Unveiling the Evolution: Multi-Patch Multi-Release Software Reliability Growth Model with Testing Effort

Veenu Singh[1], Vijay Kumar[2,*], V. B. Singh[3]
and Arun Prakash Agrawal[4]

[1]*Amity Institute of Information Technology, Amity University, Noida, Uttar Pradesh, India*
[2]*Department of Mathematics, AIAS, Amity University, Noida, Uttar Pradesh, India*
[3]*School of Computer and Systems Sciences, Jawaharlal Nehru University, India*
[4]*School of Computer Science Engineering and Technology, Bennett University Greater Noida, India*
E-mail: veens1824@gmail.com; vijay_parashar@yahoo.com; vbsingh@mail.jnu.ac.in; arunpragrawal@gmail.com
[*]*Corresponding Author*

Received 29 May 2024; Accepted 01 December 2024

Abstract

Reliance on software has increased expectations for software organizations to deliver high-quality software to meet the increasing demand from end-users. Continuous testing is imperative to ensure software quality, yet prolonged testing can lead to increased market opportunity costs. Consequently, organizations often opt to release software early and subsequently conduct testing during the operational phase, addressing existing bugs through patch deployment. These patches, small programs aimed at fixing, improving, or updating software, serve to rectify security vulnerabilities or bugs efficiently. For minor changes, patch releases prove more practical and cost-effective than launching entirely new software versions. The adoption of multi-release software endows developers with a competitive advantage, catering to the diverse needs of end-users. This paper introduces a testing effort-based software reliability growth model, evaluating the impact of multi-patching on multi-release software. The model operates under the assumption of continuous fault removal post-release, using different distribution functions to construct three framework variations. Parameter estimation employs the Statistical Package for Social Sciences, with a real dataset serving as a basis for a numerical example illustrating the model's practical application. Additionally, a comparative analysis of model performance, based on different distribution functions, is conducted through multi-criteria decision-makingtechniques.

Keywords: Software reliability growth model, multi-release, patch, testing cost, multi-criteria decision making.

1 Introduction

Software is a set of instructions, or we can say a set of commands that tells a computer what task needs to be performed. Technology is all around us and is very important in our daily lives. Due to the innovative steps in the field of technology, our regular tasks can now be done easily and effectively. Software is what we think of modern technology. However, people and businesses worldwide are using software for various purposes, reaping numerous benefits. Software development can enhance end user's experiences, introduce more feature-rich and innovative products to the market, and enhance setup processes to be safer, more productive, and more efficient (Guesmi et al. 2023; David et al. 2023). Software development represents the entire lifespan of the software and is known as the software development life cycle (SDLC). SDLC includes requirement analysis, design, implementation, testing, and maintenance phases. Testing is one of the crucial phases during the software development process. During this phase, faults are detected and removed. Debugging all the faults is not practically possible but it is possible to use the testing results in order to reduce further faults and failures in the software.

Software Reliability Growth Model (SRGM) is a mathematical model that is used by software engineers in order to predict software reliability. It also helps in managing faults during the software development process. It helps in understanding how the fault count in a software system changes when the software undergoes the testing and debugging process. SRGM is a very essential tool for software engineers as it provides a quantitative way to estimate software development progress in terms of fault removal. Different types of SRGM exist, based on different assumptions and mathematical formulations. These models can be tailored to fit into different software development scenarios, resulting in more accurate reliability prediction. SRGM plays a crucial role in ensuring that software meets the expected level of quality before it is released into the market. Some of the most commonly used SRGMs (Goel and Okumoto, 1979; Yamada et al., 1983; Ohba, 1984; Musa and Okumoto 1984; Yamada et al., 1993; Pham and Zhang, 1997; Huang et al., 1997).

In the context of software development, a “release” can be defined as the specific version of the software that is available to the end users. Each release includes fault fixes, improvements, and new features over the previous releases. By using multi-release models’ software engineers gain valuable insights regarding the reliability of different releases, which helps them in making decisions about resource allocation, release planning, quality assurance, etc. Bibyan et al. (2023) have proposed a multi-release testing coverage based on SRGM. Aggarwal et al. (2019) have proposed a multi-release software showcasing the effect of imperfect debugging and time variable fault reduction factor. Saraf and Iqbal (2019) have developed a non-homogeneous Poisson Process (NHPP) based multi-release two-stage model where fault detection/observation and fault correction/removal are considered.

To identify the relation between detected faults and faults removed during testing, various testing effort based SRGM have been proposed under a different set of assumptions (Huang and Lyu 2005; Inoue and Yamada 2013; Li et al., 2015; Saxena et al., 2021; Samal et al., 2023; Singh et al., 2024; Pradhan et al., 2024). Huang and Lyu (2005) have introduced a testing effort-based function known as generalized logistic and further incorporated this function into the software reliability growth model. Additionally, the author has also explored the effect of new testing techniques on enhancing the effectiveness of software testing. Inoue and Yamada (2013) have put forth testing effort based on SRGM and rely on continuous state space stochastic process. Furthermore, the authorhas conducteda goodness of fit assessment. Li et al.(2015) have incorporated testing effort-based functions with imperfect debugging. Saxena et al. (2021) have proposed testing effort based SRGM within a fuzzy environment. Furthermore, the author has taken into account the impact of imperfect debugging. Samal et al. (2023) have incorporated generalized logistic testing effort function and change point into the SRGM.

Introducing error-free software is an unattainable task for the testing team. However, to ensure that end users face the least number of faults during the operational phase, software organizations now persist in addressing faults even after software has been released. During this post-release phase, fault removal, feature enhancement, etc. is done through patches. Kapur et al. (2018) have proposed a framework in order to determine the optimal release time, patching time, and testing stop time of the software to minimize the overall software cost. Tickoo et al. (2016) have introduced a testing effort-based cost model to determine the optimal release and patching time of software. Saxena et al. (2024) have modeled software release time and patch release time. Anand et al. (2017) have proposed a scheduling policy for software and presented the impact of patching in lowering the overall software cost. Kansal et al. (2016) have developed a framework to determine the optimal release and patching time of the software product under warranty. Choudhary et al. (2019) have illustrated the testing effort-based cost model by taking into account the effect of change point and warranty. The authors focused on determining the optimal time for software release and patching to minimize the overall cost of the software.

Multi Criteria Decision Making (MCDM) techniques are used to handle decision-making problems that aim to determine the best alternative by considering more than one criterion during the selection process (Gupta et al., 2018; Garg 2019; Devet al., 2020; Saxena et al., 2022; Bibyan and Anand 2022; Rawat et al. 2022; Kumar et al. 2022; Singh et al., 2023; Sharma et al. 2023; Chaube 2024; Pant et al. 2024). Various MCDM techniques have been developed and used to date. Gupta et al. (2018) have done the ranking of SRGM using MCDM approaches. Garg (2019) have conducted literature-based approach to demonstrate how Multi-Criteria Decision Analysis (MCDA), specifically using the Analytical Hierarchy Process (AHP) and TOPSIS models, has been applied to address complex decision-making problems. Dev et al. (2020) have proposed methods for conducting material selection using MCDM approaches.

The literature discussed in the current section provides the foundation for this study and highlights the need to address the existing research gap. While considerable work has been conducted on multi-release, patching, and testing effort based SRGM, there is limited exploration of multi-patch multi-release models. In this research, a multi-release software reliability growth model (SRGM) is proposed, specifically focused on testing effort allocation and its impact across multiple software patches. We have also presented three frameworks featuring distinct distribution functions. These frameworks offer flexibility in accounting for various real-world distributions of software faults and their occurrences. Performance of proposed Frameworks represented

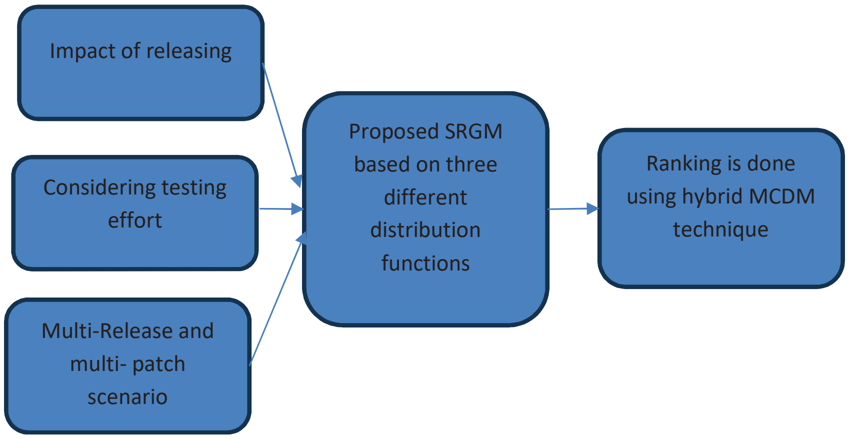

Impact of releasing
Considering testing effort
Multi-Release and multi- patch scenario
Proposed SRGM based on three different distribution functions
Ranking is done using hybrid MCDM technique

Figure 1 Proposed framework.

using goodness of fit measures. In order to get a clear image of the best-suited framework for a particular release MCDM approach is used. It is noticed that the technique VIKOR and entropy is not used in ranking testing effort based SRGM. The VIKOR (VlseKriterijumskaOptimizacija I KompromisnoResenje) is an effective ranking technique as it has simple mathematical calculations and therefore is more convenient and easier to use (Dev et al., 2020). We have used entropy for calculating the weights of model parameters. Therefore, by using these two techniques ranking of proposed frameworks is done. The proposed framework is shown in Figure 1. The key contributions of the study are.

- Proposed testing effort based multi-release model showcasing the impact of multi-patch in each release once the software is introduced to the market.
- Proposed three models using different distribution functions.
- Identifying the optimal timing for patch releases.
- Model ranking to identify the best fit model for each release is done using the ENTROPY-VIKOR technique's.

The structure of the paper is as follows: Section 2 outlines the assumptions and notations used throughout. Section 3 focuses on the development of the model. Section 4 covers data analysis, while Section 5 addresses model ranking. In Section 6, the results and discussion are presented. Section 7 highlights the limitations, and Section 8 concludes with the conclusion and future scope.

2 Assumptions and Notations

The following assumption is taken to develop the model:

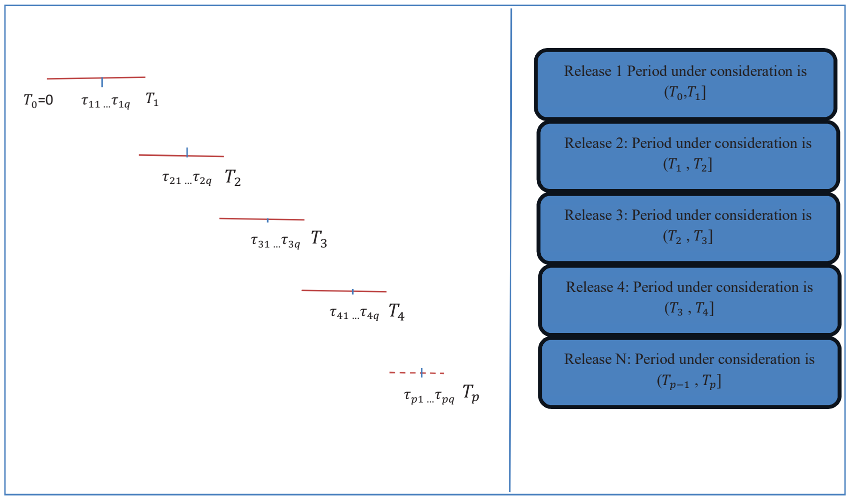
1. The fault removal process is a Non-Homogeneous Poisson Process (NHPP).
2. The software system might randomly fail due to the faults lying in it.
3. Initial fault content in every release is constant.
4. We have assumed that the market opportunity cost is an increasing function of release time. The second derivative of market opportunity cost function exists and is continuous (Jiang et al. 2012).

The following notations are taken to develop the model:

$\boldsymbol{a_i}$:	Faults in i^{th} release of software where $i = 1, 2, 3, \ldots, p$.
$\boldsymbol{p}$:	Represent releases in multi-release software.
$\boldsymbol{m_i(W(t))}$:	Expected number of faults removed during the time interval $T_{i-1} \le t \le T_i$ where $1 \le i \le p$.
$\boldsymbol{W(t)}$:	cumulative testing effort in the time interval $(0, t]$.
$\boldsymbol{t}$:	Represents time. $W(t)$ is a function of time t.
$\boldsymbol{F(W(t))}$:	Failure distribution function.
$\boldsymbol{\tau_{ij}}$:	Patch release timefor j^{th} patch in i^{th} release where $i = 1, 2, 3, \ldots, p$ and $$j = 1, 2, 3, \ldots, q$$
$\boldsymbol{q}$:	Represents patches in a particular release.
$\boldsymbol{T_i}$:	Time till which the testing process is done in i^{th} release where $i = 1, 2, 3, \ldots, p$.
$\boldsymbol{c'}$:	Testing cost.
$\boldsymbol{c''}$:	Market opportunity cost.
$\boldsymbol{c_{ik}}$:	Cost of debugging in each release where $k = 1, 2, 3 \ldots, (q+1)$ because for q number of patches there will be $(q + 1)$ intervals or phases.
$\boldsymbol{b_i}$:	Fault detection rate in i^{th} release of software where $i = 1, 2, 3, \ldots, p$.
$\boldsymbol{c_i}$:	Factor that modifies the failure detection rate over time i^{th} release of software where $i = 1, 2, 3, \ldots, p$.

3 Model Development

In the current study, we have modeled the scenario where the single patch is released between subsequent releases. We have also considered changes



T_0=0
$\tau_{11} \ldots \tau_{1q}$ T_1
$\tau_{21} \ldots \tau_{2q}$ T_2
$\tau_{31} \ldots \tau_{3q}$ T_3
$\tau_{41} \ldots \tau_{4q}$ T_4
$\tau_{p1} \ldots \tau_{pq}$ T_p
Release 1 Period under consideration is (T_0,T_1]
Release 2: Period under consideration is (T_1 , T_2]
Release 3: Period under consideration is (T_2 , T_3]
Release 4: Period under consideration is (T_3 , T_4]
Release N: Period under consideration is (T_{p-1} , T_p]

Figure 2 Timeline depicting the release of patches in a testing effort based on multi-release software.

in testing resource consumption over time. To depict the testing effort in the current study, we used Weibull distribution as defined by (Tickoo et al., 2016).

$$W(t) = W_0(1 - e^{-vt^k}) \tag{1}$$

Where W_0 is total effort expenditure, k is the shape parameter, and v is the scale parameter. Figure 2 is the timeline to depict the release of patches in a testing effort based on multi-release software.

3.1 Release 1

To remain competitive in the market, organizations release software early and address remaining faults through patches. In this study, the time period considered for Release1 is from $[T_0, T_1]$, which denotes first release available in the market. The total number of faults in this release is determined by Equation (2)(Kapur et al. 2016; Kapur et al. 2011), where $m_1(W(t))$ represents the mean value function, $F_1(W(t))$ denotes the failure distribution function, and T_1 signifies the time of the initial software introduction to the market.

Given the scenario of multiple patching, for q patches, there will be $(q + 1)$ phases. Faults reported during the operational phase of the software within intervals $[T_0, \tau_{11}], (\tau_{11}, \tau_{12}], \ldots, (\tau_{1q}, T_1]$ are debugged during phases

$1, 2, \ldots, (q+1)$ respectively. The cumulative number of faults removed at the end of Release 1 is the sum of faults removed in all phases, as expressed in the equations below.

$$m_1(W(t)) = m_{11}(W(t)) + m_{12}(W(t)) + m_{13}(W(t)) + \cdots + m_{1(q+1)}(W(t)) \tag{2}$$

$$\text{where } T_0 \le t \le T_1 \tag{3}$$

$$m_{11}(W(t)) = a_1 F_{11}(W(t)); T_0 \le t \le \tau_{11} \tag{4}$$

$$m_{12}(W(t)) = [a_1 - m_{11}(W(\tau_{11}))]F_{12}(W(t)); \tau_{11} < t \le \tau_{12} \tag{5}$$

$$m_{13}(W(t)) = [[a_1 - m_{11}(W(\tau_{11}))] - m_{12}(W(\tau_{12}))]F_{13}(W(t)); \tau_{12} < t \le \tau_{13} \tag{6}$$

$$m_{14}(W(t)) = [[[a_1 - m_{11}(W(\tau_{11}))] - m_{12}(W(\tau_{12}))] - m_{13}(W(\tau_{13}))] \times F_{14}(W(t)); \tau_{13} < t \le \tau_{14} \tag{7}$$

$$m_{1(q+1)}(W(t)) = \left[a_1 - \sum_{i=1}^{q} m_{1i}(W(\tau_{1i}))\right] F_{1(q+1)}(W(t)); \tau_{1q} < t \le T_1 \tag{8}$$

3.2 Release 2

Release 2 will incorporate faults from the current release and remaining faults from previous releases. The cumulative number of faults removed at the end will be the sum of faults removed in all phases.

$$m_2(W(t)) = m_{21}(W(t)) + m_{22}(W(t)) + m_{23}(W(t)) + \cdots + m_{2(q+1)}(W(t)); T_1 < t \le T_2 \tag{9}$$

$$m_{21}(W(t)) = [a_2 + (a_1 - m_1(W(t)))]F_{21}(W(t)); T_1 < t \le \tau_{21} \tag{10}$$

$$m_{22}(W(t)) = [(a_2 + a_1 - m_1(W(t))) - m_{21}(W(\tau_{21}))]F_{22}(W(t)); \tau_{21} < t \le \tau_{22} \tag{11}$$

$$m_{23}(W(t)) = [[(a_2 + a_1 - m_1(W(t))) - m_{21}(W(\tau_{21}))] - m_{22}(W(\tau_{22}))]F_{23}(W(t)); \tau_{22} < t \le \tau_{23} \tag{12}$$

$$m_{2(q+1)}(W(t)) = \left[(a_2 + a_1 - m_1(W(t))) - \sum_{i=1}^{q} m_{2i}(W(\tau_{2i}))\right] \times F_{2(q+1)}(W(t)); \tau_{2q} < t \leq T_2 \tag{13}$$

3.3 Release 3

Release 3 will incorporate faults from the current release and some remaining faults from previous releases. The cumulative number of faults removed at the end will be the sum of faults removed in all phases.

$$m_3(W(t)) = m_{31}(W(t)) + m_{32}(W(t)) + m_{33}(W(t)) + \cdots + m_{3(q+1)}(W(t)); T_2 < t \leq T_3 \tag{14}$$

$$m_{31}(W(t)) = [a_3 + ((a_2 + (a_1 - m_1(W(t))) - m_2(W(t)))] \times F_{31}(W(t)); T_1 < t \leq \tau_{31} \tag{15}$$

$$m_{32}(W(t)) = [(a_3 + ((a_2 + (a_1 - m_1(W(t))) - m_2(W(t)))) - m_{31}(W(\tau_{31}))]F_{32}(W(t)); \tau_{31} < t \leq \tau_{32} \tag{16}$$

$$m_{33}(W(t)) = [[(a_3 + ((a_2 + (a_1 - m_1(W(t))) - m_2(W(t)))) - m_{31}(W(\tau_{31}))] - m_{32}(W(\tau_{32}))]F_{33}(W(t)); \tau_{32} < t \leq \tau_{33} \tag{17}$$

$$m_{3(q+1)}(W(t)) = \left[(a_3 + ((a_2 + (a_1 - m_1(W(t))) - m_2(W(t)))) - \sum_{i=1}^{q} m_{3i}(W(\tau_{3i}))\right] F_{3(q+1)}(W(t)); \tau_{3q} < t \leq T_3 \tag{18}$$

3.4 Release *p*

The p_{th} release is the final release available in the market. The cumulative number of faults removed at the end will be the sum of faults removed in all phases.

$$m_p(W(t)) = \sum_{k=1}^{(q+1)} m_{pk}(W(t)); T_{(p-1)} < t \leq T_p \tag{19}$$

$$m_{p1}(W(t)) = \left[a_p + \sum_{i=1}^{p-1}(a_i - m_i(W(t)))\right] F_{p1}(W(t)); T_{p-1} < t \leq \tau_{p1} \tag{20}$$

$$\begin{aligned} m_{p2}(W(t)) = &\left[\left(a_p + \sum_{i=1}^{p-1}(a_i - m_i(W(t)))\right) - m_{p1}(W(t))\right] \\ &\times F_{p2}(W(t)); \tau_{p1} < t \leq \tau_{p2} \end{aligned} \tag{21}$$

$$\begin{aligned} m_{p3}(W(t)) = &\left[\left[\left(a_p + \sum_{i=1}^{p-1}(a_i - m_i(W(t)))\right) - m_{p1}(W(t))\right]\right. \\ &\left. - m_{p2}(W(\tau_{p2}))\right] F_{p3}(W(t)); \tau_{p2} < t \leq \tau_{p3} \end{aligned} \tag{22}$$

$$\begin{aligned} m_{p(q+1)}(W(t)) = &\left[\left[\left(a_p + \sum_{i=1}^{p-1}(a_i - m_i(W(t)))\right) - \sum_{j=1}^{q} m_{pj}(W(t))\right]\right. \\ &\times F_{p(q+1)}(W(t)); \tau_{pq} < t \leq T_p \end{aligned} \tag{23}$$

The generalized cost model $C_i(T)$ for multi-patching scenarios in multi-release software where $1 \leq i \leq p$ is given below.

$$\begin{aligned} C_1(T) = &c'T_1 + c''{T_1}^2 + c_{11}m_{11} + c_{12}m_{12} \\ &+ \cdots + c_{1(q+1)}m_{1(q+1)} \end{aligned} \tag{24}$$

$$\begin{aligned} C_2(T) = &c'T_2 + c''{T_2}^2 + c_{21}m_{11} + c_{22}m_{12} \\ &+ \cdots + c_{2(q+1)}m_{2(q+1)} \end{aligned} \tag{25}$$

$$\cdots\cdot$$

$$C_i(T) = c'T_i + c''{T_i}^2 + \sum_{i=1}^{p}\left(\sum_{k=1}^{q+1} c_{ik}m_{ik}\right) \tag{26}$$

Here $c'T_i$ denotes testing cost for i^{th} release, $c''{T_i}^2$ denoted market opportunity cost, $\sum_{i=1}^{p}(\sum_{k=1}^{q+1} c_{ik}m_{ik})$ is the cost of debugging in each interval or phase.

4 Data Analysis

The estimated values of parameters on the tandem dataset for Weibull distribution are given in Table 1. The estimated values of the proposed model for all four releases in given in Table 2. Parameter estimation is conducted using a statistical package for social sciences (SPSS).

The validation of the proposed model is conducted using the Tandem computer dataset (Wood, 1996). We have modeled the fault removal process using three different distribution functions (Saxena et al., 2021) $F(W(t)) = 1 - e^{-bW(t)}$, S-shaped $F(W(t)) = 1 - (1 + bW(t))e^{-bW(t)}$ and flexible $F(W(t)) = \frac{1-e^{-bW(t)}}{1+ce^{-bW(t)}}$.

Based on the experience of the testing team the cost parameters are assumed as: testing cost $c' = 5$, market opportunity cost $c'' = 1$, cost of debugging in the interval $T_{p-1} < t \leq \tau_1$ is 8, the cost of debugging in the interval $\tau_1 < t \leq \tau_2$ is 8; the cost of debugging in the interval $\tau_2 < t \leq T_p$ is 8. Using the cost parameters, we optimize the cost function through genetic algorithm (GA) in MATLAB. The cost function for the proposed model is shown in Equation (26). Assumed cost parameter values can be changed for different scenarios. Using the above parameters, we get the cost function which is the function of two variables i.e. τ_1 and τ_2 for each release. On optimizing the cost function using GA we obtain the optimal cost, and

Table 1 Estimated values for Weibull distribution

Parameters	W_0	v	k
Release 1	11438.411	0.023	1.480
Release 2	12293.040	0.019	1.582
Release 3	5796.362	0.013	2.068

Table 2 Estimated model parameters using SPSS

Model	Release	Estimated Parameters
Model 1	Release 1	$a_1 = 141.226$ $b_1 = 0.00013$
	Release 2	$a_2 = 184.988$ $b_2 = 0.00005$
	Release 3	$a_3 = 85.953$ $b_3 = 0.00013$
Model 2	Release 1	$a_1 = 102.784$ $b_1 = 0.00049$
	Release 2	$a_2 = 120.816$ $b_2 = 0.00027$
	Release 3	$a_3 = 62.601$ $b_3 = 0.00059$
Model 3	Release 1	$a_1 = 193.652$ $b_1 = 0.00002$ $c_1 = 0.773$
	Release 2	$a_1 = 193.650$ $b_2 = 0.00003$ $c_2 = 0.185$
	Release 3	$a_1 = 85.955$ $b_3 = 0.00013$ $c_3 = 0.1$

Table 3 Release-wise optimal cost, optimal release time for 1st patch (τ_1) and optimal release time for 2nd patch (τ_2) for model 1

Release	Optimal Cost ($)	Release Time for 1st Patch	Release Time for 2nd Patch
Release 1	6071.421	6th week	14th week
Release 2	5972.987	10th week	18th week
Release 3	4621.564	5th week	10th week

Table 4 Release-wise optimal cost, optimal release time for 1st patch (τ_1) and optimal release time for 2nd patch (τ_2) for model 2

Release	Optimal Cost ($)	Release Time for 1st Patch	Release Time for 2nd Patch
Release 1	7045.139	11th week	18th week
Release 2	6909.352	6th week	10th week
Release 3	4923.321	4th week	8th week

Table 5 Release-wise optimal cost, optimal release time for 1st patch (τ_1) and optimal release time for 2nd patch (τ_2) for model 3

Release	Optimal Cost ($)	Release Time for 1st Patch	Release Time for 2nd Patch
Release 1	6711.628	7th week	16th week
Release 2	4625.421	6thweek	10th week
Release 3	4812.346	6th week	9th week

optimal release time for 1st patch (τ_1) and optimal release time for 2nd patch (τ_2). Tables 3–5 shows the optimal data for each release.

The validity of proposed frameworks can be judged by goodness of fit parameters such as squared errors (SSE), Mean Square Error (MSE), Predictive Power (PP), correlation index of the regression curve (R^2), theil statistic (TS). The goodness of fit of models on a dataset measure how well a statistical or predictive model aligns with the actual data. The obtained values are shown in Table 6.

5 Model Ranking

Ranking of proposed models based on parameters (SSE, MSE, PP, R^2, TS) was obtained using Entropy coupled with the VIKOR approach. Firstly, the entropy method was used for weight determination, followed by ranking the proposed models using the VIKOR technique. The applied hybrid technique comprises four main phases: Phase 1. Determining frameworks and parameters. Phase 2. Making decision matrix. Phase 3. Identifying the weights of model parameters. Phase 4. Ranking each release in proposed frameworks.

Table 6 Goodness of Fit parameters for the proposed models

Model	Release	SSE	MSE	PP	R^2	TS
Model 1	Release 1	332.437	18.469	0.744	0.981	5.431
	Release 2	205.323	11.407	0.588	0.992	3.829
	Release 3	60.465	3.359	0.645	0.988	5.202
Model 2	Release 1	1057.48	58.749	1.907	0.935	9.686
	Release 2	802.873	44.604	1.998	0.968	7.572
	Release 3	124.503	6.917	1.765	0.976	7.465
Model 3	Release 1	311.528	17.307	0.688	0.981	5.257
	Release 2	205.679	11.427	0.583	0.992	3.832
	Release 3	117.523	6.529	0.658	0.977	7.252

Phase 1: Identifying framework and parameters.

In this phase, we determine the releases $(R_i, i = 1, 2, \ldots, n)$ and parameters $P_j, j = 1, 2, \ldots, m)$. n denotes the total releases of software and m denotes the total model parameters considered for ranking purposes.

Phase 2. Making Decision matrix

The results of phase 1 are represented in the form of a matrix as shown by Equation(27).

$$d_{ij} = \begin{bmatrix} d_{11} & d_{12} & \ldots & d_{1m} \\ d_{21} & d_{22} & \ldots & d_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ d_{n1} d_{n2} & \ldots & d_{nm} & \end{bmatrix} \tag{27}$$

Phase 3. Identifying the weights of model parameters.

First of all, normalize the decision matrix (n_{ij}) by using Equation (28) and computing the entropy value (e_j) using Equation (29).

$$n_{ij} = \frac{d_{ij}}{\sum_{i=1}^{n} d_{ij}} \tag{28}$$

$$e_j = -h \sum_{i=1}^{n} d_{ij} ln\, d_{ij},\ j = 1, 2, \ldots, m \tag{29}$$

$h = \frac{1}{\ln(n)}$ where n is the number of alternatives in our case n is the total software releases.

Calculation of weight vector (w_j) is done using Equation (30) mentioned below.

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{m} (1 - e_j)},\ j = 1, 2, \ldots, m \tag{30}$$

Phase 4. Ranking each release in proposed frameworks is done using the VIKOR technique.

First of all, beneficial and non-beneficial criteria must be identified. Beneficial criteria represent those with a desirable higher value, while non-beneficial criteria entail those with a preferable lower value. Next, the best and worst valuesfor each criterion need to be identified: $(d_i)_{max}$ for beneficial criteria, $(d_i)_{min}$ for non-beneficial criteria and worst-case $(d_i)_{min}$ for beneficial criteria, $(d_i)_{max}$ for non-beneficial criteria.

Next, calculate the utility measure (S_i) as shown in Equation (31) and calculate individual regret measure (R_i) as shown in Equation (32).

$$S_i = \sum_{j=1}^{m} \left(w_j * \frac{(d_i)_{max} - d_{ij}}{(d_i)_{max} - (d_i)_{min}} \right) \tag{31}$$

$$R_i = \max \left(w_j * \frac{(d_i)_{max} - d_{ij}}{(d_i)_{max} - (d_i)_{min}} \right) \tag{32}$$

Calculate the following values as shown in Equation (33).

$$S^* = minS_i;\ S^- = \max S_i;\ R^* = \min R_i;\ R^- = \max R_i \tag{33}$$

Now calculate VIKOR index (Q_i) shown in Equation (34). Here v represents maximum group utility and $(1 - v)$ represents individual regret usually the value of v is taken as 0.5.

$$Q_i = v * \frac{S_i - S^*}{S^- - S^*} + (1 - v) * \frac{R_i - R^*}{R^- - R^*} \tag{34}$$

Finally rank the models based on the VIKOR index.

6 Results and Discussion

The goodness of fit of models on the dataset can be illustrated through the graphs depicted in Figures 3–5. From the graphs, it is evident that Model 3 provides the best prediction of faults for Release 1 and Release 2, while

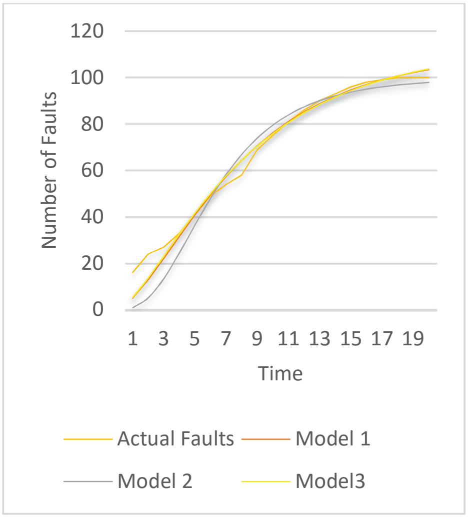


Figure 3 The goodness of fit for release 1.

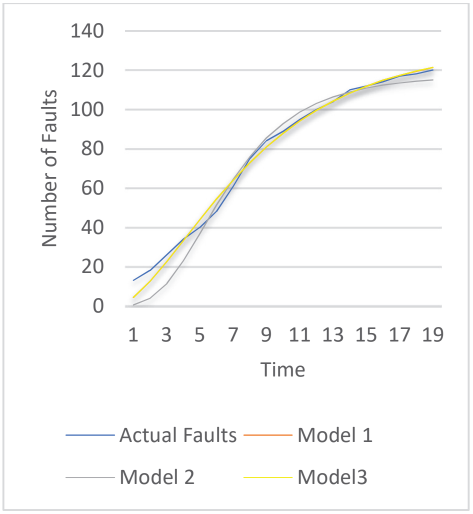


Figure 4 The goodness of fit for release 2.

Model 1 is the most effective for predicting faults in Release 3. To attain a clearer understanding of the best suited model for a specific dataset further MCDM technique is applied to obtain model ranking. The obtained rankingis shown in Table 7. It can be seen that release 1 and release 2 can be best

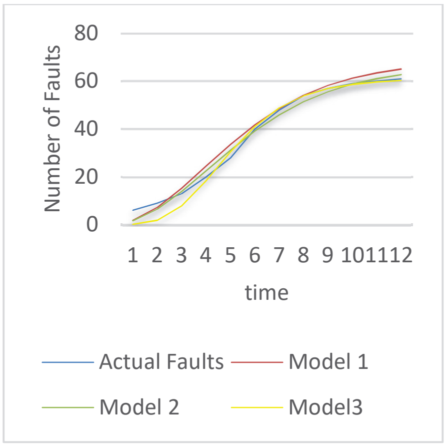


Figure 5 The goodness of fit for release 3.

Table 7 Model ranking obtained based on the ENTROPY-VIKOR technique

Models	Release 1	Release 2	Release 3
Model 1	2	2	1
Model 2	3	3	3
Model 3	1	1	2

modeled with the help of model 3. Release 3 can be best simulated with the help of model 1.

7 Limitations

The current study focuses on modeling multi-patch, multi-release testing effort based SRGM which is a novel approach. We have incorporated MCDM to get a clear image of the best-suited framework for a dataset. MCDM approach and data set can be changed as per the organization's requirement. It is essential to recognize that software functions in diverse environments are characterized by varying user configurations, hardware, and usage patterns. Existing models may not fully capture this complexity, which can restrict their applicability across different contexts. Additionally, a limitation of our model is that we utilized the Entropy-VIKOR approach for ranking; however, other hybrid approaches could also be employed depending on the specific dataset and its characteristics.

8 Conclusion and Future Scope

As our faith in software continues to grow, so too does the imperative for software organizations to deliver good quality software efficiently. Continuous testing is important for ensuring quality software, yet a prolonged testing process can incur significant opportunity costs in the fast-paced market scenario. To navigate this challenge, organizations increasingly adopt a strategy of releasing software early and subsequently issuing patches to address any remaining bugs or vulnerabilities. Patches offer a cost-effective solution for implementing minor changes and updates to software. This paper introduces a novel testing effort based SRGM to study the impact of multiple patches on multi-release software. Based on the dataset the optimal cost and patch release time are identified. Based on the MCDM analysis, Model 3 emerges as the optimal choice for Release 1 and Release 2, effectively meeting the criteria for these releases. For Release 3, Model 1 is identified as the best fit, aligning closely with the unique requirements of this release. This selection process highlights how MCDM can guide the choice of the most appropriate models across different releases by evaluating and ranking each model based on key performance criteria.

In essence, the integration of patching strategies within software development processes underscores the importance of adaptability and responsiveness in meeting evolving user demands and market dynamics. Overall, this paper provides a comprehensive framework for managing the complexities of software maintenance and quality assurance in an era of multiple releases and multiple patches. This novel approach offers software developers and stakeholders valuable insights into effective patch management practices in a dynamic market environment. By embracing such methodologies, organizations can not only enhance the reliability and quality of their software products but also maintain competitiveness in an ever-evolving technological landscape. In the future, this study can be extended by discussing more on the reliability of multi-patch multi-release software. A comparative analysis between MCDM techniques can also be done.

References

Anand, A., Agarwal, M., Tamura, Y., and Yamada, S. (2017). Economic impact of software patching and optimal release scheduling. *Quality and Reliability Engineering International*, *33*(1), 149–157.

Aggarwal, A. G., Gandhi, N., Verma, V., and Tandon, A. (2019). Multi-release software reliability growth assessment: an approach incorporating fault reduction factor and imperfect debugging. *International Journal of Mathematics in Operational Research*, *15*(4), 446–463.

Bibyan, R., Anand, S., Aggarwal, A. G., and Tandon, A. (2023). Multi-Release Testing coverage-based SRGM considering error generation and change-point incorporating the random effect.

Bibyan, R., and Anand, S. (2022). Ranking of Multi-release Software Reliability Growth Model Using Weighted Distance-Based Approach. *Optimization Models in Software Reliability*, 355–373.

Chaube, S., Pant, S., Kumar, A., Uniyal, S., Singh, M. K., Kotecha, K., and Kumar, A. (2024). An Overview of Multi-Criteria Decision Analysis and the Applications of AHP and TOPSIS Methods. *International Journal of Mathematical, Engineering & Management Sciences*, *9*(3), 581–615.

Choudhary, C., Kapur, P. K., Khatri, S. K., and Shrivastava, A. K. (2019). Effort-based release and patching time of software with warranty using change point. *International Journal of Performability Engineering*, *15*(6), 1724.

David, I. J., Stephen, M., and Thomas, E. J. (2023). Reliability Analysis with New Sine Inverse Rayleigh Distribution. *Journal of Reliability and Statistical Studies*, 255–268.

Dev, S., Aherwar, A., and Patnaik, A. (2020). Material selection for automotive piston component using entropy-VIKOR method. *Silicon*, *12*, 155–169.

Garg, R. (2019). Parametric selection of software reliability growth models using multi-criteria decision-making approach. *International Journal of Reliability and Safety*, *13*(4), 291–309.

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

Guesmi, H. A., and Madbouly, S. O. (2023). Evaluation of the Reliability of a Standby Redundancy System Under Real Conditions. *Journal of Reliability and Statistical Studies*, 357–372.

Gupta, A., Gupta, N., and Garg, R. K. (2018). Implementing weighted entropy-distance based approach for the selection of software reliability growth models. *International Journal of Computer Applications in Technology*, *57*(3), 255–266.

Huang, C. Y., Kuo, S. Y., and Chen, Y. (1997, November). Analysis of a software reliability growth model with logistic testing-effort function. In *Proceedings The Eighth International Symposium on Software Reliability Engineering* (pp. 378–388). IEEE.

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

Inoue, S., and Yamada, S. (2013). Lognormal process software reliability modeling with testing-effort.

Kansal, Y., Singh, G., Kumar, U., and Kapur, P. K. (2016). Optimal release and patching time of software with warranty. *International Journal of System Assurance Engineering and Management*, *7*, 462–468.

Kapur, P. K., Mishra, P., Shrivastava, A. K., and Khatri, S. K. (2016, February). Multi-release modeling of a software with testing effort and fault reduction factor. In *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)* (pp. 54–59). IEEE.

Kapur, P. K., Pham, H., Gupta, A., and Jha, P. C. (2011). *Software reliability assessment with OR applications* (Vol. 364). London: Springer.

Kapur, P. K., Singh, O., and Shrivastava, A. K. (2018). A unified approach for optimal release, patching and testing time of a software. *International Journal of Mathematics in Operational Research*, *13*(4), 471–491.

Kumar, A., Garg, P., Pant, S., Ram, M., and Kumar, A. (2022). Multi-Criteria Decision-Making Techniques for Complex Decision Making Problems. *Mathematics in Engineering, Science & Aerospace (MESA)*, *13*(2).

Li, Q., Li, H., and Lu, M. (2015). Incorporating S-shaped testing-effort functions into NHPP software reliability model with imperfect debugging. *Journal of Systems Engineering and Electronics*, *26*(1), 190–207.

Musa, J. D., and Okumoto, K. (1984, March). A logarithmic Poisson execution time model for software reliability measurement. In *Proceedings of the 7th international conference on Software engineering* (pp. 230–238).

Ohba, M. (1984). Software reliability analysis models. *IBM Journal of research and Development*, *28*(4), 428–443.

Pant, S., Garg, P., Kumar, A., Ram, M., Kumar, A., Sharma, H. K., &Klochkov, Y. (2024). AHP-based multi-criteria decision-making approach for monitoring health management practices in smart healthcare system. *International Journal of System Assurance Engineering and Management*, *15*(4), 1444–1455.

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

Pradhan, S. K., Kumar, A., Kumar, V., and Kapur, P. K. (2024). Imperfect Debugging, Testing Coverage, and Compiler Error-Based SRGM with Two Types of Faults Under the Uncertainty of the Operating Environment. In *Reliability Engineering for Industrial Processes: An Analytics Perspective* (pp. 361–380). Cham: Springer Nature Switzerland.

Rafi, S. M., Rao, K. N., and Akthar, S. (2010). Incorporating generalized modified Weibull TEF in to software reliability growth model and analysis of optimal release policy. *Computer and Information science*, *3*(2), 145.

Rawat, S. S., Pant, S., Kumar, A., Ram, M., Sharma, H. K., and Kumar, A. (2022). A state-of-the-art survey on analytical hierarchy process applications in sustainable development. *Int. J. Math. Eng. Manag. Serv*, *7*, 883–917.

Samal, U., Kushwaha, S., and Kumar, A. (2023). A Testing-Effort Based Srgm Incorporating Imperfect Debugging and Change Point. *Reliability: Theory & Applications*, *18*(1 (72)), 86–93.

Saxena, P., Singh, N., Shrivastava, A. K., and Kumar, V. (2021). Testing effort based SRGM and release decision under fuzzy environment. *International Journal of Reliability and Safety*, *15*(3), 123–140.

Saxena, P., Kumar, V., and Ram, M. (2022). A novel CRITIC-TOPSIS approach for optimal selection of software reliability growth model (SRGM). *Quality and Reliability Engineering International*, *38*(5), 2501–2520.

Saxena, P., Kumar, V., Tandon, S., Chaudhary, K., and Ram, M. (2024). Modeling Software Release Time and Software Patch Release Time Based on Testing Effort and Warranty. *Journal of Reliability and Statistical Studies*, 77–108.

Saraf, I., and Iqbal, J. (2019). Generalized multi-release modelling of software reliability growth models from the perspective of two types of imperfect debugging and change point. *Quality and Reliability Engineering International*, *35*(7), 2358–2370.

Saxena, P., Singh, N., Shrivastava, A. K., and Kumar, V. (2021). Testing effort based SRGM and release decision under fuzzy environment. *International Journal of Reliability and Safety*, *15*(3), 123–140.

Sharma, T., Kumar, A., Pant, S., and Kotecha, K. (2023). Wastewater Treatment and Multi-Criteria Decision-Making Methods: A Review. *IEEE Access*.

Singh, V., Kumar, V., and Singh, V. B. (2023). A hybrid novel fuzzy AHP-Topsis technique for selecting parameter-influencing testing in software development. *Decision Analytics Journal*, *6*, 100159.

Singh, V., Kumar, V., and Singh, V. B. (2024). Multi-release software: A decision making mathematical approach for analysing the impact of infected patch. *Mathematics in Engineering, Science & Aerospace (MESA)*, *15*(2).

Tickoo, A., Kapur, P. K., Shrivastava, A. K., and Khatri, S. K. (2016). Testing effort based modeling to determine optimal release and patching time of software. *International Journal of System Assurance Engineering and Management*, *7*, 427–434.

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

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

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

Biographies

Veenu Singh has published a number of papers in referred Journals. She has presented various academic as well as research-based papers at several national and international conferences. Her research activity is set to explore the developmental role for the software industry.

Vijay Kumar received his MSc in Applied Mathematics and MPhil in Mathematics from Indian Institute of Technology(IIT), Roorkee, India in 1998 and 2000, respectively. He has completed his PhD from the Department of Operational Research, University of Delhi. Currently, he is a Professor in the Department of Mathematics, Amity Institute of Applied Sciences, Amity University, Noida, India. He is co-editor of two book and has published more than 70 research papers in the areas of software reliability, mathematical modelling and optimisation in international journals and conferences of high repute. His current research interests include software reliability growth modelling, optimal control theory and marketing models in the context of innovation diffusion theory. He has edited special issues of IJAMS andRIO journal. He is an editorial board member of IJSA, Springer. He is a life member of Society for Reliability Engineering, Quality and Operations Management (SREQOM).

V. B. Singh is a computer science professor at Jawaharlal Nehru University (JNU). He has a Ph.D. in Software Engineering from the University of Delhi, an M.C.A. from M.M.M. Engineering College in Gorakhpur, and a B.Sc. from Udai Pratap College in Varanasi. His research interests include machine learning and software evolution. He has published more than 100 research articles in international journal and conferences of high repute.

Arun Prakash Agrawal is a professor at School of Computer Science Engineering and Technology Bennett University Greater Noida, India. He has a Ph.D. in Software Engineering from GGSIPU Delhi, His research interests include machine learning and software engineering. He has published more than 70 research articles in international journals and conferences of high repute.