RT&A, Special Issue No 2(64),
Volume 16, November 2021

An Optimal Resource Allocation Model Considering Two-Phase Software Reliability Growth Model with Testing Effort and Imperfect Debugging

Sujit K. Pradhan[1], Anil Kumar[2], Vijay Kumar[3*]

[1,2]Department of Mathematics, BITS Pilani-KK Birla Goa Campus, Zuarinagar, Goa-403726, India.
E-mail: [1]pradhansujit1994@gmail.com, [2]anilpundir@goa.bits-pilani.ac.in
[3]Department of Mathematics, Amity Institute of Applied Sciences, Amity University
Uttar Pradesh, Noida-201313, India.
E-mail: [3]vijay_parashar@yahoo.com
*Corresponding Author

Abstract

This study aims at investigating an optimal resource plan in order to minimize the software costs in the debugging and testing phases. We have proposed a resource allocation model to address testing efforts, imperfect debugging and change-point. We have considered two cases: In the first case, the software debugging cost is kept constant, and in this case, the optimal policy follows a bang-bang structure, which means investing entirely in the testing phase, followed by investing fully in the debugging stage. In the second case, we have taken the debugging cost in quadratic form. We have validated our model with the experimental data, and the results reveal that the presented model is reasonably accurate. We have also discussed the optimal resource allocation problems under certain conditions and examine the parameters' behavior in the model and obtain the variations in the total cost. This study provides a detailed optimal control theory-based testing resource allocation policy, which is supported by numerical examples.

Keywords: Software reliability growth model (SRGM), Change point, Testing effort, Optimal control.

1. Introduction

Computers are used in diverse areas, and with recent advances in computation, software related issues have emerged as one of the primary areas of concern. Hence, reliable software product demand has increased in the market. Software reliability models can be effectively utilized to generate quantified measures during the software development phase. SRGM attempts to tally between defect detection data and estimated residual defects with time. Therefore, software reliability is crucial in developing software and its quality. Specification, design, programming, testing-and-debugging are the four stages involved in any software development process. During the software development process, the testing-and-debugging phase is a key and expensive phase of the software development life cycle (SDLC).

The probability when the software will not result in system failure for a specific time and under specific conditions is known as software reliability (see, e.g., [1]). Further, software reliability and software cost must be up to the expectation level of users satisfaction. During the testing phase of SDLC, testing and debugging are the two main activities to be performed by the testing team, and there is always a trade-off between cost and reliability. The software testing team has to understand the variance of the software reliability and the instantaneous testing costs. Thus, software reliability, cost, and release time are important aspects of software development.

Research activities on software reliability growth models have been conducted over the past five decades. In general, the SRGMs have been proposed under consideration with the non-homogeneous Poisson process (NHPP) since software faults are associated with discrete time scales. Various SRGMs [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17] have been proposed on different assumptions and applied to different situations. Many researchers have proposed SRGMs considering different scenarios; some researchers have proposed SRGM considering perfect debugging [2, 3, 5]; some researchers considered imperfect debugging [4, 6, 7]. Huang et al. [18] have discussed an SRGM, which incorporates debugging time-lag and fault dependency.

Many researchers proposed SRGMs incorporating testing efforts. Yamada et al. [19], and Musa et al. [20] proposed a novel SRGM. The model discussed in [19] describes the relationship between the amount of testing effort and the number of software errors detected. Kapur et al. [21], and Chang et al. [22] have proposed SRGMs, which incorporate the concept of testing-effort function. Generally, CPU hours are considered as testing efforts. Huang et al. [23], and Kapur et al. [24] represented a model in which the consumption rate of testing resource expenditures with the different testing-effort functions was discussed. Jin and Jin [25] used an S-shaped curve to describe software testing efforts. Huang et al. [26, 27] proposed SRGMs with logistic testing-effort function.

In software reliability studies, the initial number of faults contained are not known. One has to carry out the program in a particular environment. This helps to improve the quality of the program by correcting the faults. SRGMs proposed by Huang et al. [28] assumed that each failure caused by a fault is independent and random in time and occurs with the same distribution during the fault detection process. In SRGMs, the testing environment, testing strategy and resources are not necessarily the same throughout the development process. The testing environment is subjected to change with the learning process. Hence, the change-point problems play a vital role in software reliability growth modeling.

The change point is a point where the software testing team changes their testing strategy from one to another during the software development process due to the complexity of the program, testing facilities, and other random factors. The fault detection process is affected by different factors. The fault detection rate may change due to the increasing knowledge of the program and the testing strategies. Chang et al. [22] and Kapur et al. [29] discussed an SRGM with change point in fault detection rate. Shyur et al. [30] explained a stochastic software reliability growth model which incorporates imperfect debugging and change points.

Kapur et al. [31, 32] have discussed a model to assign the resources and minimize the total cost during the development period of SRGM under dynamic conditions. Kumar et al. [33, 34, 35] explained a resource allocation model for fault detection and fault correction process. They assumed detection and correction efforts to be independent. However, activities such as detection and correction may have budgetary constraints. Yonghua et al. [36, 37] proposed a model which incorporates a resource allocation plan to minimize the testing cost of the software. Kumar et al. [38] discussed resource allocation model for a multi-release SRGM to minimize the testing cost under dynamic conditions.

In the present work, we incorporate testing effort, imperfect debugging, and change points. These features are concomitant with the general SRGM. Yamada et al. [39] proposed an SRGM to minimize the total expenditure under static conditions. A problem emerges when the development process is carried out under dynamic conditions. The fault detection and fault correction process relies upon the operating environment and the quality of resources utilized. Experimental data analysis is carried out to evaluate the change point. We have used the failure increasing rate function to find the change point. We also study the optimization problem for optimal resource allocation for different conditions by examining the behavior of the model parameters and obtain variations in total cost. To do so, we have taken the quadratic form of the debugging function.

The remaining manuscript is structured as follows: We concluded Section 1 by providing notations used in the manuscript. In Section 2, we briefly discussed the model developed by Zhu et al. [40]. Section 3 deals with the model development, and we introduced an optimal control problem. In Section 4, the optimal policies are developed, and optimal solutions are given. Some

theoretical results are shown in Section 5, and the optimal policies are discussed for two special cases. In Section 6, the change point is calculated with the help of the data, and the behavior of model parameters in the variation of the total cost is discussed. In Section 7, we conclude the paper with some possible research on this topic.

Notations

$[0, T]$	The complete life cycle of the software.
t_1	The change-point.
$m_1(t)$	The cumulative number of faults detected in phase-I by time t lies between 0 to t_1.
$m_2(t)$	The cumulative number of faults detected in phase-II by time t lies between t_1 and T.
$x_1(t)$	The number of faults detected at any point of time t in phase-I.
$x_2(t)$	The number of faults detected at any point of time t in phase-II.
$a_1(t)$	The total fault content function in phase-I.
a	The initial number of fault.
$a_2(t)$	The total fault content function in phase-II.
α	The fault introduction rate per detected fault.
b_1	The fault detection rate in phase-I.
b_2	The fault detection rate in phase-II.
c_1	The non-removable fault rate in phase-I.
c_2	The non-removable fault rate in phase-II.
$w_{11}(t)$	The testing effort in phase-I.
$w_{21}(t)$	The testing effort in phase-II.
$w_{12}(t)$	The debugging effort in phase-I.
$w_{22}(t)$	The debugging effort in phase-II.
$c_{11}(t)$	The debugging cost per unit at time t associated with the debugging effort $w_{12}(t)$ in phase-I.
$c_{21}(t)$	The debugging cost per unit at time t associated with the debugging effort $w_{22}(t)$ in phase-II.
$\tilde{c}_{11}$	The base cost of debugging in phase-I.
$\tilde{c}_{21}$	The base cost of debugging in phase-II.
c_{12}	The cost of testing per unit testing effort at time t in phase-I.
c_{22}	The cost of testing per unit testing effort at time t in phase-II.
$y(t)$	The observed cumulative number of failures by time t.
$y'(t)$	The failure increasing rate during time interval $(t, t + \Delta t)$.

2. Two-phase software reliability growth model

An administer SRGM is tested by software test personnel to detect and correct software faults during the development process. In realistic conditions, different types of software faults are found in SRGMs. Software test personnel do not remove all faults during the debugging process by applying the same testing effort. So different testing efforts are applied by software test personnel to remove different types of faults. Thus, to remove two types of fault, software test personnel used a two-phase debugging process. Zhu et al. [40] proposed a two-phase software reliability model under the following assumptions, which involved software fault dependency and imperfect debugging.

1. The error detection in SRGMs follows the non-homogeneous Poisson process.
2. The model considered imperfect debugging, and new faults are introduced into the program at each time.
3. Type-I and type-II software faults are defined. The type-I and type-II faults are detected and corrected during phase-I and phase-II, respectively.
4. The software development team cannot remove all the faults experienced in both phases.

5. Due to different software fault types, the fault detection and non-removable fault rates are different in phase I and II.

6. The time to debug a fault is negligible.

Using the above assumptions, the following two-phase SRGM is considered for the study.

Phase-I. The type-I faults, which are independent and easily detected, are detected and corrected in phase-I. The total number of software faults that cause failure in phase-I is proportional to the difference between the total number of detected faults and the total number of non-removable faults. The total number of detected faults that cause failure is the product of fault detection rate and the number of remaining type-I faults which are represented by the following differential equation

$$\frac{dm_1(t)}{dt} = b_1(t)\,[a_1(t) - m_1(t)] - c_1(t)m_1(t), \qquad 0 \le t \le t_1. \tag{1}$$

With a constant fault introduction rate α (> 0), new faults are introduced during the debugging phase due to imperfect debugging. Therefore, the fault present in phase-I at time t is given by

$$a_1(t) = a(1 + \alpha t). \tag{2}$$

In phase-I, let the non-removable fault rate is

$$c_1(t) = c_1, \;\; c_1 > 0, \tag{3}$$

with the initial condition $m_1 = 0$ at $t = 0$ for phase-I.

Phase-II. During this phase, type-II faults are detected after the completion of type-I faults detection and correction. No new and residue type-I faults are introduced in phase-II. The total number of software faults that cause failure is proportional to the difference between the total number of detected faults and non-removable faults. The total number of detected faults that cause failure is the product of fault detection rate, the ratio of the total number of reduced faults to the total number of failures experienced, and the number of remaining type-II faults. Therefore, we have the following differential equation

$$\frac{dm_2(t)}{dt} = b_2(t)\frac{m_2(t)}{a_2(t)}\,[a_2(t) - m_2(t)] - c_2(t)m_2(t), \;\; t_1 \le t \le T. \tag{4}$$

The cumulative number of fault present in phase-II is

$$a_2(t) = a_1(t_1) - m_1(t_1). \tag{5}$$

In phase-II, let the non-removable fault rate is

$$c_2(t) = c_2, \;\; c_2 > 0, \tag{6}$$

with the continuity condition between the two phases

$$m_2(t_1) = m_1(t_1). \tag{7}$$

3. Model development

The testing and debugging phase aims to detect and correct faults and make the software more reliable during the development process of the software. Software reliability is affected by the testing resources spent in the testing phase. We have modified the SRGM given in Section-2 without increasing the complexity. We have developed a model, which incorporates time-dependent testing effort with change point under imperfect debugging environments. Here, a model is constructed with concurrent detection and correction activities. Resources should

be allocated optimally during software testing. Thus, a tactical plan is required for allocating resources optimally. The total resource in each phase is divided into two portions, i.e. testing effort and debugging effort. The mathematical expression for the resource allocation model is written as

$$w_{11}(t) + w_{12}(t) = 1, \quad 0 \leq t \leq t_1, \tag{8}$$
$$w_{21}(t) + w_{22}(t) = 1, \quad t_1 \leq t \leq T, \tag{9}$$

which is shown in Figure 1.

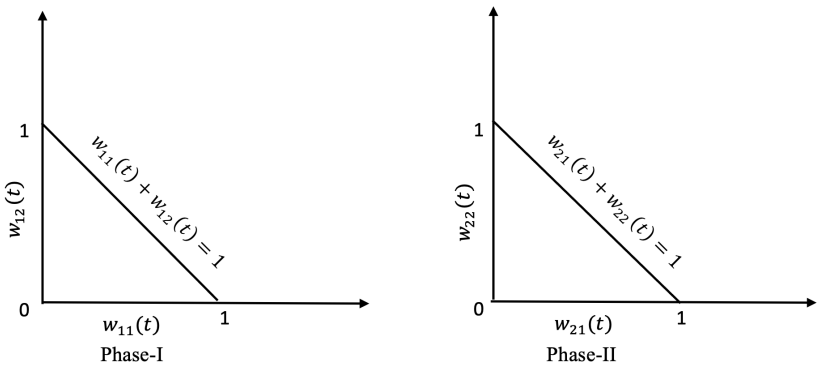

Figure 1: *Optimal resource allocation in Phase-I and Phase-II.*

Phase-I. We have assumed that the software development team detects the faults causing failure and corrects those faults during the testing phase to develop a new model. The total number of software faults that cause failure in phase-I per unit testing effort expenditure is proportional to the difference between the total number of detected faults and the total number of non-removable faults. The total number of detected faults that cause failure is the product of fault detection rate and the number of remaining type-I faults. All the detected faults are not possible to remove from software during the testing phase. Here, we have taken the testing effort in the fault detection process. We know that we should remove more faults if much testing effort is utilized in the testing phase. Some faults can't be removed. Based on the above assumptions and compatible with the idea of Zhu et al. [40], we have proposed the phase-I software reliability growth model as

$$x_1(t) = \frac{dm_1(t)}{dt} = w_{11}(t)\left[b_1(a_1(t) - m_1(t)) - c_1 m_1(t)\right], \qquad 0 \leq t \leq t_1. \tag{10}$$

Phase-II. In phase-II, the total number of software faults that cause failure per unit testing effort expenditure is proportional to the difference between the total number of detected faults and the total number of non-removable faults. The total number of detected faults that cause failure is the product of fault detection rate, the ratio of the total number of reduced faults to the total number of failures experienced and the number of remaining type-II faults. The following differential equation gives the phase-II model:

$$x_2(t) = \frac{dm_2(t)}{dt} = w_{21}(t)\left[b_2 \frac{m_2(t)}{a_2(t)}(a_2(t) - m_2(t)) - c_2 m_2(t)\right], \quad t_1 \leq t \leq T. \tag{11}$$

Optimal control problem. We aim to create a resource allocation plan to minimize the total testing expenditure. For the simplicity of the model, we have neglected all other costs except fault detection cost and correction cost over the finite planning period T. The proposed model incorporates testing effort as a control parameter. Then the problem can be described over the interval $[0, T]$ as follows:

$$\min\left[\int_0^{t_1}\Big\{c_{11}(t)x_1(t)+c_{12}w_{11}(t)\Big\}dt+\int_{t_1}^{T}\Big\{c_{21}(t)x_2(t)+c_{22}w_{21}(t)\Big\}dt\right], \tag{12}$$

subject to

$$x_1(t)=\frac{dm_1(t)}{dt} = w_{11}(t)\left[b_1(a_1(t)-m_1(t))-c_1m_1(t)\right], \qquad 0\le t\le t_1, \tag{13}$$

$$x_2(t)=\frac{dm_2(t)}{dt} = w_{21}(t)\left[b_2\frac{m_2(t)}{a_2(t)}(a_2(t)-m_2(t))-c_2m_2(t)\right], \quad t_1\le t\le T, \tag{14}$$

with the conditions $m_1(0)=0$, and $m_2(t_1)=m_1(t_1)$.

4. Optimal policies and solution

To solve dynamic optimal control problem defined by equations (12)–(14), we shall use Pontryagin minimum principle [41]. Therefore, we first define the Hamiltonian function, which is given by

$$H(t)=\begin{cases}H_1(t), & 0\le t\le t_1\\ H_2(t), & t_1\le t\le T\end{cases} \tag{15}$$

where

$$H_1(m_1(t),\lambda_1(t),w_{11}(t),t) = c_{11}(t)x_1(t)+c_{12}w_{11}(t)+\lambda_1(t)x_1(t), \ \ 0\le t\le t_1,$$
$$H_2(m_2(t),\lambda_2(t),w_{21}(t),t) = c_{21}(t)x_2(t)+c_{22}w_{21}(t)+\lambda_2(t)x_2(t), \ \ t_1\le t\le T.$$

The necessary conditions within each time interval for an optimal solution are defined similar to [42]. The co-state variables (or adjoint variables) $\lambda_1(t)$ and $\lambda_2(t)$ are given by

$$\frac{d}{dt}\lambda_1(t) = -\frac{\partial H_1(m_1(t),\lambda_1(t),w_{11}(t),t)}{\partial m_1(t)}, \quad 0\le t\le t_1, \tag{16}$$

$$\frac{d}{dt}\lambda_2(t) = -\frac{\partial H_2(m_2(t),\lambda_2(t),w_{21}(t),t)}{\partial m_2(t)}, \ \ t_1\le t\le T, \tag{17}$$

with terminal conditions $\lambda_2(T)=0$. Also, the following matching conditions are satisfied at $t=t_1$

$$\lambda_1(t_1) = \lambda_2(t_1), \tag{18}$$

$$H_1(m_1(t_1),\lambda_1(t_1),w_{11}(t_1),t_1) = H_2(m_2(t_1),\lambda_2(t_1),w_{21}(t_1),t_1), \tag{19}$$

and we have

$$H_1(m_1(t_1),\lambda_1(t_1),w_{11}(t_1),t_1) \le H_2(m_2(t_1),\lambda_2(t_1),w_{21}(t_1),t_1), \text{ if } 0=t_1<T, \tag{20}$$

$$H_1(m_1(t_1),\lambda_1(t_1),w_{11}(t_1),t_1) \ge H_2(m_2(t_1),\lambda_2(t_1),w_{21}(t_1),t_1), \text{ if } 0<t_1=T. \tag{21}$$

The control variables $w_{11}(t)$ and $w_{21}(t)$ are given by

$$\frac{\partial H_1(m_1(t),\lambda_1(t),w_{11}(t),t)}{\partial w_{11}(t)} = 0, \quad 0\le t\le t_1, \tag{22}$$

$$\frac{\partial H_2(m_2(t),\lambda_2(t),w_{21}(t),t)}{\partial w_{21}(t)} = 0, \quad t_1\le t\le T, \tag{23}$$

which will give from the equations (22) and (23)

$$w_{11}^*(t) = \frac{(c_1m_1(t)-b_1(a_1(t)-m_1(t)))(c_{11}(t)+\lambda_1(t))-c_{12}}{c_{11w_{11}}(t)b_1((a_1(t)-m_1(t))-c_1(t)m_1(t))}, \qquad 0\le t\le t_1, \tag{24}$$

$$w_{21}^*(t) = \frac{(c_2m_2(t)-b_2(\frac{m_2(t)}{a_2(t)})(a_2(t)-m_2(t)))(c_{21}(t)+\lambda_2(t))-c_{22}}{c_{21w_{21}}(t)(b_2(\frac{m_2(t)}{a_2(t)})(a_2(t)-m_2(t))-c_2m_2(t))}, \ \ t_1\le t\le T, \tag{25}$$

and from equations (8) and (9), we obtain

$$w_{12}^*(t) = 1 - \frac{(c_1 m_1(t) - b_1(a_1(t) - m_1(t)))(c_{11}(t) + \lambda_1(t)) - c_{12}}{c_{11w_{11}}(t) b_1((a_1(t) - m_1(t)) - c_1(t) m_1(t))}, \quad 0 \le t \le t_1, \quad (26)$$

$$w_{22}^*(t) = 1 - \frac{(c_2 m_2(t) - b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t)))(c_{21}(t) + \lambda_2(t)) - c_{22}}{c_{21w_{21}}(t)(b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t)) - c_2 m_2(t))}, \quad t_1 \le t \le T. \quad (27)$$

5. Theoretical results

In this section, we have discussed optimal criteria for two-phase, continuous-time optimal control problems where the integration of these problems depends upon the change point. The qualitative results developed here are demonstrated in the following theorem.

Theorem 1. Let the change point t_1 occurs in the software development life cycle $[0, T]$. Then the following holds

(i) At $t_1 \in (0, T)$, the total cost of fault detection and correction in phase-I is equal to the total cost of fault detection and correction in phase-II.

(ii) At $t_1 = 0$, the total cost of fault detection and correction in phase-I is less than the total cost of fault detection and correction in phase-II.

(iii) At $t_1 = T$, the total cost of fault detection and correction in phase-I is greater than the total cost of fault detection and correction in phase-II.

Proof. Let $[0, T]$ is the planning period of SRGM and the change point $t_1 \in (0, T)$. Since, the Hamiltonian H_1 and H_2 are equal at the change point t_1 which is given by equation (19)

$$H_1(m_1(t_1), \lambda_1(t_1), w_{11}(t_1), t_1) = H_2(m_2(t_1), \lambda_2(t_1), w_{21}(t_1), t_1).$$

We obtain

$$c_{11}(t)x_1(t) + c_{12}w_{11}(t) + \lambda_1(t)x_1(t) = c_{21}(t)x_2(t) + c_{22}w_{21}(t) + \lambda_2(t)x_2(t),$$

and hence, we get

$$(c_{11}(t) + \lambda_1(t))x_1(t) + c_{12}w_{11}(t) = (c_{21}(t) + \lambda_2(t))x_2(t) + c_{22}w_{21}(t). \quad (28)$$

Therefore, in the equation (28), the total cost of fault detection and correction in phase-I is equal to the total cost of fault detection and correction in phase-II.

To prove (ii), let the change-point be $t_1 = 0$, i.e. $t_1 \notin (0, T)$. Then we should skip directly to phase-II for fault detection and correction i.e. in phase-I, no-fault detection and correction are done. Hence, the total cost of fault detection and correction in phase-I is less than the total cost of fault detection and correction in phase-II.

Let the change-point be $t_1 = T$ i.e $t_1 \notin (0, T)$. Then we stick entirely in phase-I for fault detection and correction, i.e. in phase-II, no-fault detection and correction is done. Hence, the total cost of fault detection and correction in phase-I is greater than the total cost of fault detection and correction in phase-II. This complete the proof of part (iii). ■

Special cases. In general, at the initial time of the testing phase of SRGM, the debugging/fault correction cost is much higher as the uncertain nature of the errors. Later on, the fault intensity gradually decreases, and most of the expenditure is invested in the testing phase. Therefore the debugging cost gradually decreases with time. Here, we shall discuss two special cases to show how debugging/fault correction costs impact the optimal policies.

Case-I. Let the debugging/fault correction cost per unit for cumulative fault removed at time t before and after change point are constant, i.e.

$$c_{11}(t) = c_{11} \text{ and } c_{21}(t) = c_{21} \tag{29}$$

Then for constant debugging cost, the objective function (12) will take the following form

$$\min \left[\int_0^{t_1} \left\{ c_{11}x_1(t) + c_{12}w_{11}(t) \right\} dt + \int_{t_1}^{T} \left\{ c_{21}x_2(t) + c_{22}w_{21}(t) \right\} dt \right]$$

subject to

$$x_1(t) = \frac{dm_1(t)}{dt} = w_{11}(t)\left[b_1(a_1(t) - m_1(t)) - c_1 m_1(t)\right], \quad 0 \le t \le t_1 \tag{30}$$

$$x_2(t) = \frac{dm_2(t)}{dt} = w_{21}(t)\left[b_2 \frac{m_2(t)}{a_2(t)}(a_2(t) - m_2(t)) - c_2 m_2(t)\right], \quad t_1 \le t \le T \tag{31}$$

with the conditions $m_1(0) = 0, m_2(t_1) = m_1(t_1), w_{11}(t) + w_{12}(t) = 1$ and $w_{21}(t) + w_{22}(t) = 1$. Then the Hamiltonian for the optimal control problem is

$$\begin{aligned} H_1(m_1(t), \lambda_1(t), w_{11}(t), t) &= c_{11}x_1(t) + c_{12}w_{11}(t) + \lambda_1(t)x_1(t), \ 0 \le t \le t_1 \\ H_2(m_2(t), \lambda_2(t), w_{21}(t), t) &= c_{21}x_2(t) + c_{22}w_{21}(t) + \lambda_2(t)x_2(t), \ t_1 \le t \le T, \end{aligned}$$

The adjoint variable $\lambda_1(t)$ and $\lambda_2(t)$ are given by

$$\frac{d}{dt}\lambda_1(t) = \dot{\lambda}_1(t) = w_{11}(t)(c_{11} + \lambda_1(t))(b_1 + c_1), \quad 0 \le t \le t_1 \tag{32}$$

$$\frac{d}{dt}\lambda_2(t) = \dot{\lambda}_2(t) = w_{21}(t)\{c_{21}(t) + \lambda_2(t)\}\{c_2 - b_2\left(1 - \frac{2m_2(t)}{a_2(t)}\right)\}, \quad t_1 \le t \le T \tag{33}$$

with the terminal condition $\lambda_2(T) = 0$, which on simplifying will give

$$\lambda_1(t_1) = \lambda_1(0) + \int_o^{t_1} w_{11}(t)(c_{11} + \lambda_1(t))(b_1 + c_1)dt \tag{34}$$

$$\lambda_2(t_1) = \int_{t_1}^{T} w_{21}(t)\{c_{21} + \lambda_2(t)\}\{b_2\left(1 - \frac{2m_2(t)}{a_2(t)}\right) - c_2\}dt. \tag{35}$$

The necessary conditions for optimality are

$$\frac{\partial H_1}{\partial w_{11}} = 0, \quad 0 \le t \le t_1 \tag{36}$$

$$\frac{\partial H_2}{\partial w_{21}} = 0, \quad t_1 \le t \le T \tag{37}$$

which will give

$$H_{1w_{11}} = (c_{11} + \lambda_1(t))(b_1(a_1(t) - m_1(t)) - c_1 m_1(t)) + c_{12}, \ 0 \le t \le t_1,$$

and

$$H_{2w_{21}} = (c_{21} + \lambda_2(t))(b_2 \frac{m_2(t)}{a_2(t)}(a_2(t) - m_2(t)) - c_2 m_2(t)) + c_{22}, \ t_1 \le t \le T.$$

where $H_{1w_{11}} = \frac{\partial H_1}{\partial w_{11}}$ and $H_{2w_{21}} = \frac{\partial H_2}{\partial w_{21}}$.

The Hamiltonian in equation (15) is linear in $w_{11}(t)$ and $w_{21}(t)$, respectively. So we have the following bang-bang solution for $w_{11}(t)$ and $w_{21}(t)$ to minimize the Hamiltonian. Therefore, we obtain $w_{11}(t)$ and $w_{21}(t)$ as

(i) If $0 \leq t \leq t_1$,

$$w_{11}^*(t) = \begin{cases} 1, & \text{if } H_{1w_{11}} > 0 \\ \text{undefined}, & \text{if } H_{1w_{11}} = 0 \\ 0, & \text{if } H_{1w_{11}} < 0. \end{cases} \tag{38}$$

If the marginal value of the testing effort $H_{1w_{11}}$ is positive, then maximum possible testing effort is $w_{11}(t) = 1$ because Hamiltonian H_1 is linear in control variable $w_{11}(t)$. If $H_{1w_{11}}$ is negative, then minimum testing effort is $w_{11}(t) = 0$ i.e. all effort should be allocated to debugging. If $H_{1w_{11}} = 0$, then the testing effort $w_{11}(t)$ need not be determined from the condition $H_{1w_{11}} = 0$.

(ii) If $t_1 \leq t \leq T$,

$$w_{21}^*(t) = \begin{cases} 1, & \text{if } H_{2w_{21}} > 0 \\ \text{undefined}, & \text{if } H_{2w_{21}} = 0 \\ 0, & \text{if } H_{2w_{21}} < 0. \end{cases} \tag{39}$$

Similarly, if the marginal value of the testing effort $H_{2w_{21}}$ is positive, then maximum possible testing effort is $w_{21}(t) = 1$ because Hamiltonian H_2 is linear in control variable $w_{21}(t)$. If $H_{2w_{21}}$ is negative, then minimum testing effort is $w_{21}(t) = 0$ i.e. all effort should be allocated to debugging. If $H_{2w_{21}} = 0$, then the testing effort $w_{21}(t)$ need not be determined from the condition $H_{2w_{21}} = 0$.

Case-II. Researchers have been proposed the effect of the experience curve in SRGM. Kapur et al. [31] used Pegels' form [43] as debugging cost function. In this case, we consider that the total cost per unit fault removed is a quadratic function of the debugging efforts, i.e.

$$c_{11}(t) = \tilde{c}_{11}(w_{12}(t))^2 \text{ and } c_{21}(t) = \tilde{c}_{21}(w_{22}(t))^2. \tag{40}$$

Then the Hamiltonian (15) will reduce to

$$\begin{aligned} H_1(m_1(t), \lambda_1(t), w_{11}(t), t) &= c_{11}(t)x_1(t) + c_{12}w_{11}(t) + \lambda_1(t)x_1(t), \ 0 \leq t \leq t_1 \\ H_2(m_2(t), \lambda_2(t), w_{21}(t), t) &= c_{21}(t)x_2(t) + c_{22}w_{21}(t) + \lambda_2(t)x_2(t), \ t_1 \leq t \leq T. \end{aligned}$$

From the necessary condition of optimality described by (22) and (23) , we obtain

$$w_{12}(t) = \sqrt{\frac{\lambda_1(t)(c_1 m_1(t) - b_1(a_1(t) - m_1(t))) - c_{12}}{\tilde{c}_{11}(b_1(a_1(t) - m_1(t)) - c_1 m_1(t))}}, \ 0 \leq t \leq t_1, \tag{41}$$

$$w_{22}(t) = \sqrt{\frac{\lambda_2(t)(c_2 m_2(t) - b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t))) - c_{22}}{\tilde{c}_{21}(b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t)) - c_2 m_2(t))}}, \ t_1 \leq t \leq T. \tag{42}$$

Therefore, the optimal policy for $w_{11}(t)$ and $w_{21}(t)$ using equations (8) and (9), can be expressed as

$$w_{11}(t) = 1 - \sqrt{\frac{\lambda_1(t)(c_1 m_1(t) - b_1(a_1(t) - m_1(t))) - c_{12}}{\tilde{c}_{11}(b_1(a_1(t) - m_1(t)) - c_1 m_1(t))}}, \ 0 \leq t \leq t_1, \tag{43}$$

$$w_{21}(t) = 1 - \sqrt{\frac{\lambda_2(t)(c_2 m_2(t) - b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t))) - c_{22}}{\tilde{c}_{21}(b_2(\frac{m_2(t)}{a_2(t)})(a_2(t) - m_2(t)) - c_2 m_2(t))}}, \ t_1 \leq t \leq T. \tag{44}$$

6. Numerical Analysis

In this section, different optimal policies have been discussed on the proposed model using a numerical example. This study aims to get some view into the result and study the impact of change in efforts on the model's total cost.

Our first objective is to find the change point. We have taken PL/I database application software data, given in Table 1, which was studied by Ohba [13] to get the change point of the model. Specifically, testing for the PL/I database application software ranges from week 1 to week 19, and a total of 328 errors are found. The discussed model in section-3 comprises two phases. Generally, when the testing is carried out, the software developers team knows the change point t_1. But here, we don't know the value of t_1. Therefore, we first need to determine the value of t_1 in the model validation.

Weeks	Cumulative execution time (CPU hours)	Cumulative number of detected faults	Weeks	Cumulative execution time (CPU hours)	Cumulative number of detected faults
1	2.45	15	11	26.23	233
2	4.9	44	12	27.67	255
3	6.86	66	13	30.93	276
4	7.84	103	14	34.77	298
5	9.52	105	15	38.61	304
6	12.89	110	16	40.91	311
7	17.1	146	17	42.67	320
8	20.47	175	18	44.66	325
9	21.43	179	19	47.65	328
10	23.35	206			

Table 1: *Data set (Failure data from PL/I database application [13])*

To determine the value of t_1, we shall use the software failure increasing rate concept which is given by

$$y'(t) = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}. \tag{45}$$

For different value of Δt, the pattern of $y'(t)$ is shown in Figure 2. We know that $y'(t)$ increases until it reached its optimum due to the tester's growing fault correction experience. Afterwards, it starts to decline to stabilize the rate. A similar trend will be showing again for fault correction of another type of faults. From Figure 2, we conclude that the optimum value for failure increasing rate in phase-I is obtained at $t = 6$ and phase-II at $t = 9$. Therefore, we get the change point at $t_1 = 7$.

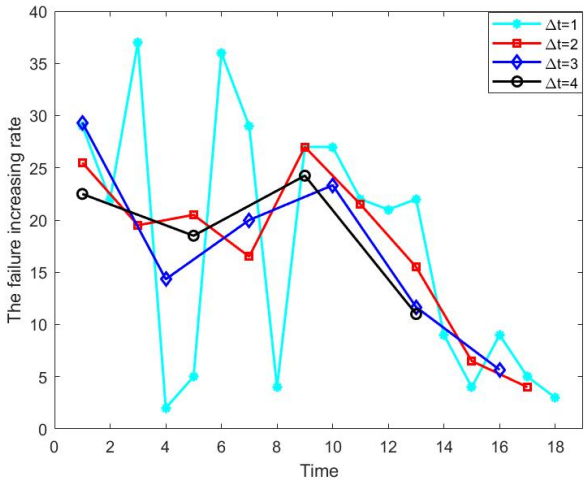

Figure 2: *Software failure rate with respect to time t.*

From the above analysis, we have concluded that the change point t_1 occurs at 7. For $t_1 = 7$, the software failure in phase-I and phase-II is defined when $t \in [0, 7]$ and $t \in [7, 19]$ respectively.

We discuss briefly the comparative analysis of the proposed SRGM with different testing efforts. The common criteria used to compare the model are the mean-squared error (MSE), the predictive-ratio risk (PRR) and the predictive power (PP), and the results are shown in Table 2.

The MSE measures the deviation between the predicted values with the actual data and is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(m(t_i) - y_i)^2,$$

where n is the number of observations in the model. The PRR measures the distance of model estimates from the actual data against the model estimate which is given by

$$PRR = \sum_{i=1}^{n}\left(\frac{m(t_i) - y_i}{m(t_i)}\right)^2,$$

whereas the PP measures the distance of model estimates from the actual data against the actual data and is defined as

$$PP = \sum_{i=1}^{n}\left(\frac{m(t_i) - y_i}{y_i}\right)^2.$$

	$w_{11}(t) = 0.5$, $w_{21}(t) = 0.6$	$w_{11}(t) = 0.6$, $w_{21}(t) = 0.7$	$w_{11}(t) = 0.7$, $w_{21}(t) = 0.8$	$w_{11}(t) = 0.8$, $w_{21}(t) = 0.9$
MSE	758.15	428.54	292.29	46.57
PRR	0.5001	0.3408	0.5849	0.1982
PP	0.4588	0.3881	0.3934	0.4087

Table 2: *Model parameters comparison criteria*

We have estimated different model parameters using Excel 2019. The estimated parameters (a, α, b_1, c_1, b_2 and c_2) and the hypothetical value of the other parameters are presented in Table 3.

Parameter	$w_{11}(t) = 0.5$, $w_{21}(t) = 0.6$	$w_{11}(t) = 0.6$, $w_{21}(t) = 0.7$	$w_{11}(t) = 0.7$, $w_{21}(t) = 0.8$	$w_{11}(t) = 0.8$, $w_{21}(t) = 0.9$
a	380	440	450	454
α	0.02	0.016	0.018	0.015
b_1	0.109	0.085	0.075	0.072
c_1	0.0573	0.0496	0.046	0.032
b_2	0.4889	0.4718	0.465	0.3354
c_2	0.0135	0.0644	0.0644	0.0185
$\tilde{c}_{11}$	500	950	1950	2200
$\tilde{c}_{21}$	1100	3800	15000	220000
c_{12}	8500	21000	25000	30000
c_{22}	5000	12000	13000	15000

Table 3: *Value of parameters for different $w_{11}(t)$ and $w_{21}(t)$.*

In this analysis, we have discussed the significance of allocation of testing efforts $w_{11}(t)$ and $w_{21}(t)$. Figure 3 describe the relationship between the cumulative number of faults removed versus time. It shows that the increase in the testing efforts $w_{11}(t)$ and $w_{21}(t)$ will increase the number of faults removed. It has been seen that when the value of $w_{11}(t)$ and $w_{21}(t)$ gradually increases, the rate of fault removal increases.

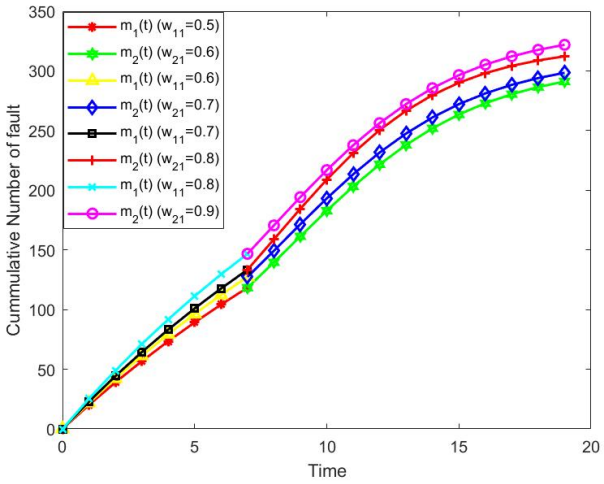

Figure 3: *Cumulative number of faults removed vs time*

The analysis is also done to show how the debugging cost impacts the future cost to remove one fault and also indicates that decreasing the debugging efforts $w_{12}(t)$ and $w_{22}(t)$ will lead to a depletion in the total debugging cost. The pattern is shown in Figure 4, which says that the co-state variable decreases with time and approaches zero.

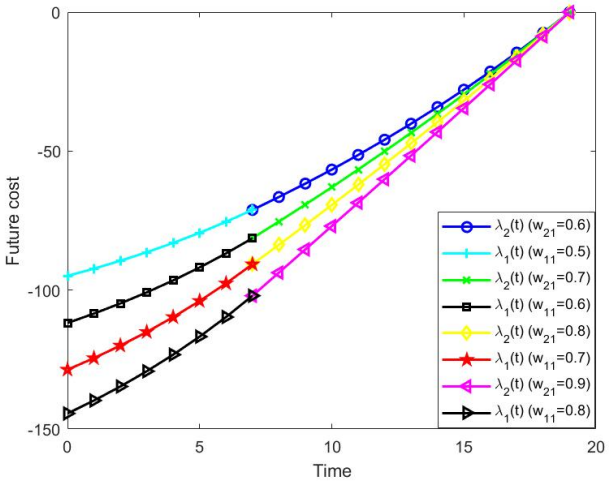

Figure 4: *Shadow cost vs time*

In the analysis, a reduction in the total debugging cost is observed when the debugging effort decreases and hence the rate of increase of the total expenditure is reduced. Also, it shows how the Hamiltonian (H) that is the sum of testing and debugging cost, starts decreasing after some time. In short, Hamiltonian represents the instantaneous total cost of the model at time t. The results are shown in Table 4, and the total developmental cost of the SRGM with different testing efforts are 93469, 247329, 357569 and 734065, respectively.

Time (Weeks)	Total Cost ($w_{11}(t) = 0.5$, $w_{21}(t) = 0.6$)	Total Cost ($w_{11}(t) = 0.6$, $w_{21}(t) = 0.7$)	Total Cost ($w_{11}(t) = 0.7$, $w_{21}(t) = 0.8$)	Total Cost ($w_{11}(t) = 0.8$, $w_{21}(t) = 0.9$)
1	4886	13516	18628	25407
2	4900	13532	18651	25413
3	4915	13550	18675	25418
4	4932	13569	18702	25425
5	4950	13590	18731	25431
6	4970	13612	18763	25437
7	4991	13637	18798	25444
8	5171	14009	23142	62883
9	5371	14329	23702	63850
10	5488	14419	23594	62797
11	5507	14270	22849	59895
12	5429	13910	21611	55575
13	5270	13393	20086	50399
14	5054	12786	18477	44925
15	4806	12150	16942	39606
16	4552	11536	15577	34744
17	4307	10974	14422	30501
18	4083	10482	13482	26925
19	3887	10065	12737	23990

Table 4: *Total Cost (Testing cost and debugging cost)*

7. Conclusions and future work

In this article, we have presented a resource allocation technique considering a two-phase software reliability growth model with testing effort, change-point, and imperfect debugging. Numerical simulations are done which support the accuracy of our proposed model. We have given an insight into how to find the change point. Generally, the software developers team knows the change point of real software testing. This paper aims to calculate the total cost of the software using two-phase SRGM. We have proposed the theoretical results using optimal control theory, and a different approach is used to allocate testing resources optimally. We can control the testing efforts when the company switches from one testing strategy to another. We observed from the graph of the future cost of detection that it eventually reaches zero with time. Moreover, from the graph of shadow cost of correction, we observed that, as time increases, the shadow cost decreases and tends to zero. The variation in cost with the change in various model parameters has also been depicted.

In contrast with the other studies, we proposed the theoretical results to find the change point from one testing strategy to another using optimal control theory. But in this paper, experimental data analysis is utilized to determine the change point. We have used the failure increasing rate function to find the change point. To propose a more realistic SRGM, more information is needed by software managers. In this direction, the stochastic model for fault detection and correction can be used for future work. We can also extend the proposed model by incorporating different fault content functions. All these issues may be part of further work.

References

[1] Musa, J. D. (1980). The measurement and management of software reliability. *Proceedings of the IEEE*, 68(9), pp. 1131-1143.

[2] Goel, A. L., & Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE transactions on Reliability*, 28(3), pp. 206-211.
[3] Ohba, M., & Yamada, S. (1984). S-shaped software reliability growth models. In *International Colloquium on Reliability and Maintainability*, 4 th, Tregastel, France, pp. 430-436.
[4] Yamada, S., Tokuno, K., & Osaki, S. (1992). Imperfect debugging models with fault introduction rate for software reliability assessment. *International Journal of Systems Science*, 23(12), pp. 2241-2252.
[5] Ohba, M. (1984). Software reliability analysis models. *IBM Journal of research and Development*, 28(4), pp. 428-443.
[6] Li, Q., & Pham, H. (2017). NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage. *Applied Mathematical Modelling*, 51, pp. 68-85.
[7] Pham, H. (1996). A software cost model with imperfect debugging, random life cycle and penalty cost. *International Journal of Systems Science*, 27(5), pp. 455-463.
[8] Yamada, S., & Osaki, S. (1985). Software reliability growth modeling: Models and applications. *IEEE Transactions on Software Engineering*, (12), pp. 1431-1437.
[9] Kareer, N., Kapur, P. K., & Grover, P. S. (1990). An S-shaped software reliability growth model with two types of errors. *Microelectronics Reliability*, 30(6), pp. 1085-1090.
[10] Pham, H. (2016). A generalized fault-detection software reliability model subject to random operating environments. *Vietnam Journal of Computer Science*, 3(3), pp. 145-150.
[11] Kapur, P. K., Pham, H., Anand, S., & Yadav, K. (2011). A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. *IEEE Transactions on Reliability*, 60(1), pp. 331-340.
[12] Kapur, P. K., Pham, H., Aggarwal, A. G.,& Kaur, G. (2012). Two dimensional multi-release software reliability modeling and optimal release planning. *IEEE Transactions on Reliability*, 61(3), pp. 758-768.
[13] Lo, J. H., & Huang, C. Y. (2006). An integration of fault detection and correction processes in software reliability analysis. *Journal of Systems and Software*, 79(9), pp. 1312-1323.
[14] Pham, H., & Zhang, X. (1997). An NHPP software reliability model and its comparison. International Journal of Reliability, *Quality and Safety Engineering*, 4(3), pp. 269-282.
[15] Pham, H. (2007). An imperfect-debugging fault-detection dependent-parameter software. *International Journal of Automation and Computing*, 4(4), pp. 325.
[16] Kumar, V., Singh, V. B., Dhamija, A., & Srivastav, S. (2018). Cost-reliability-optimal release time of software with patching considered. *International Journal of Reliability, Quality and Safety Engineering*, 25(04), pp. 1850018.
[17] Kumar, V., Sahni, R., & Shrivastava, A. K. (2016). Two-dimensional multi-release software modelling with testing effort, time and two types of imperfect debugging. *International Journal of Reliability and Safety*, 10(4), pp. 368-388.
[18] Huang, C. Y., & Lin, C. T. (2006). Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Transactions on reliability*, 55(3), pp. 436-450.
[19] Yamada, S., Ohtera, H., & Narihisa, H. (1986). Software reliability growth models with testing-effort. *IEEE Transactions on Reliability*, 35(1), pp. 19-23.
[20] Musa, J. D., Iannino, A., & Okumoto, K. (1990). Software reliability. *Advances in computers*, 30, pp. 85-170.
[21] Kapur, P. K., Goswami, D. N., Bardhan, A., & Singh, O. (2008). Flexible software reliability growth model with testing effort dependent learning process. *Applied Mathematical Modelling*, 32(7), pp. 1298-1307.
[22] Chang, Y. P. (2001). Estimation of parameters for nonhomogeneous Poisson process: Software reliability with change-point model. *Communications in Statistics-Simulation and Computation*, 30(3), pp. 623-635.
[23] Huang, C. Y., & Kuo, S. Y. (2002). Analysis of incorporating logistic testing-effort function into software reliability modeling. *IEEE Transactions on reliability*, 51(3), pp. 261-270.

[24] Kapur, P. K., Goswami, D. N., & Gupta, A. (2004). A software reliability growth model with testing effort dependent learning function for distributed systems. *International Journal of Reliability, Quality and Safety Engineering*, 11(04), pp. 365-377.

[25] Jin, C., & Jin, S. W. (2016). Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization. *Applied Soft Computing*, 40, pp. 283-291.

[26] Huang, C. Y., Lyu, M. R., & Kuo, S. Y. (2003). A unified scheme of some nonhomogenous poisson process models for software reliability estimation. *IEEE transactions on Software Engineering*, 29(3), pp. 261-269.

[27] Huang, C. Y., Kuo, S. Y., & Lyu, M. R. (2007). An assessment of testing-effort dependent software reliability growth models. *IEEE transactions on Reliability*, 56(2), pp. 198-211.

[28] Huang, C. Y., Lin, C. T., Kuo, S. Y., Lyu, M. R., & Sue, C. C. (2004, September). Software reliability growth models incorporating fault dependency with various debugging time lags. *In Proceedings of the 28th Annual International Computer Software and Applications Conference*, COMPSAC 2004, IEEE, pp. 186-191.

[29] Kapur, P. K., Gupta, A., Shatnawi, O., & Yadavalli, V. S. S. (2006). Testing effort control using flexible software reliability growth model with change point. *International Journal of Performability Engineering*, 2(3), pp. 245.

[30] Shyur, H. J. (2003). A stochastic software reliability model with imperfect-debugging and change-point. *Journal of Systems and Software*, 66(2), pp. 135-141.

[31] Kapur, P. K., Pham, H., Chanda, U., & Kumar, V. (2013). Optimal allocation of testing effort during testing and debugging phases: a control theoretic approach. *International Journal of Systems Science*, 44(9), pp. 1639-1650.

[32] Kapur, P. K., Pham, H., Kumar, V., & Anand, A. (2012). Dynamic optimal control model for profit maximization of software product under the influence of promotional effort. *The Journal of High Technology Management Research*, 23(2), pp. 122-129.

[33] Kumar, V., & Sahni, R. (2016). An effort allocation model considering different budgetary constraint on fault detection process and fault correction process. *Decision Science Letters*, 5(1), pp. 143-156.

[34] Kumar, V., Kapur, P. K., Taneja, N., & Sahni, R. (2017). On allocation of resources during testing phase incorporating flexible software reliability growth model with testing effort under dynamic environment. *International Journal of Operational Research*, 30(4), pp. 523-539.

[35] Kumar, V., Khatri, S. K., Dua, H., Sharma, M., & Mathur, P. (2014). An assessment of testing cost with effort-dependent fdp and fcp under learning effect: a genetic algorithm approach. *International Journal of Reliability, Quality and Safety Engineering*, 21(06), pp. 1450027.

[36] Ji, Y., Mookerjee, V. S., & Sethi, S. P. (2005). Optimal software development: A control theoretic approach. *Information Systems Research*, 16(3), pp. 292-306.

[37] Ji, Y., Kumar, S., Mookerjee, V. S., Sethi, S. P., & Yeh, D. (2011). Optimal enhancement and lifetime of software systems: A control theoretic analysis. *Production and Operations Management*, 20(6), pp. 889-904.

[38] Kumar, V., & Sahni, R. (2020). Dynamic testing resource allocation modeling for multi-release software using optimal control theory and genetic algorithm. *International Journal of Quality & Reliability Management*, 37(6/7), pp. 1049-1069.

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

[40] Zhu, M., & Pham, H. (2018). A two-phase software reliability modeling involving with software fault dependency and imperfect fault removal. *Computer Languages, Systems & Structures*, 53, pp. 27-42.

[41] Naidu, D. S. (2002). *Optimal control systems*. CRC press.

[42] Kamien, M. I., & Schwartz, N. L. (2012). *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Courier Corporation.

[43] Pegels, C. C. (1969). On startup or learning curves: An expanded view. *AIIE Transactions*, 1(3), pp. 216-222.