Yugoslav Journal of Operations Research
31 (2021), Number 2, 165–180
DOI: https://doi.org/10.2298/YJOR200218001K

OPTIMAL DECISIONS ON SOFTWARE RELEASE AND POST-RELEASE TESTING: A UNIFIED APPROACH

Vivek KUMAR
Department of Operational Research, Faculty of Mathematical Sciences, New Academic Block, University of Delhi, India
vivekrajput.du.aor@gmail.com

Saurabh PANWAR
Department of Operational Research, Faculty of Mathematical Sciences, New Academic Block, University of Delhi, India
saurabhpanwar89@yahoo.com

P. K. KAPUR
Amity Center for Interdisciplinary Research, Amity University, Noida, India
pkkapur1@gmail.com

Ompal SINGH
Department of Operational Research, Faculty of Mathematical Sciences, New Academic Block, University of Delhi, India
drompalsingh1@gmail.com

Received: May 2020 / Accepted: October 2020

Abstract: In this research, a novel approach is developed where a testing team delivers the software product first and extends the testing process for additional time in the user environment. During the operational phase, users also participate in the fault detection process and notify the defects to the software. In this study, a reliability growth model is proposed using a unified approach based on the expenditure of efforts during the testing process. Besides, debugging process is considered imperfect as new faults may enter the software during each fault removal. The developed model further considers that the developer's rate of defect identification changes with a software release. Thus, the software time-to-market acts as a change-point for the failure observation phenomenon. It is asserted that the accuracy of a software reliability estimation improves by implementing

the concept of change-point. The main aim of the paper is to evaluate the optimal release time and testing termination time based on two attributes, particularly, reliability, and cost. A multi-attribute utility theory (MAUT) is applied to find a trade-off between the two conflicting attributes. Finally, a numerical example is presented by using the historical fault count data. The behavior of two decision variables is measured and compared with the existing release time strategy.

Keywords: Change-point, Field-testing Phase, Imperfect Debugging, Multi-attribute Utility Theory, Optimal Software Scheduling, Software Reliability Assessment, Testing Effort, Testing Duration.

MSC: 90B25, 90C90, 68N30.

1. INTRODUCTION

Reliability of software systems has become the most important customer-centric feature of software quality. Therefore, it is of great concern for software vendors to develop effectual and reliable software products. Rigorous testing is required to debug the underlying faults of the software products before releasing them for commercial purposes [15]. Based on the fault count data retrieved from the testing phase, the software reliability can be anticipated using suitable Software Reliability Growth Models (SRGMs) [27]. To assess the reliability of software products and to determine the impact of the testing process, numerous SRGMs have been developed by academicians and analysts in the last 40 years. In general, reliability growth models depend on the assumptions of Non-homogeneous Poisson Process (NHPP). In NHPP based models, the amount of defects identified, isolated, and removed from the software at any time instance is measured using mean value function [11]. Different SRGMs have been proposed based on different suppositions. The software reliability problem was first investigated by Goel and Okumoto [3], who proposed an analytical model with an assumption that a failure observation in the software system follows an exponential curve with the perfect debugging process. Later on, different SRGMs were developed on the assumption of perfect debugging. For instance, mathematical models provided by authors [34, 23, 8, 17, 22, 13] assumed that all the discovered faults would be corrected without causing any new fault generation. Nevertheless, this presumption is impractical in real-world scenarios. Therefore, a great number of studies have been suggested to quantify the reliability of a software system by assimilating different phenomena such as testing efforts, imperfect debugging, error generation, and change-point [28, 4, 35, 9, 31, 18, 12].

NHPP models have also been thoroughly applied in the cost-control investigation, examination of software time-to-market, and resource allocation problems [15]. In software reliability literature, the evaluation of software release and testing termination time has been extensively studied and a variety of analytical models and release time policies have been suggested [32]. The persistent testing of the software for a longer duration may hinder the timely delivery of the software system. In addition, it will swiftly lead to high development costs. At the same

time, the shorter testing with an inadequate debugging process will result in the buyer's disappointment that may adversely affect the growth of software products and goodwill of the software firm. Thus, the software testing duration and release time decisions are intricate issues, wherein it is necessary to keep balance between reliability, development cost, and time-to-market.

Consequently, over many past decades, the study on optimal release policy has earned enormous attention from the researchers. Okumoto and Goel [25] developed a basic optimal release policy that minimizes the total software development cost. Yamada and Osaki [33], soon after, proposed the constrained cost minimization problem to determine the optimal software release policy. Several other studies on software time-to-market have been carried out in which testing stop time and release time coincides [7, 5, 10, 20, 19, 21, 2]. There has been another stream of research, which demonstrates that the testing duration and software release time should be treated as two distinct decision variables [1, 6, 13, 14, 30]. Based on these studies, software developers are advised to make the software available for commercial use early and carry on the testing process in the operational phase for some time. This is recommended because the companies could reduce the revenue loss that would have otherwise occur due to the market being captured by the competitors. Additionally, prolonged testing will support software engineers to achieve the desired level of software reliability.

The proposed study considers the software release time as change point for developers defect identification rate. It has been observed that the failure occurrence of the software is often different in the field environment as compared to the testing environment. Moreover, software vendors attempt to debug the defects as early as possible to avoid software breakdown at the user end. Because of this, the developers change the testing strategies to modify the debugging process thereby, altering the fault detection rate. Thus, the release time is treated as a change-point (CP). The time instance where defect identification rate alters due to factors such as a change in the testing environment, resource allocation, testing strategy and integration testing is known as change-point [26].

1.1. Summary of Contribution

The key contributions of the present study in the field of software release management are:

- The present study develops a plausible software reliability model by incorporating different testing strategies before and after the software release.
- The paper proposes a new joint optimization policy for software time-to-market and testing stop time by simultaneously incorporating the testing effort, imperfect debugging, and change-point in the failure observation process.
- The proposed methodology imparts a comprehensive analysis of software release time based on cost and reliability attributes.

- The relevancy of the proposed research in real-world scenarios is numerically illustrated by using the historical failure count data. The findings yield that the new software release time policy gives better utility function than the conventional release time approach.

1.2. Problem Definition

This paper examines the optimal date for software release under the influence of testing effort, change-point, and imperfect debugging. A constrained optimization problem is framed to determine the optimal release time strategy. The failure occurrence behavior is represented using the software reliability growth model with an assumption that the release time acts as a change-point. The optimal decisions on software time-to-market and testing stop time are evaluated by using multi-attribute utility theory (MAUT). Specifically, two crucial decision variables, reliability, and cost function are chosen simultaneously to infer the optimal results. Besides, the proposed release time policy (PRT) is compared to the conventional release time policy (CRT). A case study is also conducted to numerically compare the proposed methodology with the existing testing and release time polices.

2. MODEL DEVELOPMENT

In this section, a generalized SRGM is developed using a unified approach. In the proposed FT release time policy, the fault detection process is modeled by segregating the software lifecycle in three phases. In the first phase, pre-release or in-house testing phase, the testers thoroughly detect and correct the faults using the efforts allocated for the testing process. It is worth noting that the present study assume that the debugging process is imperfect, i.e., there is a probability that detected faults may not be completely removed from the system. Moreover, there is a chance of introduction of new faults during the debugging process. In the second phase, post-release or field-testing phase, both software producers and customers identify the latent defects. However, the detected faults are fixed by the testers alone who then send patches to end-users for updating the software product. The developers terminate the testing process when the desired level of reliability is achieved. Therefore, in the third phase, post-testing phase, the task of fault detection is moved to the users. In the conventional release time (CRT) policy, the software lifecycle involves two phases: testing phase, and operational phase. The pictorial representations of these phases of the proposed and conventional release time policies are provided in Figure 1 and 2 respectively.

2.1. Assumptions

The proposed software growth model is based on the following assumptions.

1. Non-homogenous Poisson Process (NHPP) is applied to exemplify the debugging process.
2. There is no time lag between the fault detection and correction process.

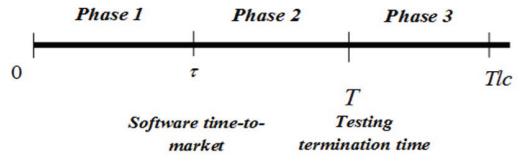

Figure 1: Three phases of fault observation phenomenon for proposed release time (PRT) policy

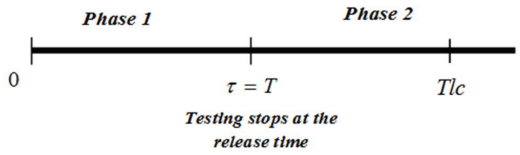

Figure 2: Two phases of fault observation phenomenon for conventional release time (CRT) policy

3. Each fault in the software system is independent and identically distributed throughout the software lifecycle.
4. The total number of faults lying dormant in the system is finite but may increase due to the imperfect debugging.
5. The introduction rate of faults is dependent on the latent faults and new faults are produced with the rate α_1
6. During the debugging process, there occur two possibility:
   - Fault is corrected completely with probability p_1
   - Fault is not removed with probability $(1–p_1)$ and the total fault content remains unchanged.
7. After the software release, both testers and customers meticulously discover failures in the system. Beyond the testing process, the identification of faults shifts completely to the users.
8. On failure observation, users instantly notify the software manufacturer who then debug the fault and send a patch to their customers for system update.
9. The expenditure on software patch is assumed negligible.

2.2. Generalized SRGM under proposed release time (PRT) policy

2.2.1. In-house testing or pre-release testing phase [0,τ)

Prior the release, software engineers ensure that the software goes through rigorous testing process to deliver a reliable software product to its customers. Based on the assumptions (1)-(6), the differential equation representing the failure

observation phenomenon incorporating imperfect debugging is given as:

$$\frac{dm_{1t}(W(t))}{dW(t)} = p_1 \frac{f_{1t}(W(t))}{1 - F_{1t}(W(t))} \left(a + \alpha_1 m_{1t}(W(t)) - m_{1t}(W(t))\right) \tag{1}$$

where $\frac{f_{1t}(t)}{1-F_{1t}(t)}$ describes the hazard rate function, p_1 is the probability of perfect debugging, α_1 is the rate of generation of new faults, and $W(t)$ is the cumulative amount of testing efforts consumed over time t. The analytical solution representing the expected number of defects detecting during pre-release testing phase is obtained by solving equation (1) under the condition that at $t = 0$, $W(t) = 0$, $m_{1t}(W(t)) = 0$, i.e.

$$m_{1t}(W(t)) = \frac{a}{1 - \alpha_1} \left[1 - (1 - F_{1t}(W(t)))^{p_1(1-\alpha_1)}\right] \tag{2}$$

Moreover, if the distribution function follows an exponential distribution, then the following solution is obtained:

$$m_{1t}(W(t)) = \frac{a}{1 - \alpha_1} \left(1 - e^{-b_1 p_1 (1-\alpha_1) W(t)}\right) \tag{3}$$

where b_1 is the fault detection rate parameter of exponential distribution function. Equation (3) expresses the number of defects identified by the testing team in the first phase.

2.2.2. Phase 2: Field-testing or post-release testing phase [τ, T)

During this phase, when softwrae system is in the field-environment, both developers and users attempt to identify defects from the software system. Nevertheless, testers are repsonsible for correcting the identified defects. It is assumed (assumption 7) that the a fixed proportion, λ of latent faults left after the in-house testing are identified by the tetsers while the $1 - \lambda$ proportion will be carefully identified by the customers. Now, the software is in the operating environment, the testing team alters their testing strategy to recify faults as soon as possible. Thus, in the present study, the release time is referred to as a change-point because the fault detection rate is modified at time-point τ.

Now, the differential equation representing defects identified by the software manufacturer at any instant in the seond phase is given as:

$$\begin{array}{c} \frac{dm_{2t}(W(t-\tau))}{dW(t)} = p_1 \frac{f_{2t}(W(t))}{1-F_{2t}(W(t))} \\ (\lambda a(1-F_{1t}(W(t)))^{p_1(1-\alpha_1)} + \alpha_2 m_{2t}(W(t-\tau)) - m_{2t}(W(t-\tau))) \end{array} \tag{4}$$

In equation (4), $\frac{f_{2t}(W(t))}{1-F_{2t}(W(t))}$ represents the hazard rate function after change-point τ and α_2 denotes the rate of error generation for this phase. On further solving the above equation, using the condition that at $t = \tau$, $m_{2t}(W(t-\tau)) = 0$, following closed-form solution is achieved:

$$m_{2t}(W(t-\tau)) = \lambda \frac{a}{(1-\alpha_2)} 1 - F_{1t}(W(\tau)^{p_1(1-\alpha_1)} [1 - (\frac{1-F_{2t}(W(t))}{1-F_{2t}(W(\tau))})^{p_1(1-\alpha_2)}] \tag{5}$$

Equation (5) represents the number of defects identified by the software manufacturer in the post-release testing phase. Furthermore, when exponential distribution is used to represent the failure observation phenomenon, then the analytical solution becomes:

$$m_{2t}(W(t-\tau)) = \lambda \frac{a}{(1-\alpha_2)} e^{-b_1 p_1 (1-\alpha_1) W(\tau)} 1 - e^{-b_2 p_1 (1-\alpha_2)(W(t)-W(\tau))}) \tag{6}$$

where b_2 is the rate parameter of the exponential distribution function after change-point τ.

Moreover, on executing instructions on the software, users may encounter failure because of the latent faults. Upon system breakdown, they may immediately notify to the developers. Consequently, the instantaneous defect identification by customers at any time t is described as:

$$\begin{array}{l} \frac{dm_{2u}(W(t-\tau))}{dW(t-\tau)} \\ ((1-\lambda)a(1-F_{1t}(W(t)))^{p_1(1-\alpha_1)} + \\ \alpha_2 m_{2u}(W(t-\tau)) - m_{2u}(W(t-\tau))) \end{array} = p_1 \frac{f_u(W(t-\tau))}{1-F_u(W(t-\tau))} \tag{7}$$

The closed-form solution of the above differential equation (7) can be obtained by using the initial condition that at $t = \tau$, $m_{2u}(W(t-\tau)) = 0$, i.e.

$$m_{2u}(W(t-\tau)) = (1-\lambda)\frac{a}{(1-\alpha_2)}(1 - F_{1t}(W(\tau))^{p_1(1-\alpha_1)} \left[1 - (1 - F_u(W(t-\tau)))^{p_1(1-\alpha_2)}\right] \tag{8}$$

Furthermore, when exponential distribution is used to represent the failure observation phenomenon, then the solution is expressed as:

$$m_{2u}(W(t-\tau)) = (1-\lambda)\frac{a}{(1-\alpha_2)} e^{-b_1 p_1 (1-\alpha_1) W(\tau)} \left(1 - e^{-b_3 p_1 (1-\alpha_2)(W(t-\tau))}\right) \tag{9}$$

where b_3 is the combined rate of users'defect identification of this testing phase.

2.2.3. Phase 3: Post-testing phase [T, T_{lc}]

After the termination of testing process at time T, the task of detecting the remaining faults shifts entirely to the users (assumption 8). Therefore, during this phase, users inform the developers about the defects identified, who upon the notification make an effort to correct it instantly. However, the fault detection rate of

users remains the same as that in the previous phase. Accordingly, the differential equation for the failure observation phenomena during this phase becomes:

$$\frac{dm_{3u}(W(t-T))}{dW(t-\tau)}=p_1\frac{f_u(W(t-\tau))}{1-F_u(W(t-\tau))}Z_a+\alpha_2 m_{3u}(W(t-T))-m_{3u}(W(t-T)) \tag{10}$$

where

$$Z_a=a(1-F_{1t}(W(\tau)))^{p_1(1-\alpha_1)}$$
$$1-\lambda\left[1-\left(\frac{1-F_{2t}(W(t))}{1-F_{2t}(W(\tau))}\right)^{p_1(1-\alpha_2)}\right]$$
$$-(1-\lambda)\left[1-(1-F_u(W(t-\tau)))^{p_1(1-\alpha_2)}\right]$$

Using the initial condition, $t=\tau$, $m_{2u}(W(t-T))=0$, above equation can be solved to obtain the following expression:

$$m_{3u}(W(t-T))=\frac{Z_a}{(1-\alpha_2)}\left[1-\left(\frac{1-F_u(W(t-\tau))}{1-F_u(W(T-\tau))}\right)^{p_1(1-\alpha_2)}\right] \tag{11}$$

Now, when exponential distribution is used to represent the failure observation phenomenon, then the analytical solution becomes:

$$\begin{aligned} m_{3u}(W(t-T))=\frac{a}{(1-\alpha_2)}&e^{-b_1p_1(1-\alpha_1)W(\tau)} \\ &\left(\begin{array}{l} 1-\lambda\left(1-e^{-b_2p_1(1-\alpha_2)(W(T)-W(\tau))}\right) \\ +(1-\lambda)\left(1-e^{-b_3p_1(1-\alpha_2)(W(T-\tau))}\right) \end{array}\right) \\ &\left(1-e^{-b_3p_1(1-\alpha_2)(W(t-T))}\right) \end{aligned} \tag{12}$$

3. GENERALIZED SRGM UNDER CONVENTIONAL RELEASE TIME (CRT) POLICY

In case of conventional release time policy followed by previous authors [15, 6], the testing duration coincides with the software time-to-market. Therefore, the software lifecycle consists of two phases: testing phase, and operational phase. In the first phase, only testers execute the failure observation process while in the operational phase, detection of the faults is the responsibility of users. Therefore, the expected number of defects observed by the testing team in period $[0,\tau)$ is given as:

$$m_{1t}(W(t))=\frac{a}{1-\alpha_1}\left[1-(1-F_{1t}(W(t)))^{p_1(1-\alpha_1)}\right] \tag{13}$$

Furthermore, in the second phase, $[\tau,T_{lc}]$ defects unidentified by the testers will be detected only by the users. Thus, the number of defects identified in operation phase is given as:

$$\begin{aligned} m_{2u}(W(t-\tau))=\frac{a}{(1-\alpha_2)}&(1-F_{1t}(W(\tau))^{p_1(1-\alpha_1)} \\ &\left[1-(1-F_u(W(t-\tau)))^{p_1(1-\alpha_2)}\right] \end{aligned} \tag{14}$$

4. MODELING TESTING-EFFORT FUNCTION

In the proposed modelling framework, the fault debugging process is governed by the testing resource consumption. For the present study, the consumption pattern of testing resources is considered to follow an exponential distribution function. The differential equation representing the consumption rate of testing resources during the testing process is give as:

$$\frac{dW(t)}{dt} = v\left(\bar{W} - W(t)\right) \tag{15}$$

where v denotes the rate with which testing-efforts are expended and $\bar{W}$ represents the total testing resources at hand for debugging process. On further solving the above equation using the boundary condition, $W(0) = 0$ following closed-form solution can be obtained:

$$W(t) = \bar{W}\left(1 - e^{-vt}\right) \tag{16}$$

Equation (16) represents the total expenditure of testing resources for fault debugging process over time t.

5. COST FUNCTION FOR TWO RELEASE POLICIES

In the present section, the major cost functions that have the most influence on the two crucial decision variables, software time-to-market and testing duration, are described.

a) *Testing cost*, $C_1W(T)$: the cost involved with testing activities such as planning, execution, test case generation, and analysis is known as testing cost. In software reliability literature, testing cost is considered directly dependent on the amount of efforts expended during the testing period.

b) *Market opportunity cost*, $C_2\tau^2$: the cost involved with the loss incurred by the firm due to the delay in the market entry of their software product. Market opportunity cost is an important factor that reflects the manipulation in the market by competitor firms. In past, it has been established that this cost is a quadratic function of the software release time.

c) *Cost of debugging during in-house testing phase*, $C_3m_{1t}(W(\tau))$: the cost linked to defect identification and correction during the in-house testing phase is referred to as debugging cost. The cost is directly dependent on the number of defects identified in this phase.

d) *Cost of debugging during field-testing phase*, $C_4m_{2t}(W(T-\tau)) + C_5m_{2u}(W(T-\tau))$: the cost due to fault correction post software release is the debugging cost in the field-testing phase. In the field environment, various unanticipated cost components such as liability costs, user disapproval cost, revenue losses, and indirect costs due to damaged reputation arise. All the components substantially increase the debugging cost per fault corrected post software release.

e) *Cost of debugging during post-testing phase*, $C_6 m_{3u}(W(T_{lc}-T))$: After the testing terminates at time T, only users identify the faults and immediately report the failure to the developers.

Under Proposed Release Time (PRT) Policy

The sum total of all the cost functions considering the field-testing framework is given as:

$$C(\tau,\ T) = C_1 W(T) + C_2 \tau^2 + C_3 m_{1t}(W(\tau)) + C_4 m_{2t}(W(T-\tau)) + C_5 m_{2u}(W(T-\tau)) + C_6 m_{3u}(W(T_{lc}-T)) \quad (17)$$

Under Conventional Release Time (CRT) Policy

Under traditional release time strategy, the cost function is structured as:

$$C(\tau) = C_1 W(\tau) + C_2 \tau^2 + C_3 m_{1t}(W(\tau)) + C_6 m_{2u}(W(T_{lc}-\tau)) \quad (18)$$

6. OPTIMAL RELEASE TIME DECISIONS USING MAUT

Multi-attribute utility theory is a well-known approach to solve the optimization problem involving multiple factors with contradictory objective functions and ensure the best solution [16]. In software reliability, the MAUT has been increasingly applied to evaluate the tradeoff between the conflicting attributes for analyzing the optimal software release policies [30]. In the present study, two critical attributes, cost and reliability functions, are identified for determining the optimal solution. MAUT describes tradeoff between these two attributes by modeling the utility function of each attribute. This approach comprises of following four steps:

6.1. Selection of suitable attributes

The release time strategy should be computed based on the most decisive attributes. These factors must be measurable and with a practical applicability. The foremost aim of software producers is to deliver both reliable and safe software system to their users. Correspondingly, the *reliability* is the necessary characteristic that affects optimal decisions involved with software time-to-market and testing duration. Thus, the first attribute included in the proposed optimization problem is:

$$R(x|t) = e^{-[m(W(t+x))-m(W(t))]} \quad (19)$$

where $R(x|0) = e^{-m(W(x))}$ and $R(x|\infty) = 1$.

Under proposed release time (PFT) policy:

$$Maximize \quad R(x|\tau,T) = e^{-[m(W(\tau+x_1))-m(W(\tau))]-[m(W(T+x_2))-m(W(T))]} \quad (20)$$

where x_1 and x_2 denotes the small time durations.

Under conventional release time (NFT) policy:

$$Maximize \quad R(x|\tau) = e^{-[m(W(\tau+x_1))-m(W(\tau))]} \tag{21}$$

The second attribute considered for the proposed optimization problem is the cost function. The investigation of cost budget is necessary for the software producers to develop reliable software at minimal cost. Therefore, the cost function for the proposed problem is given as:

Under proposed release time (PFT) policy:

$$Minimize \quad C(\tau,\ T) = \frac{C(W(\tau,\ T))}{C_b} \tag{22}$$

Under conventional release time (NFT) policy:

$$Minimize \quad C(\tau) = \frac{C(W(\tau))}{C_b} \tag{23}$$

6.2. Elicitation of SAUF for each attribute

Utility functions are applied to represent the goal of each attribute. Single attribute utility theory (SAUF) expresses the satisfaction level of management towards each attribute. For the proposed framework, the utility function of two attributes, namely, reliability and cost function, is:

$$u(C) = l_c + u_c C \text{ and } u(R) = l_r + m_r R \tag{24}$$

The boundary values for the utility function are: $u(y^{best}) = 1$ and $u(y^{worst}) = 0$ value. The bounds are calculated based on the management and decision makers' aspirations: a) For the reliability function, at least 60% of the defects should be identified and utmost 100% must be identified; b) For the cost function, the minimum budget prerequisite is 90% and the maximum requirement is 100%. Therefore, the bounds for these attributes are: $C^{worst} = 0.9$, $C^{best} = 1$, $R^{worst} = 0.6$ and $R^{best} = 1$. Under these boundary conditions, the SAUF for the two attributes takes the following functional form:

$$u(C) = 10C - 9 \text{ and } u(R) = 2.5R - 1.5 \tag{25}$$

6.3. Estimation of weight parameters

For the present study, the weight has been allotted on the managements' judgments. The value of the weight parameter lies between zero and one, where the value closer to 1 denotes the higher significance. Moreover, the sum of weight parameters should be equal to 1, i.e., $\omega_r + \omega_c = 1$. For the present problem, weight given by the software development management to the reliability attribute is $\omega_r = 0.6$, and consequently, weight assigned to cost attribute is $\omega_c = 0.4$.

6.4. Formulation of MAUF

Finally, the Multi-attribute Utility Function (MAUF) is developed by arithmetically summing all the single utility functions (SAUF), using the weight parameters. Therefore, the MAUF (u) for the proposed framework is expressed as:

$$Maximize \quad u(R,C) = \omega_r U(R) - \omega_C U(C) \tag{26}$$

where $\omega_r + \omega_c = 1$; *u(R)* and *u(C)* represent the single utility functions for reliability and cost element, respectively. In the present study, the focus of the software producers is to maximize the overall utility function. Therefore, the utility of the cost attribute is multiplied by negative sign to synchronize it with the reliability attribute and to obtain the maximum value of the MAUF. After substituting the values from previous steps, the MAUF function can be re-written as:

$$Maximize \quad u(R,C) = 0.6 \times (2.5R - 1.5) - 0.4 \times (10C - 9) \tag{27}$$

where $\omega_R + \omega_c = 1$ and $\frac{C(\tau,\, T)}{C_b} \leq 1$

The solution of the above-formulated utility function will yield the optimal values of software release time and testing stop time.

7. NUMERICAL EXAMPLE

In this section, the practical applicability of the proposed problem is illustrated through an example by using the historical fault discovery data. The data set used for the numerical illustration is the fault count data of Alcatel/Lucent Technologies, a French global telecommunications equipment company. The data is from the test of stability and was reported by Okumoto [24]. During the testing period of 56 weeks, 124 faults were identified with total testing effort consumption of 6316.5 CPU hours. Parameters of the model are estimated using the non-linear least regression. The estimated results of the model parameters for the pre-release testing period are: $\bar{W} = 76404.6$, $v = 0.001568$, $a = 167.32$, $b_1 = 0.000329$, $\alpha_1 = 0.06407$, and $p_1 = 0.7285$. The goodness-of-fit measures results are: $RMSE = 6.7618$ and $R_2 = 0.9722$. Moreover, the tester's rate in the second phase becomes $b_2 = 0.0004935$ (50% rise after the change-point) and the user's fault detection rate is taken as $b_3 = 0.0001974$ (60% of tester's detection rate in the pre-release phase). Also, it is considered the rate introduction of faults before and after remains the same, i.e., $\alpha_1 = \alpha_2$.. The remaining parameter values used for the optimization problem are:

$\bar{W} = 76404.6$, $x_1 = 2$, $x_2 = 2$, $C_1 = 20$, $C_2 = \$16$, $C_3 = \$60$, $C_4 = \$80$, $C_5 = \$140$, $C_6 = \$2,000$, $C_b = \$450,000$, $T_{lc} = 300$ weeks, $\lambda = 0.6$

The formulated MAUF problem is solved using the parameter values to obatin the optimal results. The compuatational software, MAPLE is utilized for solving

the maximization problem for both PRT (proposed release time policy) and CRT (existing release time policy). The optimal result for release time and testing stop time under two release time policy is given in Table 1. From the results, it can be clearly inferred that the release time policy with field testing yields better utility for the software vendors. Additionally, Figures 3 (a) and (b) provide the concavity plot of the utility functions. In Table 2, a phase-wise description of defects identified is listed. Out of 177 faults estimated to be present in the software, 73% of faults are successfully removed before testing termination at the 47^{th} week. The remaining faults are expected to be reported by the customers in the post-testing phase.

Release Policies	$U(\tau^*, T^*)$	τ^*(in weeks)	T^*(in weeks)
Field-testing (FT) policy	0.759	9.584	47.182
No field-testing (NFT) policy	0.727	48.362	-

Table 1: Optimal Results

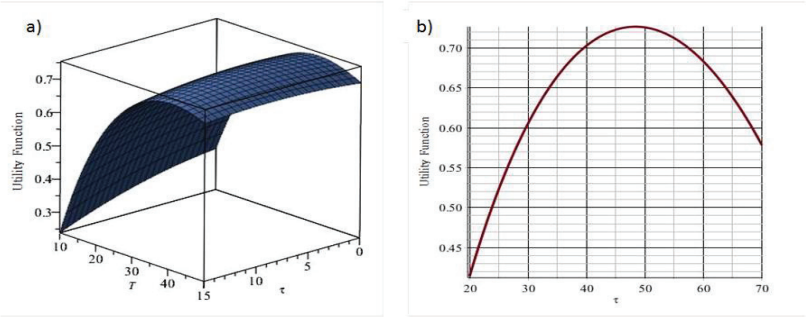

Figure 3: Utility function plot under a) PRT b) CRT policy

Software Lifecycle Phases	Mean Value Function	Number of detected faults (Approx.)
In-house testing phase	$m_{1t}(W(t))$	40 (by testers)
Field-testing phase	$m_{2t}(W(t-\tau))$	64 (by testers)
	$m_{2u}(W(t-\tau))$	25 (reported by users)
Post-testing phase	$m_{3u}(W(t-T))$	48 (reported by users)

Table 2: Stage wise description of failure observation under proposed release time policy

8. CONCLUDING REMARKS

In the present study, software release time strategy is evaluated by following a strategy of releasing the software early on and continuing the testing for an extra time in the user environment to avoid high market opportunity cost. To reflect on the current trends in the software industry, the present study discusses an imperfect debugging based SRGM for the software testing process, which is formulated as a continuous function of testing effort consumption. This paper also deals with the realistic situation of altering the detection rate of the testers after the software release time. The software performs differently in the field environment therefore, the company needs to adopt different testing strategies for the operational phase. To avoid software failure at the user environment and unforeseen costs components, testers intentionally intensify the testing process to debug the faults as soon as possible. The main aim of the proposed study is to assess the optimal time to release software, terminate testing process that maximizes the reliability function, and simultaneously minimize the cost components. The findings of the present study show the robustness and practicality of the proposed framework. The current research also offers new insights to project managers. The proposed release time policy assists project managers in making critical decisions under different realistic environments. By pursuing the policy of early software release, managers will be able to have a market share. Also, by protracted testing, the chance of system breakdown in the operational environment minimizes and will support developers to offer extremely reliable software, which enhances the clients' satisfaction. Furthermore, the developed study is worthy of future research. In the current paper, the optimal decisions are based on two critical factors, namely, reliability and cost. Therefore, in the consecutive research, other factors such as risk may also be considered in the optimization problem. Furthermore, the developed framework may be extended into a multi-release framework to examine the optimal frequency of software upgrades.

REFERENCES

[1] Arora, A., Caulkins, J. P., Telang, R., "Research note—Sell first, fix later: Impact of patching on software quality", *Management Science*, 52 (3) (2006) 465–471.

[2] Cao, P., Yang, K., and Liu, K., "Optimal selection and release problem in software testing process: a continuous time stochastic control approach", *European Journal of Operational Research*, 285 (1) (2019) 211-222.

[3] Goel, A. L., and Okumoto, K., "Time-dependent error-detection rate model for software reliability and other performance measures", *IEEE transactions on Reliability*, 28 (3) (1979) 206-211.

[4] Huang, C. Y., "Cost-reliability-optimal release policy for software reliability models incorporating improvements in testing efficiency", *Journal of Systems and Software*, 77 (2) (2005) 139-155.

[5] Huang, C. Y., and Lyu, M. R., "Optimal release time for software systems considering cost, testing-effort, and test efficiency", *IEEE transactions on Reliability*, 54 (4) (2005) 583-591.

[6] Jiang, Z., Sarkar, S., and Jacob, V. S., "Postrelease testing and software release policy for enterprise-level systems", *Information Systems Research*, 23 (3-part-1), (2012) 635-657.

[7] Kapur, P. K., and Garg, R. B., "Optimal release policies for software systems with testing effort", *International journal of systems science*, 22 (9) (1991) 1563-1571.
[8] Kapur, P. K., and Garg, R. B., "A software reliability growth model for an error-removal phenomenon", *Software Engineering Journal*, 7 (4) (1992) 291-294.
[9] Kapur, P. K., Goswami, D. N., Bardhan, A., and Singh, O., "Flexible software reliability growth model with testing effort dependent learning process", *Applied Mathematical Modelling*, 32 (7) (2008) 1298-1307.
[10] Kapur, P. K., Khatri, S. K., Tickoo, A., and Shatnawi, O., "Release time determination depending on number of test runs using multi attribute utility theory", *International Journal of System Assurance Engineering and Management*, 5 (2) (2014) 186-194.
[11] Kapur, P. K., Kumar, S., and Garg, R. B., *Contributions to hardware and software reliability*, (Vol. 3), World Scientific Publishing Company, Singapore (1999).
[12] Kapur, P. K., Panwar, S., Kumar, V., and Singh, O., "Entropy-Based Two-Dimensional Software Reliability Growth Modeling for Open-Source Software Incorporating Change-Point", *International Journal of Reliability, Quality and Safety Engineering*, 27 (05) (2020) 2040009.
[13] Kapur, P. K., Panwar, S., Singh, O., and Kumar, V., "Joint optimization of software time-to-market and testing duration using multi-attribute utility theory", *Annals of Operations Research*, (2019a).https://doi.org/10.1007/s10479-019-03483-w.
[14] Kapur, P. K., Panwar, S., Singh, O., and Kumar, V., "Joint Release and Testing Stop Time Policy with Testing-Effort and Change Poin", *In Risk Based Technologies*, Springer, Singapore (2019b) 209-222.
[15] Kapur, P. K., Pham, H., Gupta, A., and Jha, P. C., *Software reliability assessment with OR applications*, London, UK, Springer, 2011.
[16] Keeney, R. L., "Utility independence and preferences for multi attributed consequences", *Operations Research*, 19 (4) (1971) 875-893.
[17] Kumar, Vijay, Paridhi Mathur, Ramita Sahni, and Mohit Anand, "Two-dimensional multi-release software reliability modeling for fault detection and fault correction processes", *International Journal of Reliability, Quality and Safety Engineering*, 23 (03) (2016a) 1640002.
[18] Kumar, Vijay, Ramita Sahni, and Shrivastava, A. K., "Two-dimensional multi-release software modelling with testing effort, time and two types of imperfect debugging", *International Journal of Reliability and Safety*, 10 (4) (2016b) 368-388.
[19] Kumar, Vijay, V. B. Singh, Ashish Dhamija, and Shreyas Srivastav, "Cost-reliability-optimal release time of software with patching considered", *International Journal of Reliability, Quality and Safety Engineering*, 25 (04) (2018) 1850018.
[20] Li, Q., and Pham, H., "NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage", *Applied Mathematical Modelling*, 51 (2017) 68-85.
[21] Minamino, Y., Sakaguchi, S., Inoue, S., and Yamada, S., "Two-Dimensional NHPP Models Based on Several Testing-Time Functions and Their Applications", *International Journal of Reliability, Quality and Safety Engineering*, 26 (04) (2019) 1950018.
[22] Nagaraju, V., Fiondella, L., and Wandji, T., "A heterogeneous single changepoint software reliability growth model framework", *Software Testing, Verification and Reliability*, 29 (8) (2019) e1717.
[23] Ohba, M., and Yamada, S., "S-shaped software reliability growth models", in: *International Colloquium on Reliability and Maintainability, 4th, Tregastel, France*, (1984) (430-436).
[24] Okumoto, K., Customer-perceived software reliability: measurement, prediction, application, keynote talk at the 22nd *IEEE international symposium on software reliability engineering (ISSRE 2011)*, Tokyo, Japan 2011, downloadable from *http* : $//2011.issre.net/sites/default/-files/okumoto_data.pdf$.
[25] Okumoto, K., and Goel, A. L., "Optimum release time for software systems based on reliability and cost criteria", *Journal of Systems and Software*, 1 (4) (1980) 315-318.
[26] Panwar, S., Kapur, P. K., and Singh, O., "Modeling technological substitution by incorporating dynamic adoption rate", *International Journal of Innovation and Technology Management*, 16 (01) (2019) 1950010.
[27] Peng, R., Li, Y. F., Zhang, J. G., and Li, X., "A risk-reduction approach for optimal software

release time determination with the delay incurred cost", *International Journal of Systems Science*, 46 (9) (2015) 1628-1637.

[28] Pham, H., Nordmann, L., and Zhang, Z., "A general imperfect-software-debugging model with S-shaped fault-detection rate", *IEEE Transactions on reliability*, 48 (2) (1999) 169-175.

[29] SAS, S. STAT User guide, Version 9.1.2. *SAS Institute Inc, Cary, NC, USA*, 2004.

[30] Singh, O., Panwar, S., Kapur, P. K., "Determining software time-to-market and testing stop time when release time is a change-point", *International Journal of Mathematical, Engineering and Management Sciences*, 5 (2) (2020) 208-224.

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

[32] Yamada, S., *Software reliability modeling: fundamentals and applications*, (Vol. 5) Tokyo: Springer, 2014.

[33] Yamada, S., and Osaki, S., "Optimal software release policies with simultaneous cost and reliability requirements", *European Journal of Operational Research*, 31 (1) (1987) 46-51.

[34] Yamada, S., Ohba, M., and Osaki, S., "S-shaped reliability growth modeling for software error detection", *IEEE Transactions on reliability*, 32 (5) (1983) 475-484.

[35] Zhao, J., Liu, H. W., Cui, G., and Yang, X. Z., "Software reliability growth model with change-point and environmental function", *Journal of Systems and Software*, 79 (11) (2006) 1578-1587.

[36] Zhao, M., "Statistical reliability change-point estimation models", in: *Handbook of Reliability Engineering*, Springer, London, 2003, 157-163.