JOURNAL OF INFORMATION SCIENCE AND ENGINEERING XX, XXXX-XXXX (2016)
DOI: XX.XXX.XXXX.XX

An Enhanced Software Reliability Growth Model Considering Dynamic Fault Removal Efficiency and Residual Error Change Rate*

UMASHANKAR SAMAL+, AJAY KUMAR
Department of Engineering Sciences
Atal Bihari Vajpayee Indian Institute of Information Technology and Management
Gwalior, India
E-mail: umashankar@iiitm.ac.in

In our fast-paced modern world, software systems have become indispensable in both personal and professional spheres. With increasing reliance on software products, the demand for reliable and high-quality software has intensified, placing significant pressure on developers to stay competitive. Software reliability growth models (SRGMs) play a vital role in assessing the dependability of software systems during their development. These models mathematically analyze the relationship between detected faults, testing time, and failures, enabling the prediction of software failures. This paper introduces an approach to software reliability evaluation, considering the dynamic nature of fault removal efficiency (FRE) during development. Additionally, the model accounts for the change rate of residual errors, considering both error introduction and correction processes. Moreover, the adoption of S-shaped curves captures the learning process of software developers, enhancing the model's accuracy. This approach guides software developers to make informed decisions, leading to improved software reliability and performance, meeting escalating demands.

Keywords: Software reliability growth model, non-homogeneous Poisson process, fault removal efficiency, mean value function, residual error

1. Introduction

In the modern era, software systems have become indispensable in our lives, permeating both personal and professional spheres. From smartphones and smart home devices to critical banking and financial systems, our reliance on software products is ever-increasing. As the demand for dependable software rises, developers face mounting pressure to deliver high-quality and reliable products to stay competitive in the market.

Software reliability models play a critical role in assessing the dependability of software systems throughout their development process. The ultimate goal is to ensure that the software performs without failure within specified time frames and under predefined conditions [1]. These models achieve this by mathematically analyzing the relationship between the number of faults detected and removed, or failures experienced, and the testing time. Such mathematical frameworks are commonly referred to as software reliability

Received ** **, 2014; revised ****, ****; accepted ****.
+Corresponding author
Communicated by

growth models (SRGMs). Over the last few decades, researchers have proposed numerous SRGMs that leverage the concept of non-homogeneous Poisson process (NHPP) [2, 3, 4]. These models aim to predict software failures and determine the optimal release time, taking into account the dynamic nature of software development and testing.

Numerous researchers have actively contributed to the domain by proposing innovative models that adeptly fit historical failure data. Goel and Okumoto [5] put forth a reliability assessment stochastic model, grounded in NHPP, which employs an exponential curve to represent failure occurrences. Yamada et al. [6] introduced the delayed S-shaped model, a type of SRGM that divides testing into fault detection and removal phases. This model considers the team's learning process and growing skills. Meanwhile, Ohba [7] proposed the inflection S-shaped model, assuming that some faults remain undetected until others are removed. Pham et al. [8] presented a software reliability model based on NHPP that incorporates imperfect debugging and dynamic improvement of testing efficiency throughout the testing phase. Previous models and empirical evidence indicate that the learning-induced efficiency growth can follow various curves, ranging from linear to logistic in nature.

Fault removal efficiency (FRE) is a vital concept in software reliability engineering, assessing the effectiveness of fault removal during testing and development. It calculates the proportion of identified and corrected faults compared to the total faults present in the software, reflecting how well the testing process detects and resolves faults to enhance software reliability. Numerous studies have explored SRGMs, integrating FRE as a key factor in their analyses. Zhang et al. [9] presented a software reliability model in which constant FRE and fault introduction rate were integrated. The model captured the learning process of software developers with an inflection S-shaped curve and addressed imperfect debugging by considering faults that could be introduced during debugging with a constant fault introduction probability. Liu et al. [10] proposed a model with constant FRE and fault introduction rate, implying a reduction in the number of residual faults during effective software debugging. Li and Pham [11] proposed an innovative software reliability model, taking into account error generation, constant FRE, and testing coverage. By utilizing testing coverage to express the fault detection rate and incorporating FRE for fault repair, their model represents a substantial advancement in software reliability evaluation. Li and Mao [12] considered two crucial assumptions: the influence of different field environments on software performance and reliability, and the imperfect nature of fault removal during the debugging process. Their paper aimed to incorporate FRE, testing coverage, and the randomness of field environments into a new software reliability model. Kapur et al. [13] proposed a generalized SRGM that estimates faults during testing and can be extended to the operation phase. In testing, it focuses on reliability growth with respect to testing resources, while in the operational phase, FRE determines the effort spent on user-reported faults, and reliability growth depends on software usage. Chatterjee and Shukla [14] introduced a general SRGM based on NHPP, constant FRE, and an optimal software release policy considering cost and reliability criteria. Their study aimed to develop a software release time decision model, taking into account maintenance cost and warranty cost in a fuzzy environment. Chatterjee et al. [15] proposed an SRGM that simultaneously considers fault detectability, fault removability, fault exposure ratio, and constant FRE, while also incorporating the effect of the change point. The study's results revealed a significant correlation between FRE and fault removability, indicating that an

increase in FRE corresponds to higher fault removability, and vice versa. Verma et al. [16] introduced a unified SRGM that accounts for time-dependent fault removal factor (FRF), constant FRE, and constant error generation. The model employed exponential, Weibull, and delayed S-shaped distributions to effectively model FRF. Haque and Ahmad [17] presented a software reliability model that factored in perfect debugging and constant FRE. The fault detection rate was effectively represented using a two-parameter logistic function. Samal and Kumar [18] proposed an SRGM that accounts for a constant FRE within an perfect debugging environment.

Debugging assumes a critical role in maintaining and enhancing software reliability. Whenever an error is reported, the debugging process is initiated to identify and resolve the underlying issue. However, debugging is not always perfect and may unintentionally introduce new faults during the fix. The error generation rate serves as a significant factor in SRGM, quantifying the probability of introducing new faults while debugging. These newly introduced faults can stem from coding errors, logic flaws, or incomplete understanding of the problem at hand during debugging. Numerous authors have explored the influence of the error generation rate on SRGM.

Kapur et al. [19] introduced two general frameworks to derive multiple SRGMs using NHPP with considerations for imperfect debugging and error generation. The proposed models were formulated to accommodate scenarios with no differentiation between failure observation and fault removal testing processes, and were later extended to address cases where a clear differentiation between these processes exists. Roy et al. [20] introduced a software reliability model with an exponentially increasing fault content function and a constant fault detection rate. The total fault content of the software exhibited rapid growth at the initial stages of testing but gradually slowed down towards the end, attributed to the increasing efficiency of the testing team.

In the current literature, many authors have treated FRE as a constant value, overlooking its dynamic nature throughout the software development process. This oversight neglects potential variations in fault removal effectiveness over time as software development involves a complex interplay of factors such as evolving requirements, changing team dynamics, and shifting project priorities, all of which can influence the effectiveness of fault removal activities. While some research has considered FRE, limited attention has been given to exploring the dynamic nature of FRE and its impact on the change rate of residual errors. Within the research paper, we present the following key contributions:

- Introducing a dynamic FRE function that depends on the mean value function (mvf), representing the expected number of failures by a certain time.
- Taking into account the change rate of residual errors, encompassing both the introduction of new errors and the correction of faults.
- Adopting S-shaped curve to describe the fault detection rate, capturing the learning process of software developers.

The rest of the paper is systematized as follows: Section 2 of the paper explores the study's background and offers essential definitions to provide a foundational understanding. In Section 3, the model assumptions and formulation are elaborated, presenting the analytical model used for software reliability prediction. Section 4 presents experimental

details, including dataset descriptions, model comparisons, and goodness-of-fit criteria, to validate the proposed SRGM. The results obtained from applying the model and parameter estimation technique to real-world data are presented and discussed in Section 5. Finally, Section 6 presents the conclusion, summarizing the key findings, and discusses future research directions, proposing areas for improvement in the model.

2. Background

In this section, we provide the foundational background necessary for understanding the subsequent discussions on NHPP and software reliability. NHPP is characterized by its intensity function, $\lambda(t)$, which dictates the rate of events occurring over time. Following this, we discuss concept of software reliability which is crucial in ensuring the dependability and performance of software systems, with implications spanning from user satisfaction to system safety criticality.

2.1 NHPP

A computing process $N(t)$ is said to be **NHPP** with $\lambda(t)$ as its intensity function if a Poisson distribution with a mean value function $m(t), t \geq 0$ describes it [8].

$$Pr\{N(t) = k\} = \frac{[m(t)]^k}{k!} e^{-m(t)},\ k = 0, 1, 2, \cdots \tag{1}$$

$$\text{where } m(t) = E[N(t)].$$

The fault intensity function $\lambda(t)$ can be used to solve m(t).

$$m(t) = \int_0^t \lambda(s) ds. \tag{2}$$

2.2 Software reliability

Software reliability $R(x/t)$ is the likelihood that no software fault will be found between the times $(t, t+x)$, assuming that the prior fault occurred at time t, where $t \geq 0$, $x > 0$ [2].

$$R(x/t) = e^{-[m(t+x)-m(t)]}. \tag{3}$$

2.3 Least squares estimation (LSE)

Parameter estimation plays a crucial role in predicting software reliability. It involves determining unknown parameters in a model's analytical solution. The widely adopted technique for this task is LSE, which works by minimizing the sum of squared differences between observed data and model predictions. Mathematically, LSE aims to minimize the objective function

$$\sum_{min} (y_i - m(t_i))^2, \tag{4}$$

which is the sum of squared discrepancies between the observed data y_i and the model predictions $m(t_i)$.

Table 1. Notations

$m(t)$	The anticipated quantity of software defects fixed by time t
$a(t)$	Fault content function
$b(t)$	Software fault detection rate per unit of time
ψ	Fault removal efficiency
Ψ	Initial fault removal efficiency
α	Shape parameter
p	Change rate of generalized residual errors
a_0	Initial fault content

3. Model Development

In this section, we undertake the formulation of the proposed SRGM, focusing on the integration of a dynamic FRE, the consideration of the change rate of residual errors, and the incorporation of the learning process observed in software developers. We present the notations employed throughout the study, establishing a foundational understanding for our subsequent discussions. Following this, we articulate several key assumptions that underpin our model's construction, serving as guiding principles for our approach. Subsequently, we explore the formulation of our model, drawing upon relevant literature. However, while some research has addressed FRE, there has been limited exploration into its dynamic nature and its impact on the change rate of residual errors. Our goal is to address potential variations in FRE over time, recognizing the complex interaction of factors such as evolving requirements, changing team dynamics, and shifting project priorities, all of which can influence the efficacy of fault removal activities.

3.1 Notations

The study utilizes various symbols and abbreviations, which are listed in Table 1.

3.2 Assumption

The proposed model relies on the following assumptions:

1. Software failure process conforms to the principles of NHPP.
2. The debugging process promptly initiates as soon as an error occurs.
3. The debugging process is imperfect i.e. new errors are introduced.
4. The correction of errors is not absolute; rather, the detected error will be rectified with a known probability ψ, referred to as the FRE.
5. The change rate of residual errors is modeled using a dynamic process that includes both error introduction and error correction.
6. The fault detection process exhibits characteristics akin to a learning curve phenomenon.

3.3 Formulation

An NHPP software reliability model that incorporates the consideration of FRE is given by [9]:

$$\frac{dm(t)}{dt} = b(t)\left[a(t) - \psi m(t)\right]. \tag{5}$$

The term $\frac{dm(t)}{dt}$ represents the rate of change of the number of faults with respect to time t. The expression $b(t)\left[a(t) - \psi m(t)\right]$ represents the net fault arrival rate at time t after accounting for the effectiveness of fault removal. It captures the balance between new faults being introduced into the system $a(t)$ and the removal of existing faults $\psi m(t)$.

During actual debugging processes, FRE may decline as the number of detected faults (errors) increases. This can be attributed to several reasons: the correlation information among errors reduces over time, making it more difficult to modify errors, the natural tendency to focus on the current task may limit consideration of error relationships, while the mean modifying level and experience of debuggers can decrease due to unpredictable factors. As a result, FRE can be represented by a decreasing exponential function of m(t), as given in Equation 6 accounting for the decrease in efficiency as the number of detected errors increases.

$$\psi = \Psi e^{-\alpha m(t)}, \tag{6}$$

where Ψ is the initial FRE and α determines the shape of FRE.

The introduction of new errors tends to occur during the error correction phase rather than the error detection phase. Regardless of whether corrections are successful or not during the error debugging period, the probability of introduction of new errors increases with the number of correction processes (sometimes requiring multiple attempts until the error is completely resolved). Since the introduction of new errors happens alongside error correction, it is appropriate to describe the change rate of residual errors using a dynamic process that considers both the introduction and correction of errors. We define the expression $\left[a(t) - \psi m(t)\right]'$ as the change rate of generalized residual errors. Assuming that the number of residual errors generally decreases over time, we consider the average ratio between the error introduction rate and the change rate of generalized residual errors to be $-p$, as given in Equation 7.

$$\frac{a'(t)}{\left[a(t) - \psi m(t)\right]'} = -p. \tag{7}$$

In accordance with assumption 6, we have chosen a S-shaped curve, as given in Equation 8, to represent the fault detection rate. As the project team gains experience and improves their skills over time, they become more proficient at detecting faults, leading to the adoption of the S-shaped curve to represent this learning behavior.

$$b(t) = \frac{b^2 t}{1 + bt}. \tag{8}$$

Now, by solving Equation 5 considering Equation 6 and 7 under the initial conditions

$m(0) = 0$, $a(0) = a_0$, we will get:

$$m(t) = \frac{1}{\alpha}\log\left(\frac{p\Psi - \Psi + a_0\alpha e^{bt(a_0\alpha - \Psi + p\Psi)}(bt+1)^{\Psi - a_0\alpha - p\Psi}}{a_0\alpha - \Psi + p\Psi}\right). \tag{9}$$

4. Experimental setting

4.1 Model validation

A model's effectiveness is assessed by contrasting its advantages, disadvantages, and degree of accuracy. The proposed model's performance analysis is described and contrasted with a few SRGMs as given in Table 2. Notably, Model M_4, M_5, M_6, M_7, and M_8 incorporate FRE, while Model M_1, M_2, and M_3 do not.

Table 2. SRGMs taken for comparison

Sl.No.	Model	MVF
1	M_1 [6]	$m(t) = \frac{ab}{b+\alpha}\left(e^{\alpha t} - e^{-bt}\right)$
2	M_2 [8]	$m(t) = \frac{a}{1+\beta e^{-bt}}\left(\left[1-e^{-bt}\right]\left[1-\frac{\alpha}{\beta}\right]+\alpha t\right)$
3	M_3 [21]	$m(t) = N\left(1-\frac{\beta}{\beta+\ln\left(\frac{a+e^{bt}}{1+a}\right)}\right)^{\alpha}$
4	M_4 [16]	$m(t) = \frac{a}{\psi-\alpha}\left(1-e^{-b\beta}(\psi-\alpha)t\right)$
5	M_5 [16]	$m(t) = \frac{a}{\psi-\alpha}\left(1-e^{-b\beta}(\psi-\alpha)t^k\right)$
6	M_6 [16]	$m(t) = \frac{a}{\psi-\alpha}\left(1-\left((1+bt)^{\beta(\psi-\alpha)}e^{-bt\beta(\psi-\alpha)}\right)\right)$
7	M_7 [17]	$m(t) = \frac{N}{\psi}\left[1-\frac{\beta+1}{(\beta+e^{bt})^{\psi}}\right]$
8	M_8 [18]	$m(t) = \frac{N}{\psi}\left[1-e^{\psi(\ln(bt+1)-bt)}\right]$

4.2 Description of dataset

We have used two datasets to validate our model. The first dataset (DS-1), as given in Table 3, was obtained from testing a medium-sized software system [22]. A number of 144 cumulative faults (C.F.) were observed during a 17-week testing period. The second dataset (DS-2), as given in Table 4, was obtained from second release of Tandem Computer project [23]. A total number of 120 faults were observed during a 19-week testing period. These datasets have been extensively used in numerous researchers.

Table 3. DS-1

week	C.F.	week	C.F.	week	C.F.
1	12	8	111	15	132
2	23	9	112	16	141
3	43	10	114	17	144
4	64	11	116	-	-
5	84	12	123	-	-
6	97	13	126	-	-
7	109	14	128	-	-

Table 4. DS-2

week	C.F.	week	C.F.	week	C.F.
1	13	8	75	15	112
2	18	9	84	16	114
3	26	10	89	17	117
4	34	11	95	18	118
5	40	12	100	19	120
6	48	13	104	-	-
7	61	14	110	-	-

4.3 Goodness-of-fit criteria

In this paper, we compare the goodness of fit for all models using three criteria. The mean squared error (MSE) measures the average squared error between the estimated and actual values. Mean absolute error (MAE) quantifies the average absolute difference between predicted and true values in a regression problem. It considers the magnitude of errors without considering their direction. R-squared (R^2) is a statistical metric that indicates the proportion of the variance in the dependent variable explained by the independent variables. R^2 ranges from 0 to 1, with 0 indicating no explanation of variability and 1 representing a perfect explanation of all variability.

$$\mathbf{MSE} = \frac{\sum_{i=1}^{n} (\widehat{m}(t_i) - y_i)^2}{n-p}. \tag{10}$$

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} \mid \widehat{m}(t_i) - y_i \mid}{n}. \tag{11}$$

$$\mathbf{R}^2 = 1 - \frac{\sum_{i=1}^{n} (\widehat{m}(t_i) - y_i)^2}{\sum_{i=1}^{n} (\widehat{m}(t_i) - \overline{y})^2}. \tag{12}$$

The terms n, p, $\widehat{m}(t_i)$, y_i and $\overline{y}$ given in Equations 10, 11, and 12 represent the number of samples, number of parameters, estimated cumulative number of faults by time t_i, total number of faults observed by time t_i, and mean number of faults observed by time t_i, respectively.

5. Results and Discussion

The results obtained from analyzing the estimated parameters for DS-1 clearly demonstrate the superiority of the proposed model over eight other models as listed in Table 2. This conclusion is supported by the evaluation metrics of MSE, MAE, and R^2,

which exhibit the smallest values for the proposed model, 40.1132, 4.7471, and 0.9907, respectively as given in Table 5. Among the nine models compared, the proposed model stands out with the lowest MSE, indicating its excellent predictive accuracy in capturing the cumulative failures of the software. The MAE, another important performance metric, also confirms that the proposed model outperforms the others. The R^2 value, which measures the goodness of fit, further supports the superiority of the proposed model, as it achieved an impressive value of 0.9907, closely followed by M_6 with an R^2 value of 0.9902. Figure 1 provides a visual representation of the comparison between the actual cumulative failures and the predictions made by the seven software reliability models for release-1.

The analysis of the estimated parameters for DS-2 leaves no doubt about the superiority of the proposed model when compared to the other eight models listed in Table 2. This superiority is evident through the evaluation metrics of MSE, MAE, and R^2, all of which yield the smallest values for the proposed model: 9.3885, 1.9196, and 0.9974, respectively as given in Table 6. Out of the nine models compared, the proposed model stands out with the lowest MSE, signifying its exceptional predictive accuracy in capturing the cumulative failures of the software. Furthermore, the MAE results confirm that the proposed model outperforms all other models, including the second-best performer, Model M_5, which obtained MSE=13.7593 and MAE=2.2109. The R^2 value, serving as a measure of goodness of fit, further strengthens the case for the proposed model's superiority, achieving an impressive value of 0.9974, with Model M_7 coming close with an R^2 value of 0.9972. For a more intuitive understanding of the comparisons, Figure 2 visually depicts the disparity between the actual cumulative failures and the predictions made by all seven software reliability models for release-2.

Table 5. Parameter estimation: DS-1

Model	MSE	MAE	R^2	Parameter values
M_1	57.8172	6.0271	0.9858	$\hat{a} = 318.0521$, $\hat{b} = 0.0637$, $\hat{\alpha} = -0.0449$
M_2	45.9697	5.2001	0.9897	$\hat{a} = 87.5504$, $\hat{b} = 0.8607$, $\hat{\alpha} = 0.0407$, $\hat{\beta} = 13.0771$
M_3	56.9518	5.6175	0.9852	$\hat{N} = 175.0198$, $\hat{a} = 10.3152$, $\hat{b} = 5.2920$, $\hat{\alpha} = 3.2577$, $\hat{\beta} = 7.1144$
M_4	69.1474	5.9041	0.9862	$\hat{a} = 121.6908$, $\hat{b} = 0.3624$, $\hat{\psi} = 0.7019$, $\hat{\alpha} = -0.0872$, $\hat{\beta} = 0.4922$
M_5	49.3629	4.8692	0.9896	$\hat{a} = 124.4562$, $\hat{b} = 0.3063$, $\hat{\psi} = 0.6714$, $\hat{\alpha} = -0.2466$, $\hat{\beta} = 0.3448$, $\hat{k} = 1.3583$
M_6	47.9498	5.2059	0.9902	$\hat{a} = 158.5272$, $\hat{b} = 0.8125$, $\hat{\psi} = 1.1405$, $\hat{\alpha} = 0.0103$, $\hat{\beta} = 0.3149$
M_7	46.2469	5.1230	0.9886	$\hat{N} = 91.5623$, $\hat{b} = 0.5174$, $\hat{\beta} = 2.4821$, $\hat{\psi} = 0.6821$
M_8	57.1466	5.8191	0.9818	$\hat{N} = 49.8623$, $\hat{b} = 0.8169$, $\hat{\psi} = 0.3478$
Proposed	40.1132	4.7471	0.9907	$\hat{a_0} = 27.3493$, $\hat{b} = 1.2388$, $\hat{p} = -0.7057$, $\hat{\alpha} = -0.0003$, $\hat{\psi} = 0.1119$

6. Conclusion and future scope

This study introduces an SRGM that incorporates dynamic FRE as a function of the mvf, departing from the traditional assumption of constant FRE. The proposed model

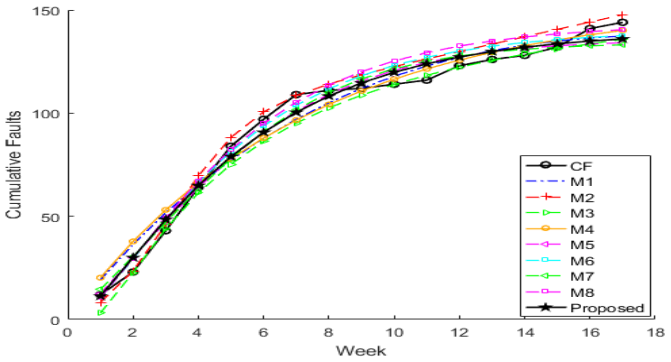

Fig. 1. Performance analysis of models: DS-1

Table 6. Parameter estimation: DS-2

Model	MSE	MAE	R^2	Parameter values
M_1	25.4541	4.1298	0.9926	$\hat{a} = 322.2684$, $\hat{b} = 0.0364$, $\hat{\alpha} = -0.0213$
M_2	15.4319	2.8263	0.9963	$\hat{a} = 148.9516$, $\hat{b} = 0.2361$, $\hat{\alpha} = -0.0074$, $\hat{\beta} = 3.6741$
M_3	37.9825	4.1728	0.9915	$\hat{N} = 202.0216$, $\hat{a} = 1.7805$, $\hat{b} = 6.7509$, $\hat{\alpha} = 4.3357$, $\hat{\beta} = 14.4469$
M_4	31.5738	4.3562	0.9971	$\hat{a} = 118.4503$, $\hat{b} = 0.2486$, $\hat{\psi} = 0.6511$, $\hat{\alpha} = 0.0037$, $\hat{\beta} = 0.3821$
M_5	13.7593	2.2109	0.9968	$\hat{a} = 124.4731$, $\hat{b} = 0.2226$, $\hat{\psi} = 0.7165$, $\hat{\alpha} = -0.2571$, $\hat{\beta} = 0.1944$, $\hat{k} = 1.4381$
M_6	28.2737	4.0491	0.9936	$\hat{a} = 155.2459$, $\hat{b} = 0.4662$, $\hat{\psi} = 1.2921$, $\hat{\alpha} = 0.0331$, $\hat{\beta} = 0.3417$
M_7	15.4335	3.1593	0.9972	$\hat{N} = 99.4022$, $\hat{b} = 0.3173$, $\hat{\beta} = 3.3173$, $\hat{\psi} = 0.8398$
M_8	13.8120	2.5449	0.9921	$\hat{N} = 55.8739$, $\hat{b} = 0.0.4211$, $\hat{\psi} = 0.4199$
Proposed	9.3885	1.9196	0.9974	$\hat{a_0} = 1.3493$, $\hat{b} = 0.3589$, $\hat{p} = 0.9971$, $\hat{\alpha} = -0.0424$, $\hat{\psi} = 0.1253$

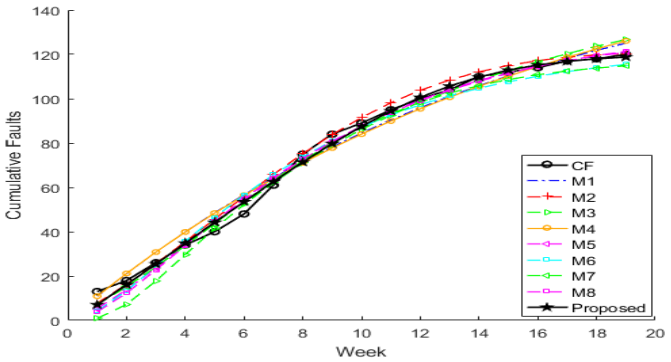

Fig. 2. Performance analysis of models: DS-2

provides a more accurate representation of software reliability improvement over time, as

demonstrated through comparative analysis with widely used NHPP models and evaluation of goodness-of-fit criteria using two datasets. Our model's validation encompasses two datasets. For DS-1, we observe an MSE value of 40.1132, an MAE value of 4.7471, and an R^2 value of 0.9907, surpassing all other models considered in this study. Similarly, for DS-2, we note an MSE value of 9.3885, an MAE value of 1.9196, and an R^2 value of 0.9974, indicating superior performance compared to alternative models. Our research addresses a research gap in existing literature, where FRE is often treated as a static parameter, overlooking its dynamic nature during software development. Moving forward, there is ample scope to enhance our model by incorporating more advanced and robust fault removal efficiency mechanisms. Research can focus on developing sophisticated algorithms that dynamically adapt to changing conditions during the development and testing phases. Moreover, exploring the integration of machine learning and artificial intelligence techniques may further optimize the prediction accuracy and enhance the model's performance.

REFERENCES

1. U. Samal and A. Kumar, "Enhancing software reliability forecasting through a hybrid arima-ann model," *Arabian Journal for Science and Engineering*, 2023, pp. 1–14.
2. U. Samal, S. Kushwaha, and A. Kumar, "A testing-effort based srgm incorporating imperfect debugging and change point," *Reliability: Theory & Applications*, Vol. 18, no. 1 (72), 2023, pp. 86–93.
3. U. Samal and A. Kumar, "Redefining software reliability modeling: embracing fault-dependency, imperfect removal, and maximum fault considerations," *Quality Engineering*, 2023, pp. 1–10.
4. U. Samal and A. Kumar, "A neural network approach for software reliability prediction," *International Journal of Reliability, Quality and Safety Engineering*, 2024.
5. A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE transactions on Reliability*, Vol. 28, no. 3, 1979, pp. 206–211.
6. S. Yamada, M. Ohba, and S. Osaki, "S-shaped software reliability growth models and their applications," *IEEE Transactions on Reliability*, Vol. 33, no. 4, 1984, pp. 289–292.
7. M. Ohba, "Inflection s-shaped software reliability growth model," in *Stochastic Models in Reliability Theory: Proceedings of a Symposium Held in Nagoya, Japan, April 23–24, 1984*. Springer, 1984, pp. 144–162.
8. H. Pham, L. Nordmann, and Z. Zhang, "A general imperfect-software-debugging model with s-shaped fault-detection rate," *IEEE Transactions on reliability*, Vol. 48, no. 2, 1999, pp. 169–175.
9. X. Zhang, X. Teng, and H. Pham, "Considering fault removal efficiency in software reliability assessment," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 33, no. 1, 2003, pp. 114–120.
10. H.-W. Liu, X.-Z. Yang, Q. Feng, and Y.-J. Shu, "A general nhpp software reliability growth model with fault removal efficiency," *Iranian journal of electrical and computer engineering*, Vol. 4, no. 2, 2005, pp. 144–149.

11. Q. Li and H. Pham, "A testing-coverage software reliability model considering fault removal efficiency and error generation," *PloS one*, Vol. 12, no. 7, 2017, p. e0181524.
12. Q. Li and C. Mao, "Considering testing-coverage and fault removal efficiency subject to the random field environments with imperfect debugging in software reliability assessment," in *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2016, pp. 257–263.
13. P. Kapur, A. Gupta, and P. Jha, "Reliability analysis of project and product type software in operational phase incorporating the effect of fault removal efficiency," *International Journal of Reliability, Quality and Safety Engineering*, Vol. 14, no. 03, 2007, pp. 219–240.
14. S. Chatterjee and A. Shukla, "An ideal software release policy for an improved software reliability growth model incorporating imperfect debugging with fault removal efficiency and change point," *Asia-Pacific Journal of Operational Research*, Vol. 34, no. 03, 2017, p. 1740017.
15. S. Chatterjee, A. Shukla, and H. Pham, "Modeling and analysis of software fault detectability and removability with time variant fault exposure ratio, fault removal efficiency, and change point," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, Vol. 233, no. 2, 2019, pp. 246–256.
16. V. Verma, S. Anand, P. Kapur, and A. G. Aggarwal, "Unified framework to assess software reliability and determine optimal release time in presence of fault reduction factor, error generation and fault removal efficiency," *International Journal of System Assurance Engineering and Management*, Vol. 13, no. 5, 2022, pp. 2429–2441.
17. M. A. Haque and N. Ahmad, "A software reliability model using fault removal efficiency," *Journal of Reliability and Statistical Studies*, 2022, pp. 459–472.
18. U. Samal and A. Kumar, "A software reliability model incorporating fault removal efficiency and it's release policy," *Computational Statistics*, 2023, pp. 1–19.
19. P. Kapur, H. Pham, S. Anand, and K. Yadav, "A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation," *IEEE Transactions on Reliability*, Vol. 60, no. 1, 2011, pp. 331–340.
20. P. Roy, G. Mahapatra, and K. Dey, "An nhpp software reliability growth model with imperfect debugging and error generation," *International Journal of Reliability, Quality and Safety Engineering*, Vol. 21, no. 02, 2014, p. 1450008.
21. K. Y. Song, I. H. Chang, and H. Pham, "Nhpp software reliability model with inflection factor of the fault detection rate considering the uncertainty of software operating environments and predictive analysis," *Symmetry*, Vol. 11, no. 4, 2019, p. 521.
22. M. Xie, Q. Hu, Y. Wu, and S. H. Ng, "A study of the modeling and analysis of software fault-detection and fault-correction processes," *Quality and Reliability Engineering International*, Vol. 23, no. 4, 2007, pp. 459–470.
23. A. Wood, "Software reliability growth models," *Tandem technical report*, Vol. 96, no. 130056, 1996, p. 900.

Umashankar Samal graduated with a master's degree in Mathematics in 2019 from Sant Longowal Institute of Engineering & Technology, India. Currently, he is a research scholar at Atal Bihari Vajpayee-Indian Institute of Information Technology and Management, Gwalior, India. His research interests include safety, quality, and reliability engineering.

Ajay Kumar joined ABV-IIITM, Gwalior in July 2009 and now he is an associate professor at Department of Engineering Sciences, ABV-IIITM, Gwalior. His primary areas of interest are Reliability, Statistics, Fuzzy Sets, Fuzzy Logic, Optimization, Machine Learning, and Modeling & Simulation. He has published over 45 research papers in reputed journals and conferences.