and after change during testing environment. Lin and Huang [46] analyzed the performance of reliability growth models that considered Weibull TEF and multiple change points. He considered that the rate of fault detection between failure occurrences may change at a point.

C. FRF

Musa [17] coined the term FRF defining it as ratio of failures experienced to number of failures removed. Musa Musa, Iannino and Okumoto [47] further explored the concept of FRF through its basic execution time model. He expressed FRF as:

$$FRF = \frac{Number\ of\ failures\ experienced}{Number\ of\ Faults\ removed} \quad (1)$$

Several other definitions of FRF also came into light. Malaiya, Von Mayrhauser and Srimani [21] defined FRF in terms of fault exposure ratio. Friedman, Tran and Goddard [20]defined FRF in terms of three ratios namely detectability, associability and fault growth. Among these definitions, Musa's version of FRF became very popular in the research community. After Realizing the importance of FRF in affecting the failure behavior, several NHPP based SRGMs were proposed considering FRF as one of the key component [6, 8, 9, 18, 19, 48]. These studies also captured the FRF trend.

Most of the studies considered FRF to be constant with value lying between 0 and 1. Hsu, Huang and Chang [18] proposed SRGM with constant, increasing and decreasing curves of FRF and validated these trends on six real life fault datasets. They also plotted the FRF trends followed by each dataset. Pachauri, Dhar and Kumar [19]considered Inflexion S-Shaped FRF to increase accuracy of growth model. The authors also discussed the multi-release model considering the proposed FRF. Later,Aggarwal, Dhaka and Nijhawan [9] proposed Exponentiated Weibull FRF based SRGM with change point. Recently Aggarwal, Gandhi, Verma and Tandon [6] considered a Delayed S-shaped FRF based SRGM to model the failure process.

D. Optimal Release Policy

One of the significant areas of study in Software reliability engineering is to determine testing stop time and release time for software into the market taking into consideration the organization's goal with respect to reliability target. The management of any IT firm is concerned about minimizing the cost of development while maintaining the reliability of the system. Several release policies have been proposed in the past using the available failure models. Okumoto and Goel [49] proposed the optimal release policy by minimizing development cost. Kapur, Pham, Aggarwal and Kaur [50]

proposed a release planning model to obtain the optimal time and effort required for software development by minimizing the cost of development. Optimization problems for release time were discussed by Kumar, Kapur, Shrivastava and Sharma [51], Lyu [52], Pham [53], Anand, Agarwal, Tamura and Yamada [54], Xie [55]. Ehrlich, Prasanna, Stampfel and Wu [56] determined the cost involved in testing. Pham [57] proposed a cost optimization model with penalty cost for failures occurring after release. Hou, Kuo and Chang [58] obtained optimal release time with scheduled delivery.

All these models were only concerned about the time of release. Apart from time the resources spent on the testing and debugging process also needs to be optimized. The amount of effort expenditure affects the cost of development cost and reliability level of the software system. Many a times the target reliability is set for the system before releasing into the market. But the target reliability is only achievable if right amount of time and effort is employed during testing process. This may possibly be result of inadequate amount of testing effort employed during testing and incompetence of testing team. The optimal effort planning helps to find testing effort required to expedite the fault removal process. Few studies related to analysis of effort required are Kapur, Aggarwal, Kapoor and Kaur [59], Huang and Lyu [60], Aggarwal, Kapur, Kaur and Kumar [61]etc.

III. MODELLING FAULT REMOVAL PROCESS (FRP)

Notations used for development of FRP are as follows:

$W(t)$	Testing effort spent by time t
$m(W)$	Mean value function (MVF) corresponding to testing effort $W(t)$
a	Faults initially present in the software
r	Fault detection rate (FDR)
$\overline{W}$	Total available testing effort expenditure
$k, \propto$	Shape and scale parameters of Weibull TEF
$B(W)$	Testing-effort dependent fault reduction factor
$r(W)$	Testing –effort based FRF
b_1	Scale parameter before Change point
b_2	Scale parameter after Change point
r_1	FDR before Change point
r_2	FDR after Change point
l_1	Shape parameter before Change point
l_2	Shape parameter after Change point

The proposed model is based on following assumptions:

1. Fault removal process follows NHPP.
2. The faults present in the system cause random failures of the software system.
3. All the faults present in the system are mutually independent.
4. Failure rate is consistently affected by the residual faults in the software.
5. The number of faults removed is directly proportional to the faults latent in the system.
6. Fault detection /removal rate may change at any time moment known as change point.
7. The proportionality factor is given by FRF modelled using logistic function.
8. The TEF is modelled through Weibull-function.

The MVF, $m(W)$ for testing effort dependent SRGM is given by:

$$\frac{d}{dt}m(W) = r(W)\big(a - m(W)\big)\ \frac{dW}{dt} \quad (2)$$

Where, $r(W) = r \times B(W)$

where, $r(W)$ and $B(W)$ are the testing effort dependent fault detection rate function and FRF respectively.

A. Weibull TEF

The TEF, $W(t)$ is modelled using Weibull function is given as:

$$W(t) = \overline{W} \times (1 - e^{-\alpha t^k}) \quad (3)$$

where, $0 < \propto < 1 \quad , \quad k > 0$

The Instantaneous rate of effort consumption is given by:

$$\frac{d}{dt}W(t) = w(t) = \overline{W}\alpha k t^{k-1} e^{-\alpha t^k} \quad (4)$$

B. Time-dependent FRF

We consider the FRF to follow logistic distribution.

$$B(W) = \frac{l}{(1+be^{-lW})} \quad (5)$$

where, l and b denote the shape and scale parameter respectively of logistic function.

Case 1: Without change point.

The differential equation corresponding to without change point model is:

$$\frac{d}{dW}m(W) = \frac{dm/dt}{dW/dt} = r(W)(a - m(W)) \quad (6)$$

$$\frac{d}{dW}m(W) = r \times B(W)\ \big(a - m(W)\big) \quad (7)$$

$$\frac{d}{dW}m(W) = r \times \frac{l}{(1+be^{-lW})}\ \big(a - m(W)\big) \quad (8)$$

On integrating using initial condition $m(0) = 0$ we get,

$$m(W) = a\left[1 - \frac{(1+b)^r e^{-lrW}}{(1+be^{-lW})^r}\right] \quad (9)$$

Case 2: With change point

The differential equation for model with change point is as:

$$\frac{d}{dt}m(W) = r \times B(W)\ \big(a - m(W)\big)\ \frac{dW}{dt} \quad (10)$$

Where,

$$r = \begin{cases} r_1 & t \le \tau \\ r_2 & t < \tau \end{cases}$$

$$B(W) = \begin{cases} \dfrac{l_1}{1 + b_1 e^{-l_1 W}} & ; t \le \tau \\ \dfrac{l_2}{1 + b_2 e^{-l_2 W}} & ; t > \tau \end{cases}$$

On integrating the above differential equation, we get:

$$m(W) = a\left[1 - \frac{(1+b_1)^{r_1} e^{-l_1 r_1 W}}{(1+b_1 e^{-l_1 W})^{r_1}}\right];\ \ t \le \tau \quad (11)$$

$$m(W) = a\left[1 - \frac{(1+b_1)^{r_1 e^{-l_1 r_1 W(\tau) - l_2 r_2 (W - W(\tau))}} \left(\frac{1+b_2 e^{-l_2 W(\tau)}}{1+b_2 e^{-l_2 W}}\right)^{r_2}}{\left(1+b_1 e^{-l_1 W(\tau)}\right)^{r_1}}\right] ; \quad t > \tau \quad (12)$$

IV. MODEL VALIDATION

Accuracy of the proposed model in predicting the faults is validated using two real-life fault datasets. The two datasets considered are presented in Table II.

Table II: Datasets

Dataset	Reference	Description	Testing time (Weeks)	Execution time (CPU hours)	Faults
DS-1	[Wood [62]]	Tandem Computers	20	10000	100
DS-2	Ohba [63]	PL/I database application software system	19	47.65	328

For both the datasets firstly, we predict the effort consumption corresponding to Weibull TEF (Eq. 3). The estimated parameter values for Weibull TEF are given in Table III.

Table III: Estimated parameters of TEF

Parameters	DS-1	DS-2
$\overline{W}$	11710.754	799.016
α	0.024	0.002
k	1.46	1.115

We focus on data related to testing-effort which is given by execution time in CPU hours per week. Fig. 2 shows the goodness of fit curves for TEF.

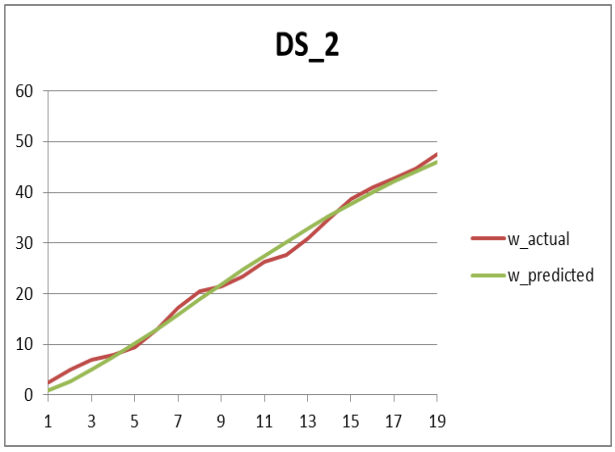


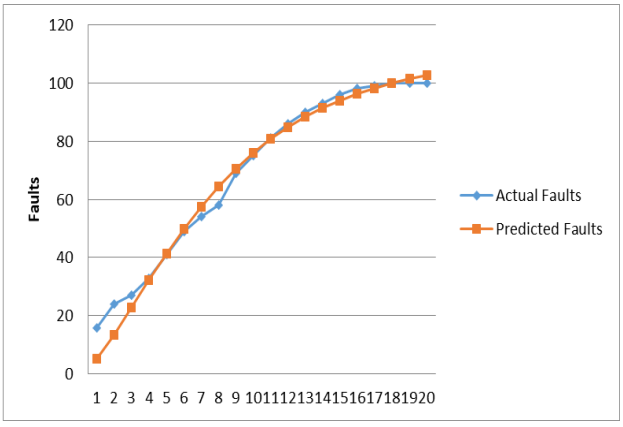
Fig. 2. Goodness of fit curves for testing effort corresponding to two datasets

Based on the predicted TEF, we further predict the faults based on the proposed models with and without change point (Eq. 11 and Eq. 12). These predicted faults are corresponding to effort consumed. The estimation results for the both the models are shown in Table IV. Fig. 3 and Fig. 4 depicts curves of predicted and actual faults for the two datasets respectively.

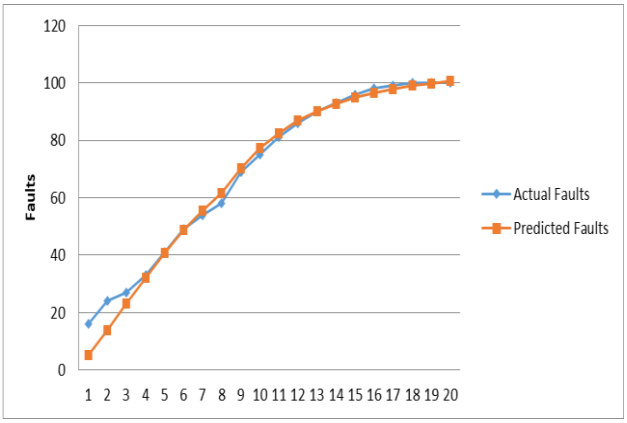
Table IV: Estimated Parameters for the two models

Model-1 Parameters	DS-1	DS-2
a	110.751	567.473
b	0.945	0.055
r	0.005	0.044
l	0.043	0.443

Model-2 Parameters	DS-1	DS-2
a	108.947	353.656
b_1	0.044	0.003
b_2	0.770	137.64
r_1	0.004	0.071
r_2	0.032	0.373
l_1	0.042	0.518
l_2	0.010	0.238

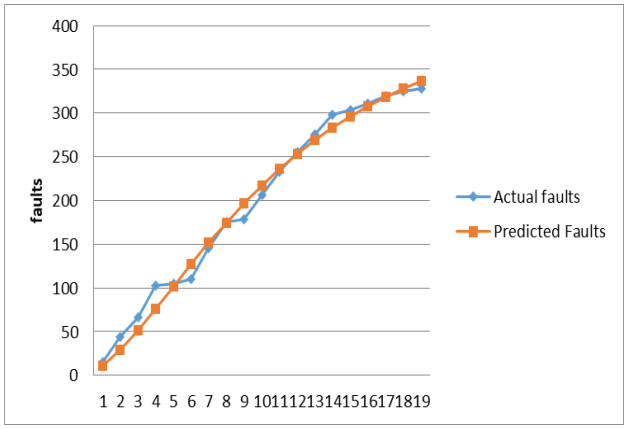


(a)

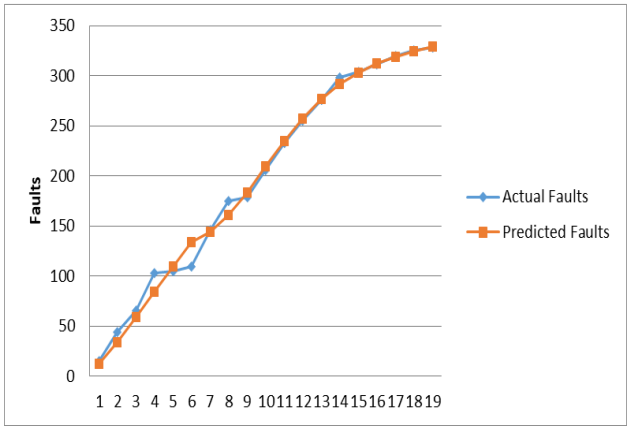


(b)

Fig. 3: Goodness of fit curve for DS-1 (a) without change point (b) with change point



(a)



(b)

Fig 4: Goodness of fit curve for DS-2 (a) without change point (b) with change point

To determine the accuracy of our proposed model we evaluate performance criteria values. Following comparison criteria are used in the study. R^2 , Predictive ratio Risk, Predictive Relative Variation, Predictive Power , Mean square Error and Root mean square prediction. For all the mentioned criteria except the first one, lower values signify better fit hence better accuracy of the model. The values of performance criteria are based on the predicted faults for each of the dataset corresponding to the both models is presented in Table V. We can observe that for all criteria's the model with change point has lower value except R^2. This implies model considering change point along with FRF and testing effort gives better fitting.

Table V: Model Performance Comparison

Model	Dataset	R^2	MSE	PP	PRR	PRV	RMSPE
1	DS-1	0.965	16.58	13.99	33.27	4.07	4.17
	DS-2	0.988	117.39	23.04	28.96	10.99	11.10
2	DS-1	0.984	13.40	12.64	30.54	3.61	3.75
	DS-2	0.993	69.52	13.71	14.43	8.50	8.55

V. OPTIMAL RELEASE POLICY MODEL

In this section, we propose optimal release plan to determine effort requirements based on our developed models. For this purpose, we will use the estimated values of the parameters given in Table 4. In the table given below, we will define few additional notations that are used for modelling optimization model.

C_1 Per unit cost of removing fault before change point during testing phase

C_2 Per unit cost of removing fault after change point during testing phase

C_3 Per unit cost of removing fault after release of software product

C_4 Cost per unit time of testing during testing period

R_O Reliability Goal

$m1$ MVF for FRP before change point

$m2$ MVF for FRP after change point

Here, the objective is minimization of the total expected software development cost. It consists of different cost components of testing and operational phase. The cost components are discussed in detail below:

The various components of cost function are as follows:

- Testing phase cost

The cost incurred due to fault removal in testing phase is given by:

$$= C_1 m1\big(W(\tau)\big) + C_2\Big(m2\big(W(t)\big) - m1\big(W(\tau)\big)\Big) \quad (13)$$

- Operational phase cost

The cost to remove faults operational phase is given by:

$$= C_3(a - m2(W(t))) \quad (14)$$

- Cost of Testing due to testing effort W

It is assumed that the cost of testing is a linear function of effort W

Expected cost of testing $= C_4 W$ (15)

The constrained and unconstrained optimization problems formulated using expressions (13)-(15) are given below:

I. Unconstrained Optimization problem (O_1)

$$Minimize\ Z(W) = C_1 m1(W(\tau)) + C_2\left(m2(W(t)) - m1(W(\tau))\right) + C_3(a - m2(W(t))) + C_4 W$$

$$W(t) \geq 0$$

II. Constrained Optimization problem (O_2)

$$Minimize\ Z(W) = C_1 m1(W(\tau)) + C_2\left(m2(W(t)) - m1(W(\tau))\right) + C_3(a - m2(W(t))) + C_4 W$$

$$subject\ to$$

$$R\left(\frac{x}{W}\right) \geq R_O$$

$$W(t) \geq 0$$

The reliability $R\left(\frac{x}{W}\right)$ of the proposed model is given as follows $= m2(W(t))/a$.

VI. COST OPTIMIZATION

To demonstrate the optimal release policy, we will determine the optimal effort required to develop a reliable software system. For this purpose, we have utilized estimated parameter values for two models (Eq. 11 and Eq. 12) are corresponding to DS-1 and DS-2 given in Table 4. Here we have assumed cost values based on literature. The values for solving O_1 corresponding to DS-1 are:

$$\tau = 8, W(\tau) = 4536.39, m1(W(\tau)) = 65.85,$$
$$C_1 = 90, C_2 = 100, C_3 = 1500, C_4 = 10$$

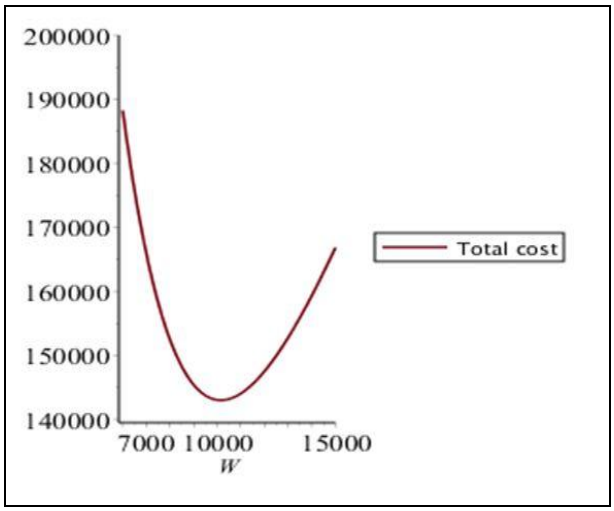
Using the above values the optimization problem is solved in Maple. The optimal testing effort is obtained as $W_0 = 7116.077$ and the corresponding minimum cost is $C_0 =$ \$112646.922

For solving problem O_2 the target reliability to be achieved is set as R_O =0.98. The results are obtained as $W_0 = 14387.036\ units$ and the corresponding minimum cost is $C_0 =$ \$157157.082. The cost curve corresponding to these solutions is shown in Fig. 5.

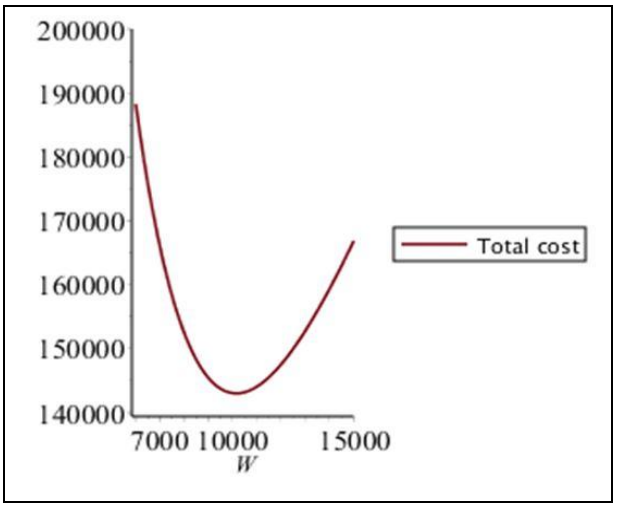
Now, we solve the optimization problems for DS-2. The estimated parameter values form Table IV is substituted in eq. 11 and eq. 12. For O_1 the assumed cost values are as follows:

$$\tau = 6, W(\tau) = 12.99, m1(W(\tau)) = 126.9$$
$$C_1 = 9, C_2 = 10, C_3 = 15, \quad C_4 = 5$$

Solving the problem in Maple, we obtain $W_0 = 54.782$ and the corresponding minimum cost is $C_0 =$\$ 3800.970. To solve O_2 the target reliability to be achieved is set as R_O =0.98. As done for the DS-1. The optimal solution for the problem results to be $W_0 = 60.030$ and the corresponding minimum cost is $C_0 =$ \$3806.228. The corresponding cost curves for DS-2 solutions are shown in Fig. 6.

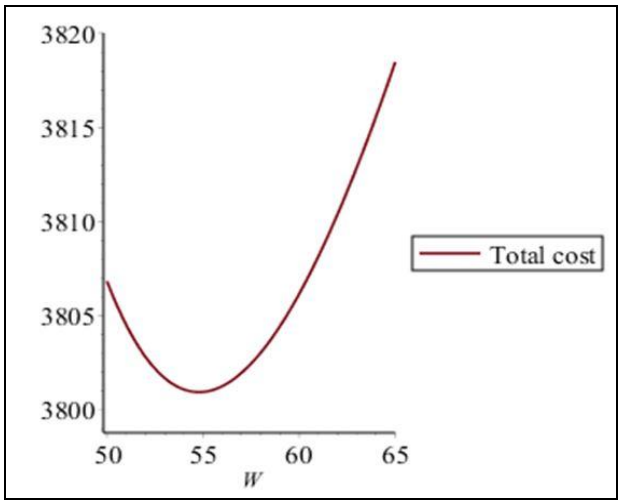


(a)

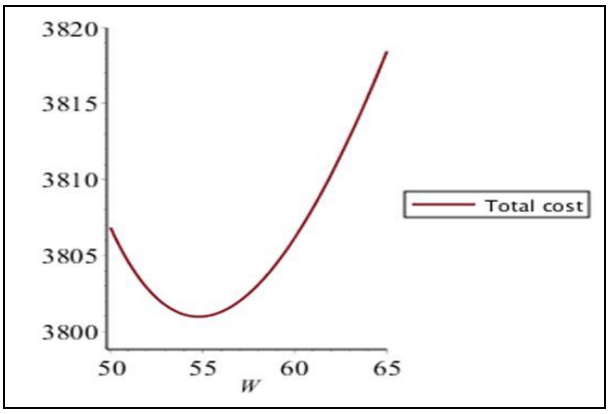


(b)

Fig. 5. Total cost curve for DS-1 (a) Unconstrained Model (b) Constrained Model



(a)



(b)

Fig. 6. Total cost curve for DS-2 (a) Unconstrained Model (b) Constrained Model

Next, we perform sensitivity analysis over each cost component to observe the corresponding relative percentage change in effort requirement and software development cost. Each of the cost component except C_1 is increased and decreased by 10% corresponding to both models and datasets. We have not performed sensitivity analysis with respect to cost component that represents cost of removing faults before change point because it acts as a constant in the optimization problem and does not affect the results. The relative change is measured using the following equation:

$$Relative\ \%\ Change = \frac{old\ value - new\ value}{old\ value} \times 100 \quad (16)$$

Table VI and Table VII discuss the sensitivity results for problem O_1 the unconstrained optimization problem corresponding to both the datasets. Table VIII and Table IX discuss the sensitivity results for problem O_2 the constrained optimization problem corresponding to both the datasets. In the sensitivity results for the constrained model, optimal effort requirement remains same throughout because adding reliability constraint makes it the more important aspect of the problem. Here, the aim to achieve the set reliability goal is attained by compromising with effort requirement. Fig. 7 and Fig. 8 show the relative change in total cost for both models corresponding to DS-1 and DS-2 respectively. Fig. 9 shows the percentage change in efforts for unconstrained model corresponding to both the datasets.

Table VI: Cost Sensitivity results for Unconstrained Model DS-1

Cost component	Original Cost	Cost after 10% increase	Testing Effort after 10% increase	Relative change in Testing Effort (%)	Total cost after 10% increase	Relative change in Total cost (%)
C_2	100	110	7093.67	-0.31	112853.88	0.18
C_3	1500	1650	7434.14	4.47	115827.63	2.82
C_4	10	11	6818.23	-4.19	119611.71	6.18
Cost factor	**Original Cost**	**Cost after 10% decrease**	**Testing Effort after 10% decrease**	**Relative change in Testing Effort (%)**	**Total cost after 10% decrease**	**Relative change in Total cost (%)**
C_2	100	90	7138.31	0.31	112438.37	-0.186
C_3	1500	1350	6761.92	-4.98	109105.40	-3.14
C_4	10	9	7445.32	4.63	105369.11	-6.46

Table VII: Cost Sensitivity results for Unconstrained Model-DS-2

Cost factor	Original Cost	Cost after 10% increase	Testing Effort after 10% increase	Relative change in Testing Effort (%)	Total cost after 10% increase	Relative change in Total cost (%)
C_2	10	11	52.27	-4.58	4076.20	7.241
C_3	15	16.5	57.74	5.40	3815.75	0.389
C_4	5	5.5	53.71	-1.95	3828.09	0.714
Cost factor	**Original Cost**	**Cost after 10% decrease**	**Testing Effort after 10% decrease**	**Relative change in Testing Effort (%)**	**Total cost after 10% decrease**	**Relative change in Total cost (%)**
C_2	10	9	56.84	3.76	3523.43	-7.302
C_3	15	13.5	50.76	-7.34	3780.87	-0.529
C_4	5	4.5	55.97	2.17	3773.29	-0.728

We can make following observations from Table VII and Table VII:

1) Increase in cost of fault removal after change point in testing phase leads to decrease in effort employed for testing and debugging process during testing whereas if there is subsequent decrease in the cost then more effort is required during testing.
2) Increase in fault removal cost during operational phase leads to increase in testing effort. This is so because if more faults are passed on to operational phase then it will cost more to the developers. Hence developers apply more effort in testing and debugging process and try to minimize the faults in the system. Similarly if there is decrease in fault removal cost during operational phase then developer spends fewer efforts on the testing process.
3) Increase/ Decrease in the fixed testing cost leads to decrease/ increase in efforts used for testing.
4) Also increase/decrease in any of the cost during software development leads to increment/ decrement in the overall cost respectively.

VII. RESULT AND DISCUSSION

The contents of the journal are peer-reviewed and archival. The journal publishes scholarly articles of archival value as well as tutorial expositions and critical reviews of classical subjects and topics of current interest.

Table VIII: Cost Sensitivity results for Constrained Model-DS-1

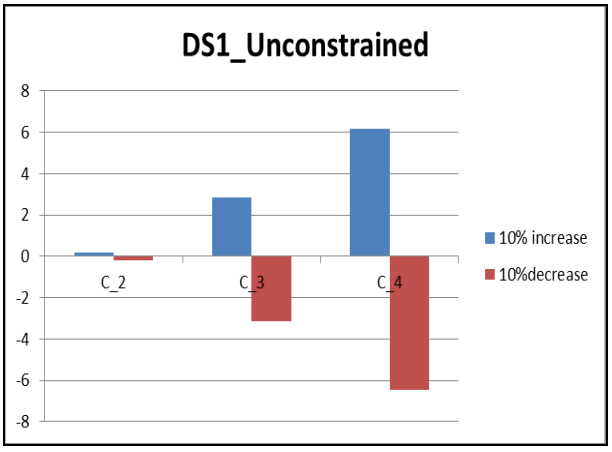
Cost factor	Original Cost	Cost after 10% increase	Total cost after 10% increase	Relative change in Total cost (%)
C_2	100	110	157566.26	0.26
C_3	1500	1650	157483.92	0.21
C_4	10	11	171544.12	9.15

Cost factor	Original Cost	Cost after 10% decrease	Total cost after 10% decrease	Relative change in Total cost (%)
C_2	100	90	156747.90	-0.26
C_3	1500	1350	156830.24	-0.21
C_4	10	9	142770.05	-9.15

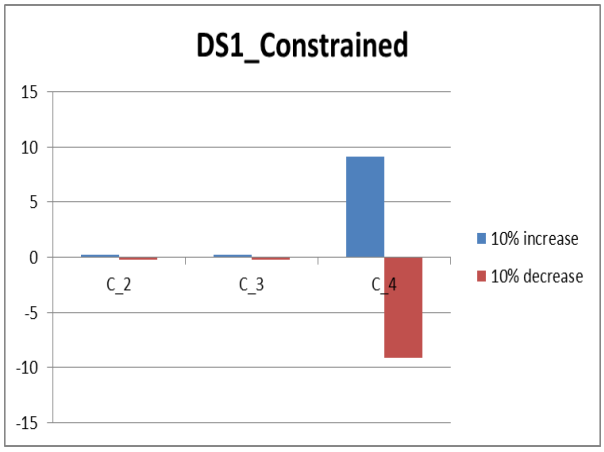
Table XI: Cost Sensitivity results for Constrained Model-DS-2

Cost factor	Original Cost	Cost after 10% increase	Total cost after 10% increase	Relative change in Total cost (%)
C_2	10	11	4086.96	7.376
C_3	15	16.5	3816.84	0.279
C_4	5	5.5	3836.24	0.788
Cost factor	**Original Cost**	**Cost after 10% decrease**	**Total cost after 10% decrease**	**Relative change in Total cost (%)**
C_2	10	9	3525.49	-7.376
C_3	15	13.5	3795.62	-0.279
C_4	5	4.5	3776.21	-0.788

The impact of change in cost parameters in case of unconstrained problem was observed both on effort and the overall cost of development. But here, in the case of constrained optimization we only observe changes in the cost of development irrespective of change in any cost parameter. This is so because in case of constrained problem focus turns towards the reliability level of the system which should be achieved at any cost and by applying the required amount effort. As is clear form Table VIII and Table XI that any kind of change in any of the cost component of constrained optimization model leads to change in the overall development cost.

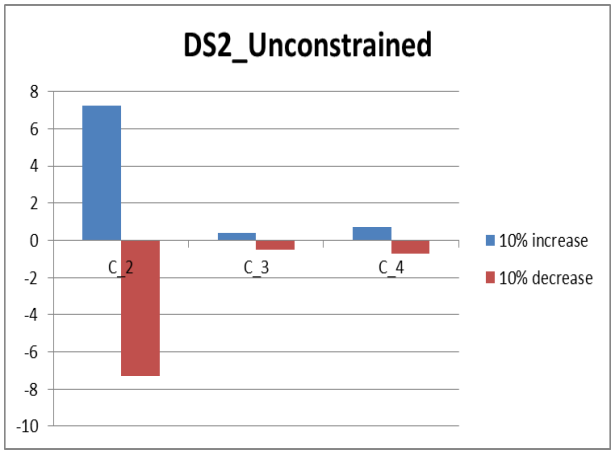


(a)

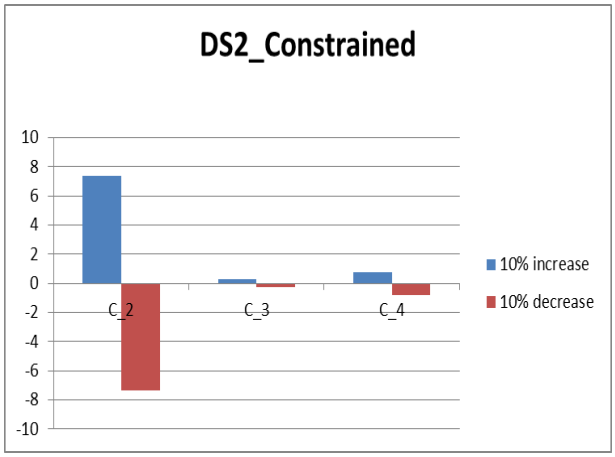


(b)

Fig. 7. Relative change in Total cost for (a) Unconstrained Model-DS-1(b) Constrained Model-DS-1

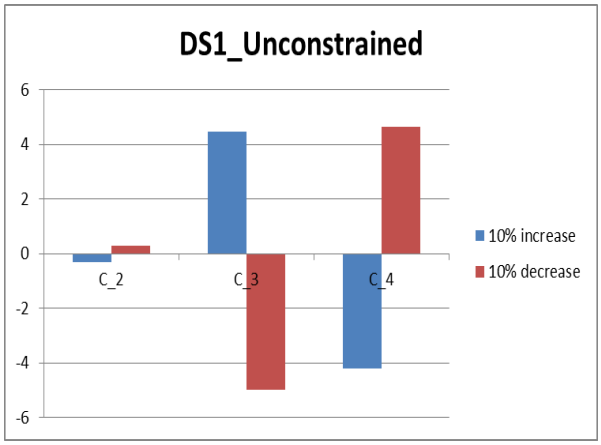


(a)

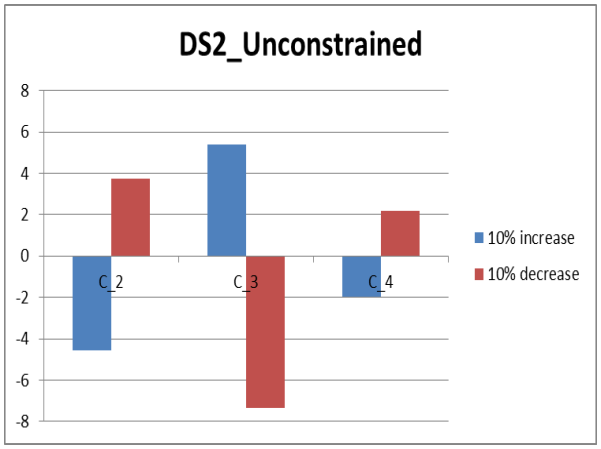


(b)

Fig. 8. Relative change in Total Cost for (a) Unconstrained Model-DS-2(b) Constrained Model-DS-2



(a)



(b)

Fig. 9. Relative change in Effort for (a) DS-1(b) DS-2 corresponding to unconstrained Model

We have also studied the impact of change in reliability goal. At first we obtain results corresponding to 0.98 reliability goal and then we obtain results with 0.95 reliability goal. We observe that at 0.95 the results are same as of unconstrained problem. For DS-1 we observe increase in the optimal effort value for constrained model when reliability goal is set at 0.95. For DS-2 the optimal effort value is same for both constrained and unconstrained problems when reliability is set at 0.95. This is so because with the available efforts developers are able to achieve the desired reliability for the system. There was no need of increasing the effort when goal is 0.95 and the constraint gets redundant.

VIII. CONCLUSION

In this paper, we have proposed SRGMs before and after change point based on Weibull TEF, FRF and Change point. First model considers there is no change in the FDR. While in the second model, we assume that at a particular time moment there is a change in the FDR (Change point). The testing effort function has been modeled by Weibull function and FRF is represented by the logistic function. The models are validated on two real fault datasets obtained from literature. The Least Square Estimation results and goodness-of-fit values show that model with change point has better performance and is more accurate in predicting faults. All the six criteria suggest better model fitting. The goodness of fit curves shows the closeness of actual and predicted values for both the effort function and the proposed model.

The study also presents optimal release policy for taking key decisions of planning effort requirements for software development. The objective of release planning problem is minimization of the overall cost of development. We have obtained effort requirement for unconstrained problem and for problem with reliability constraint. The optimal solution obtained for the unconstrained optimization problem for DS-1 is $W_0 = 7116.072$ and the corresponding minimum cost is $C_0 =$ \$ 112646.922. The results for constrained optimization problem are $W_0 = 14387.036$ and minimum cost is $C_0 =$ \$157157.082.

Similarly we do for DS-2 and obtain the optimal release testing effort for the unconstrained optimization problem as $W_0 = 54.784$ and cost is $C_0 =$ \$ 3800.970 and for the constrained optimization problem $W_0 = 60.030$ with minimum cost obtained as $C_0 =$ \$3806.228.

Next we study the sensitivity of cost components on the total cost of development and optimal effort requirement to attain the desired reliability level. We observe that change in any cost component leads to change in overall cost both for unconstrained and constrained optimization problems. But effort changes only for unconstrained model. Increase in debugging cost during testing leads to decrease is effort requirement while increase in debugging cost during operational phase leads to increase in effort required during testing. We also observe changes with respect to change in reliability level. To obtain higher reliability develop needs to spend more efforts.

This model can be extended to incorporate multiple change points, imperfect debugging conditions and multi-release software growth modelling. Many a times due to fault dependency more errors get generated during the debugging activity which also plays important role in the failure process.

REFERENCES

1. Pham, H., *Software reliability*. Wiley encyclopedia of electrical and electronics engineering, 2001.
2. Singh, L.K., A.K. Tripathi, and G. Vinod, *Software reliability early prediction in architectural design phase: Overview and Limitations.* Journal of Software Engineering and Applications, 2011. **4**(03): p. 181.
3. Van den Brand, M. and J.F. Groote, *Software engineering: Redundancy is key.* Science of Computer Programming, 2015. **97**: p. 75-81.
4. Goel, A.L. and K. Okumoto, *Time-dependent error-detection rate model for software reliability and other performance measures.* IEEE transactions on Reliability, 1979. **28**(3): p. 206-211.
5. Kapur, P. and R. Garg, *A software reliability growth model for an error-removal phenomenon.* Software Engineering Journal, 1992. **7**(4): p. 291-294.
6. Aggarwal, A.G., N. Gandhi, V. Verma, and A. Tandon, *Multi-Release Software Reliability Growth Assessment: An approach incorporating Fault Reduction Factor and Imperfect Debugging*, in *International Journal of Mathematics in Operational Research*. 2019: Accepted for publication.
7. Chatterjee, S. and A. Shukla, *An ideal software release policy for an improved software reliability growth model incorporating imperfect debugging with fault removal efficiency and change point.* Asia-Pacific Journal of Operational Research, 2017. **34**(03): p. 1740017.
8. Anand, S., V. Verma, and A.G. Aggarwal, *2-Dimensional Multi-Release Software Reliability Modelling considering Fault Reduction Factor under imperfect debugging.* Ingeniería Solidaria ISSN online 2357-6014 ISSN print 1900-3102, 2018. **14**(25): p. 1-12.

9. Aggarwal, A.G., V. Dhaka, and N. Nijhawan, *Reliability analysis for multi-release open-source software systems with change point and exponentiated Weibull fault reduction factor*. Life Cycle Reliability and Safety Engineering, 2017. **6**(1): p. 3-14.
10. Zou, F.Z., *A change-point perspective on the software failure process.* Software testing, verification and reliability, 2003. **13**(2): p. 85-93.
11. Zhao, J. and J. Wang, *Testing the existence of change-point in NHPP software reliability models.* Communications in Statistics—Simulation and Computation®, 2007. **36**(3): p. 607-619.
12. Kopetz, H., *Software reliability*. 2016: Macmillan International Higher Education.
13. Huang, C.-Y. and S.-Y. Kuo, *Analysis of incorporating logistic testing-effort function into software reliability modeling.* IEEE Transactions on reliability, 2002. **51**(3): p. 261-270.
14. Huang, C.-Y. and M.R. Lyu, *Optimal testing resource allocation, and sensitivity analysis in software development.* IEEE Transactions on Reliability, 2005. **54**(4): p. 592-603.
15. Huang, C.-Y., *Performance analysis of software reliability growth models with testing-effort and change-point.* Journal of Systems and Software, 2005. **76**(2): p. 181-194.
16. Musa, J.D., *Iannino. A., Okumoto, K.: Software Reliability, Measurement, Prediction, Application.* McGraw-Hill, 1987. **61**: p. 46-51.
17. Musa, J.D., *A theory of software reliability and its application.* IEEE transactions on software engineering, 1975(3): p. 312-327.
18. Hsu, C.-J., C.-Y. Huang, and J.-R. Chang, *Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor*. Applied Mathematical Modelling, 2011. **35**(1): p. 506-521.
19. Pachauri, B., J. Dhar, and A. Kumar, *Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth.* Applied Mathematical Modelling, 2015. **39**(5-6): p. 1463-1469.
20. Friedman, M.A., P.Y. Tran, and P.I. Goddard, *Reliability of software intensive systems*. 1995: William Andrew.
21. Malaiya, Y.K., A. Von Mayrhauser, and P.K. Srimani, *An examination of fault exposure ratio.* IEEE Transactions on Software Engineering, 1993. **19**(11): p. 1087-1094.
22. Leung, Y.-W., *Optimum software release time with a given cost budget.* Journal of Systems and Software, 1992. **17**(3): p. 233-242.
23. Huang, C.-Y., S.-Y. Luo, and M.R. Lyu. *Optimal software release policy based on cost and reliability with testing efficiency*. in *Proceedings. Twenty-Third Annual International Computer Software and Applications Conference (Cat. No. 99CB37032)*. 1999. IEEE.
24. Inoue, S. and S. Yamada. *Optimal software release policy with change-point*. in *International Conference on Industrial Engineering and Engineering Management, IEEM.* . 2008. Singapore: IEEE.
25. Sachdeva, N., P. Kapur, and A. Shrivastava, *Software Release Time Problem Revisited.*, in *System Performance and Management Analytics. Asset Analytics (Performance and Safety Management)* 2019, Springer: Switzerland. p. 295-305.
26. Lewis, W.E., *Software testing and continuous quality improvement*. 2017: Auerbach publications.
27. Kapur, P., H. Pham, A. Gupta, and P. Jha, *Software reliability assessment with OR applications*. 2011: Springer.
28. Yamada, S., J. Hishitani, and S. Osaki, *Software-reliability growth with a Weibull test-effort: a model and application.* IEEE Transactions on Reliability, 1993. **42**(1): p. 100-106.
29. Huang, C.-Y., S.-Y. Kuo, and Y. Chen. *Analysis of a software reliability growth model with logistic testing-effort function*. in *Proceedings The Eighth International Symposium on Software Reliability Engineering*. 1997. IEEE.
30. Huang, C.-Y., S.-Y. Kuo, and M.R. Lyu, *An assessment of testing-effort dependent software reliability growth models.* IEEE transactions on Reliability, 2007. **56**(2): p. 198-211.
31. Bokhari, M. and N. Ahmad. *Analysis of a software reliability growth models: the case of log-logistic test-effort function*. in *Proceedings of the 17th IASTED international conference on modeling and simulation (MS'2006), Montreal, Canada*. 2006.
32. Gokhale, S.S. and K.S. Trivedi, *A time/structure based software reliability model.* Annals of Software Engineering, 1999. **8**(1-4): p. 85-121.
33. Quadri, S., N. Ahmad, M. Peer, and M. Kumar, *Nonhomogeneous Poisson process software reliability growth model with generalized exponential testing effort function.* RAU Journal of Research, 2006. **16**(1-2): p. 159-63.
34. Ahmad, N., M.G. Khan, and L.S. Rafi, *A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging.* International Journal of Quality & Reliability Management, 2010. **27**(1): p. 89-110.
35. Li, Q., H. Li, M. Lu, and X. Wang, *Software reliability growth model with S-shaped testing effort function*. Journal of Beijing University of Aeronautics and Astronautics, 2011. **37**(2).
36. Inoue, S. and S. Yamada, *Lognormal process software reliability modeling with testing-effort.* Journal of Software Engineering and Applications, 2013. **6**(04): p. 8.
37. Khatri, S.K., S.A. John, and R. Majumdar. *Quantifying software reliability using testing effort*. in *2016 International Conference on Information Technology (InCITe)-The Next Generation IT Summit on the Theme-Internet of Things: Connect your Worlds*. 2016. IEEE.
38. Bokhari, M., A. Siddiqui, and N. Ahmad, *Testing effort dependent delayed S-shaped software reliability growth model with imperfect debugging.* International Journal of Computer Science and Engineering, 2017. **9**(5): p. 138-148.
39. Gupta, A., S. Saxena, and N. Achrya, *Software reliability growth model involving log logistic test effort function and wiener process.* ARYABHATTA JOURNAL OF MATHEMATICS & INFORMATICS, 2018.
40. Yamada, S. and Y. Tamura, *Software Reliability*, in *OSS Reliability Measurement and Assessment*. 2016, Springer, : Switzerland.
41. Chen, T., S. Zheng, H. Liao, and J. Feng, *A Multi-attribute Reliability Allocation Method Considering Uncertain Preferences.* Quality and Reliability Engineering International, 2016. **32**(7): p. 2233-2244.
42. Song, K.Y., I.H. Chang, and H. Pham, *A Testing Coverage Model Based on NHPP Software Reliability Considering the Software Operating Environment and the Sensitivity Analysis.* Mathematics, 2019. **7**(5): p. 450.
43. Gandhi, N., H. Sharma, A.G. Aggarwal, and A. Tandon, *Reliability Growth Modeling for OSS: A Method Combining the Bass Model and Imperfect Debugging*, in *Smart Innovations in Communication and Computational Sciences*. 2019, Springer: Singapore. p. 23-34.
44. Li, X., M. Xie, and S.H. Ng, *Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points.* Applied Mathematical Modelling, 2010. **34**(11): p. 3560-3570.
45. Inoue, S. and S. Yamada, *Software reliability measurement with effect of change-point: Modeling and application.* International Journal of System Assurance Engineering and Management, 2011. **2**(2): p. 155-162.
46. Lin, C.-t. and C.-y. Huang. *Software Reliability Modeling withWeibull-type Testing-Effort and Multiple Change-Points*. in *TENCON 2005-2005 IEEE Region 10 Conference*. 2005. IEEE.
47. Musa, J.D., A. Iannino, and K. Okumoto, *Software reliability: Measurement, prediction, application. 1987*. 1987, McGraw-Hill.
48. Jain, M., T. Manjula, and T. Gulati. *Software reliability growth model (SRGM) with imperfect debugging, fault reduction factor and multiple change-point*. in *Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*. 2012. Springer.
49. Okumoto, K. and A.L. Goel, *Optimum release time for software systems based on reliability and cost criteria.* Journal of Systems and Software, 1979. **1**: p. 315-318.
50. Kapur, P., H. Pham, A.G. Aggarwal, and G. Kaur, *Two dimensional multi-release software reliability modeling and optimal release planning.* IEEE Transactions on Reliability, 2012. **61**(3): p. 758-768.
51. Kumar, V., P. Kapur, A. Shrivastava, and R. Sharma. *Optimal strategies for price-warranty decision model of software product with dynamic production cost*. in *3rd International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*. 2014. IEEE.
52. Lyu, M.R., *Handbook of software reliability engineering*. Vol. 222. 1996: IEEE computer society press CA.
53. Pham, H., *System software reliability*. 2007: Springer Verlag London.
54. Anand, A., M. Agarwal, Y. Tamura, and S. Yamada, *Economic impact of software patching and optimal release scheduling.* Quality and Reliability Engineering International, 2017. **33**(1): p. 149-157.
55. Xie, M. *On the determination of optimum software release time*. in *Proceedings. 1991 International Symposium on Software Reliability Engineering*. 1991. IEEE.
56. Ehrlich, W., B. Prasanna, J. Stampfel, and J. Wu, *Determining the cost of a stop-test decision (software reliability).* IEEE Software, 1993. **10**(2): p. 33-42.

57. Pham, H., *A software cost model with imperfect debugging, random life cycle and penalty cost.* International Journal of Systems Science, 1996. **27**(5): p. 455-463.
58. Hou, R.-H., S.-Y. Kuo, and Y.-P. Chang, *Optimal release times for software systems with scheduled delivery time based on the HGDM.* IEEE Transactions on Computers, 1997. **46**(2): p. 216-221.
59. Kapur, P., A.G. Aggarwal, K. Kapoor, and G. Kaur, *Optimal testing resource allocation for modular software considering cost, testing effort and reliability using genetic algorithm.* International Journal of Reliability, Quality and Safety Engineering, 2009. **16**(06): p. 495-508.
60. Huang, C.-Y. and M.R. Lyu, *Optimal release time for software systems considering cost, testing-effort, and test efficiency.* IEEE transactions on Reliability, 2005. **54**(4): p. 583-591.
61. Aggarwal, A.G., P. Kapur, G. Kaur, and R. Kumar, *Genetic algorithm based optimal testing effort allocation problem for modular software.* Bharati Vidyapeeth's Institute of Computer Applications and Management (BVICAM), 2012: p. 445.
62. Wood, A., *Predicting software reliability.* Computer, 1996. **29**(11): p. 69-77.
63. Ohba, M., *Software reliability analysis models.* IBM Journal of research and Development, 1984. **28**(4): p. 428-443.

AUTHORS PROFILE

Rajat Arora is a Ph.D. scholar in the Department of Operational Research, University of Delhi, India. She obtained her M.Sc. degree in Operational Research from the University of Delhi. She has completed her bachelors in mathematics. She has written few research papers which have been published in some International Journals and Conference Proceedings. Her research area is Software Reliability

Abhishek Tandon Dr. Abhishek Tandon received his Ph.D degree in Software Reliability and Marketing from Department of Operational Research, University of Delhi. He is currently working as a Deputy Director in Indian Council of Social Sciences Research, India. He has published papers in the field of Reliability Modeling, Optimization Theory, Forecasting and Marketing Research. He is a life member of the Society for Reliability Engineering, Quality and Operations Management (SREQOM).

Anu G. Aggarwal is working as Professor in the Department of Operational Research, University of Delhi. She obtained her Ph.D., M.Phil. and M.Sc. degrees in Operational Research from the University of Delhi in year 2007, 1999 and 1996, respectively. She has published several papers in the area of Marketing Management and Theory of Reliability. Her research interests include modelling and optimization in Consumer Buying Behaviour, Innovation-Diffusion modelling and Soft Computing Techniques.

Vibha Verma is a Ph.D. scholar in the Department of Operational Research, University of Delhi, India since 28 September 2017. She obtained her M.Sc. and M.Phil. degree in Operational Research from the University of Delhi in the year 2015 and 2017 respectively. She has completed her bachelors in mathematics. She has written few research papers which have been published in some International Journals and Conference Proceedings. Her research area is Software Reliabilitv.