IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 13, No. 3, March, 2026, pp. 595-609
Journal homepage: http://iieta.org/journals/mmep

A Non-Homogeneous Poisson Process Software Reliability Growth Model with Imperfect Debugging via Hybrid Neural Network–Crow Optimization Parameter Estimation

Entesar K. Jasim[1], Hasanain J. Neamah Alsaedi[2], Safaa J. Alwan[3], Mohammad A. Tashtoush[4,5*], Adel S. Hussain[6,7,8]

[1] Statistics Department, College of Management and Economics, University of Mustansiriyah, Baghdad 10001, Iraq
[2] Business Information Technology Department, College of Business Information, University Information Technology and Communications, Baghdad 18310, Iraq
[3] Electronic Computer Center, University of Information Technology and Communications, Baghdad 18310, Iraq
[4] Department of Basic Sciences, AL-Huson University College, AL-Balqa Applied University, Salt 19117, Jordan
[5] Faculty of Education and Arts, Sohar University, Sohar 311, Sultanate of Oman
[6] IT Department, Amedi Technical Institutes, University of Duhok Polytechnic, Duhok 1006, Iraq
[7] Department of Administrative and Financial Affairs, Faculty Division, Al-Iraqi University, Baghdad 10071, Iraq
[8] Department of Computer Engineering, Al-Kitab University, Kirkuk 36001, Iraq

Corresponding Author Email: tashtoushzz@su.edu.om

https://doi.org/10.18280/mmep.130314

Received: 11 January 2026
Revised: 14 March 2026
Accepted: 24 March 2026
Available online: 10 April 2026

Keywords:
software reliability growth model, Non-Homogeneous Poisson Process, imperfect debugging, error reduction factor, feedforward neural network, Crow Optimization Algorithm

ABSTRACT

Software reliability is essential for ensuring dependable systems, but the current Non-Homogeneous Poisson Process (NHPP)-based software reliability growth models (SRGMs) have limited generalizability, high sensitivity to datasets, and insufficient fault-prediction accuracy. These shortcomings make them less effective in real-world software testing and maintenance. This study proposes a new NHPP-based SRGM with an error reduction factor to improve the dynamics of fault detection and imperfections of debugging. The proposed approach combines a Feedforward Artificial Neural Network (FFANN) and the Crow Optimization Algorithm (COA) to estimate model parameters. Such a hybrid approach allows better modeling of error detection and correction rates, as well as the possibility of introducing new errors as debugging progresses. This model was evaluated using simulation results and benchmark datasets with different sample sizes, and its performance was assessed via various statistical measures, including Root Mean Squared Error (RMSE), Akaike Information Criterion (AIC), and reliability curves. The findings indicate that the proposed model performs better than SRGMs, with improved predictive capabilities for short-term fault identification and long-term reliability prediction. The study contributes to the reliability testing of complex software systems by addressing major shortcomings of the previous models. It provides a robust and flexible model to predict the behavior of failures, thereby supporting safer and more effective software development and maintenance patterns.

1. INTRODUCTION

One of the foundations of reliable system design is software reliability since malfunctions in the course of operation may result in dramatic economic and safety repercussions. Software reliability growth models (SRGMs) have also been extensively used to predict and deal with these risks, especially the Non-Homogeneous Poisson Process (NHPP). These models estimate the cumulative fault discovery process as time goes by and consider two big questions: (i) the average time that it takes to uncover a certain number of faults, and (ii) the dependability of the software at a specified number of usage periods. Due to their adaptability and their ability to be applied to a wide variety of datasets, NHPP-based SRGMs have taken on a central role in software reliability analysis [1-5].

Although they are important, there are severe limitations with the current SRGMs. Most of the models are very sensitive to the nature of particular data sets, and this limits their applicability in different systems. Others make restrictive assumptions, such as having perfect debugging, fixed detection rates, or fixed fault profiles. As a matter of fact, the practice of debugging is rarely based on these assumptions: the new errors can be introduced when fixing the fault, the rate of detection can decline with time, and there are learning effects that have been proposed in S-shaped models that are not always seen in the industrial setting. Consequently, traditional NHPP-driven strategies tend to underestimate the actual dynamics of fault detection and correction and do not have a high level of predictability [6-14]. Table 1 shows the mean value function (MVF) for the suggested model and compares it with other existing NHPP-SRGM models.

Table 1. Mean value function (MVF) of the proposed model and its comparison with existing non-homogeneous Poisson process (NHPP) software reliability growth models

Model	$m(t)$
Goel-Okumoto (GO)	$a(1-e^{-bt}),\ a(t)=a;\ b(t)=b$
Delayed Shape (DS)	$a(1-(1+bt)e^{-bt}),\ a(t)=a;\ b(t)=\frac{b^2t}{1+bt}$
Inflection Shape (IS)	$\frac{a(1-e^{-bt})}{1+\beta e^{-bt}},\ a(t)=a;\ b(t)=\frac{b}{1+\beta e^{-bt}}$
Hossain-Dahiya/ Goel-Okumoto (H-D/G-O)	$\log\left[\frac{(e^a-c)}{(e^{ae^{-bt}}-c)}\right]$
Yamada Exponent (YE)	$a(1-e^{-\gamma\alpha(1-e^{-bt})}),\ a(t)=a;\ b(t)=r\alpha\beta e^{-\beta t}$
Yamada Rayleigh (YR)	$a(1-e^{-\gamma\alpha(1-e^{-bt^2/2})}),\ a(t)=a;\ b(t)=r\alpha\beta t e^{\frac{-\beta t^2}{2}}$
Yamada Imperfect (YI-D1)	$\frac{ab}{b-a}(e^{-at}-e^{-bt}),\ a=ae^{\alpha t};\ b(t)=b$
Yamada Imperfect (YI-D2)	$a(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha at,\ a(t)=a(1+\alpha t);\ b(t)=b$
Pham-Nordmann-Zhang (P-N-Z)	$\frac{a}{1+\beta e^{-bt}}\left[(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha t\right],\ a(t)=a(1+\alpha t);\ b(t)=\frac{b}{1+\beta e^{-bt}}$
Huang-Kuo-Lyu (H-K-L)	$\frac{a}{1-\beta}\left[1-e^{-b(1-\beta)w^*(t)}\right],\ a(t)=a+\beta n(t);\ b(t)=b$

Within recent years, researchers have started to discuss the adoption of machine learning and optimization algorithms to improve the predictive power of SRGMs [1, 2, 5]. Classical NHPP-based models usually make hard assumptions like the uniform rates of fault detection or ideal debugging; thus, they cannot model complex nonlinear dynamics that are present in the modern software testing space [9, 11]. These limitations have been overcome recently through the use of smart optimization and learning-based methods. As an illustration, an imperfect debugging NHPP model is offered, a model that takes into account the dynamic fault detection behavior and enhances the reliability estimation in practice under testing conditions [1]. On the same note, the team proposed a generalized NHPP framework that can model the nonlinear fault detection process and enhance the accuracy of predictions in large-scale software systems [2].

The other research direction that is of great importance is the utilization of machine learning and deep learning in estimating parameters and predicting failures. SRGMs based on neural networks have demonstrated a great ability to model nonlinear associations amid failure and testing time [15-18]. Such approaches make use of less rigorous distributional assumptions and enhance extrapolation among datasets. Moreover, recent efforts have investigated hybrid methods that merge neural networks and metaheuristic parameter optimization algorithms in order to improve local learning ability and global parameter search [19]. Indicatively, hybrid models based on neural networks and swarm intelligence algorithms have been suggested to enhance accuracy in the estimation of parameters and solve problems of local minima in the training of a model [20]. These methods indicate the increasing significance of integrating artificial intelligence methods and conventional reliability modeling approaches. In spite of these developments, most of the currently available models are still plagued by issues of computational complexity, sensitivity to the properties of the datasets, and limited capability to capture the dynamic interaction between fault detection, correction, and error introduction [1, 2, 5]. Thus, the current research suggests a hybrid NHPP-based SRGM, which combines Feedforward Artificial Neural Network (FFANN) and COA. The nonlinear learning property of neural networks and the exploration power of metaheuristic optimization more effectively combine the global exploration power of metaheuristic optimization with the nonlinear learning capability of neural networks to give more stable and accurate parameter estimation.

The given framework builds on the recent progress in the domain of intelligent reliability modeling and adds to the current research trend of applying machine learning methods and combining them with classical software reliability theory [21]. These failures ensure the existence of a research gap: the reliability models that account for a realistic manner imperfect debugging, error introduction, and error reduction are much needed. Moreover, the conventional approaches to the estimation of parameters are not always robust, which diminishes the practical utility of the models.

This study aims to resolve these problems by suggesting a more advanced NHPP-based SRGM, which uses an error reduction factor in exponential form to reflect the dynamics of faults. To determine the model parameters, the method combines FFANN with the Crow Optimization Algorithm (COA), which allows addressing the nonlinear dependencies effectively in the failure data. The simulations are used to verify the proposed model alongside various benchmark datasets, and its performance is measured by some important statistical measures like Root Mean Squared Error (RMSE) and Akaike Information Criterion (AIC). Findings support the fact that the model is always superior to traditional SRGMs in terms of both fitting performance and predictive performance.

In short, the study provides a stronger and more flexible system of software reliability analysis. It helps to develop software and maintain it more effectively, safely, and in a cost-efficient manner because it helps to overcome the drawbacks of the traditional models. To formulate criteria for evaluating the goodness of fit of SRGM, the entropy principles were incorporated with existing measurements. Advanced reliability concepts for hardware and software systems were described by references [22-25], including maintenance strategies; multi-criteria decision-making techniques for analysing SRGMs were discussed by reference [26]. Recent research has also employed machine learning and deep learning for software reliability analysis [27-30].

To ensure the topic of SRGM is explored to the fullest, the study is divided into several sections. The introduction section appreciates the role of software reliability and establishes the fact that existing models have their drawbacks. The methodology section then presents the NHPP-based model that is proposed and explains how the COA is used for parameter estimation. The data obtained from the model evaluation are described in the results section, where

comparisons with other methods are made to show the advantage of the proposed technique. Finally, the discussion section brings together the outcomes of the study and the direction for future research, and concludes that the idea of the presented approach can effectively contribute to the increase in software reliability.

2. CROW OPTIMIZATION ALGORITHM

The COA is a metaheuristic based on population that is aimed at solving nonlinear optimization problems. In comparison to the early swarm-based algorithms, including Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), COA adds to the search space a memory mechanism that improves exploratory and exploitative search. Every candidate solution is indicated as a parameter space vector. At iteration t, the position of the i^{th} solution is denoted as $\boldsymbol{x}_i^t \in \mathbb{R}^d$, where d is the number of parameters to be optimized. Each solution retains a memory $\boldsymbol{m}_i^t$, which corresponds to its historically best position [31]. The update rule is defined as:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{x}_i^t + r \cdot f_l \cdot \left(\boldsymbol{m}_j^t - \boldsymbol{x}_i^t\right), & \text{if } r \geq AP_j, \\ \text{random position in search space}, & \text{otherwise,} \end{cases}$$

where, $r \sim U(0,1)$ is a random number, $f_l > 0$ is the flight length parameter controlling step size, $AP_j \in [0,1]$ is the awareness probability of solution j, $\boldsymbol{m}_j^t$ is the best-known solution of a randomly chosen peer j. Algorithm 1 shown below outlines the main procedure of the COA, and its Pseudocode represent the initializes a population of crows, assigns memory to each solution, and iteratively updates positions based on a random peer selection and awareness probability.

Algorithm 1. Crow Optimization Algorithm (COA)

Input: Population size n, dimension d, flight length f_l, and awareness probability AP, maximum iterations T

Output: Best solution found x^*

1. Initialize population $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ within bounds.
2. Set memory $M = X$ (each solution's best-known position).
3. For $t = 1$ to T:
   (a) For each solution i:
   - Randomly select peer $j \neq i$.
   - Generate random number $r \sim U(0,1)$.
   - If $r \geq AP_j$: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + rand \cdot f_l \cdot \left(\boldsymbol{m}_j - \boldsymbol{x}_i\right)$

   else: reinitialize $\mathbf{x}_i$ randomly.
   - Evaluate fitness $f(\mathbf{x}_i)$.
   - Update memory: if $f(\mathbf{x}_i)$ is better than $f(\mathbf{m}_i)$, set $\mathbf{m}_i \leftarrow \mathbf{x}_i$.
4. End For
5. Return best memory $\mathbf{x}^* = \arg\min f(\mathbf{m}_i)$.

Pseudocode for Crow Optimization (CO)

```
Initialize population X = {x_1, x_2, ..., x_n} within search bounds
Initialize memory M = X (best known positions)
Repeat until stopping criterion:
   For each solution i:
      Randomly select a peer j ≠ i
      Generate random number r ∈ [0,1]
      If r ≥ AP_j:
         x_i ← x_i + rand · f_l · (m_j − x_i)
      Else:
         x_i ← random position in search space
      Evaluate fitness f(x_i)
      If f(x_i) is better than f(m_i), set m_i ← x_i
   Update the global best solution
Return the best solution found
```

3. FEED FORWARD ARTIFICIAL NEURAL NETWORKS

FFANN has a general nonlinear mapping structure, which is best adapted to estimating parameters in SRGMs. Because the failure process in NHPP assumptions tends to have nonlinear relationships between error detection, error correction, and introduction, standard estimation techniques can hardly handle such relationships. FFANNs address this weakness by acquiring input-output mappings with complex input-output mappings directly by using data, and thus offer powerful parameter estimates with weak distributional assumptions [32].

To prevent the overfitting of a standard risk during the training of neural networks, a number of strategies were used. The normalization of the data was performed before the training, and the dataset was divided into the training and validation subsets in order to guarantee generalization. The Mean Squared Error (MSE)-based early stopping criteria were used, which stopped training once it began to level, and regularization mechanisms were also present to limit the magnitude of weights. All these actions decrease the chances of memorizing the training set, which guarantees consistent estimates between sample sizes [33].

The selected network architecture is a three-layer FFANN, the structure of which is 6-9-1: six neurons in the first hidden layer, nine neurons in the second, and one output neuron. This arrangement was arrived at after making comparative experiments with other design configurations, striking a balance between the complexity of the model and predictive ability. In capturing nonlinear dependencies of SRGM parameters, the architecture was found to be superior to other architectures in the sense that it was computationally efficient. The output neuron is a single function that is directly proportional to the predicted reliability growth function, which makes the network's output aligned with the goal of the SRGM estimation. The limitations of FFANNs are listed as follows:

- Fixed input and output size: A weakness of FFANNs lies in the predictability of the number of input and output neurons, which makes it rigid for tasks dealing with sequences of variable lengths.
- Overfitting: Because of having numerous neurons and layers, the FFANNs have a propensity to memorize the training data and fail to generalize well where there is no abundant data or where further regularization measures have not been used.
- No memory: There is no method in which FFANNs can hold information about past input, thus non-functional for any task where temporal progression or data sequence is relevant (for which, recurrent neural nets are often used).

4. THE NON-HOMOGENEOUS POISSON PROCESS FRAMEWORK AND ITS ASSOCIATED KEY TASKS

NHPP gives a well-established method for software failure process modeling. A generalized NHPP formulation, which clearly considers imperfect debugging, error introduction, and error reduction, was taken. The reliability function is modeled in the form of a cumulative intensity function, and the failure MVF is the center of the parameter estimation. To enhance the readability, the key modeling assumptions are discussed in separate subsections [34].

Assumption 1: Error reduction

Error correction efficiency generally declines as the number of errors detected increases because of interdependence, human factors, and organizational limitations. This is modeled by the exponential reduction error factor. The errors that are corrected at time t are:

$$m(t) = m_0 \exp(-\beta t) \tag{1}$$

where, m_0 is the initial correction rate and β is the reduction coefficient.

Assumption 2: Error introduction during debugging

Correction of faults can also cause new errors, especially in complicated systems. This is modeled as a rate of error generation that depends on the current activity of debugging:

$$\frac{dx(t)}{dt} = \gamma m(t) \tag{2}$$

where, γ is the error introduction coefficient.

Assumption 3: Per-error fault identification rate

The probability of finding a certain fault reduces with time as the remaining faults are more difficult to find. This is modelled as a decaying rate of detection exponentially:

$$a(t) = a_0 \exp(-\delta t) \tag{3}$$

where, a_0 is the initial detection rate and δ is the decay constant.

The generalized NHPP-based SRGM, which is a combination of these assumptions, is subjected to the system of differential Eqs. (5)-(10). To make them readable, the main text of the resulting MVF is given:

$$M(t) = f(a(t), x(t), m(t)) \tag{4}$$

4.1 Utilizing a new method to analyze software reliability growth models

In the NHPP framework, the derivation of the traditional failure MVF is based on four key assumptions: Failure model assumptions are (a) each failure indicates a single fault; (b) the failure rate is constant in terms of faults; (c) if a failure is identified then the corresponding fault is repaired and deselected; (d) the rate of failure identification increases along with the number of unidentified faults in the software with a constant of proportionality. Additionally, the generalized imperfect debugging NHPP model incorporates three more assumptions: Some of these are: (i) The proportion of failure detection, above mentioned, is postulated as depending on time, τ, i.e., $k(\tau)$; (ii) By correction of read errors, other errors may be produced in the process or is dependent of time t. Other assumptions made in the context of this research include the following: (iii) Error correction is noisy, that is, every error that has been found up to the time τ is corrected probabilistically. The generalized NHPP model for imperfect debugging, which incorporates the efficiency of error removal, is formulated as follows [18]:

$$m'(\tau) = k(\tau)[a(\tau) - Qm(\tau)] \tag{5}$$

where, Q is the measurement of the likelihood of error correction, while (t) is the number of errors that have been corrected by the time (t). For generalization, the newly proposed SRGM is developed starting from the differential equation below:

$$m'(\tau) = k(\tau)[a(\tau) - x(\tau)] \tag{6}$$

In particular, it is acknowledged that the actual count of corrected errors (t), depends directly on the error reduction coefficient [20]. Using the basis given in Eq. (6), we develop an improved model that involves additional assumptions.

4.1.1 Mathematical derivation of the proposed Non-Homogeneous Poisson Process model

In order to enhance the interpretability of the proposed SRGM, the detailed description in this section follows the step-by-step mathematical formulation, finally arriving at the desired MVF. Where $m(t)$ refers to the MVF, which is the average count or number of software faults that will be detected up to time t. Within the context of an NHPP, the fault detection rate is the derivative of the MVF:

$$\lambda(t) = \frac{dm(t)}{dt} \tag{7}$$

where, $\lambda(t)$ represents the failure intensity function, describing how quickly faults are detected during testing. The productivity of error correcting in the practical debugging processes declines with an increase in the number of faults found. The model of this effect is an exponential error reduction factor:

$$\eta(t) = \eta_0 e^{-\alpha m(t)} \tag{8}$$

where, η_0 represents the initial correction efficiency, and α denotes the reduction coefficient controlling how rapidly the correction efficiency declines.

This formulation reflects the increasing complexity of remaining faults as testing progresses. Under realistic debugging conditions, it is possible that fixing a identified fault will create new errors. Let gamma represent the coefficient of error introduction, which is the percentage of error induced in the process of modification activities. This is because both the fault elimination and the fault creation processes affect the net change in the number of residual faults. The other useful learning in software testing is that as time goes by, the likelihood of the software identifiers finding the remaining faults also reduces. This is the case because simpler faults are normally identified at an earlier period, and then later faults are harder to detect. The detection rate per fault can thus be represented as an exponential type of function:

$$\beta(t) = \beta_0 e^{-\delta t} \tag{9}$$

where, β_0 is the initial detection rate, and δ represents the

decay constant of detection efficiency.

Combining the above mechanisms, the rate of fault detection in the system depends on three factors:

- the remaining number of faults,
- the efficiency of debugging, and
- the time-dependent detection capability.

Thus, the generalized NHPP differential equation describing the failure process can be expressed as:

$$\frac{dm(t)}{dt} = \beta(t)(a - m(t))\eta(t) \tag{10}$$

where, a represents the number of inherent faults in the software system. The replacement of the expressions of $\beta(t)$ and $\eta(t)$ into the equation leads to a nonlinear differential model of fault detection, correction, and introduction dynamics. The solution of this equation, under the first condition.

4.1.2 Three important assumptions

(i) Modeling assumptions for exponential error reduction

In this framework, the error reduction factor serves as a governing parameter that constrains the differential formulation of the cumulative number of observed failures t and correctable errors t. In other words, the effectiveness of the following error correction steps may reduce with the number of discovered errors in the actual debugging processes. This assumption is premised on three sources that stem from a reduction in interdependencies among errors, the mental inertia associated with focusing on the current task, and the instability within the debugging team. First, modifying errors may be influenced in such a manner; for instance, a particular error may be associated with certain other removed errors to reduce the information regarding the rest of the sections that are available for modification in due course. Second, the currently induced mental set may interfere with the ability of the corrector to consider other relations between the errors. Such occurrences lead to the notion that other related mistakes are not noticed by the corrector and hence cannot be changed using the modified mistake. Last but not least, the average level of correction and relevant experience of the correctors may deteriorate owing to some unpredictable circumstances, for example, transfers and replacements. In many software companies nowadays, skilled testers or correctors can be transferred or dismissed, as engineering needs appear. When successors are not engaged in programming, their inexperience can hinder the efficiency of correcting errors in the early stages of change. As such, instability in the debugging team can greatly reduce its efficiency in correcting errors.

Since the effectiveness of error correction may decrease with the increase of the number of identified errors, the error reduction factor may be defined as the constrained relationship between the number of detected errors and the error reduction factor, which is modeled as a decreasing exponential function of (τ). This formulation is derived from the defined characteristics of the error reduction factor. And the number of modified errors (τ) can be mathematically represented as [20]:

$$\frac{x'(\tau)}{m'(\tau)} = qe^{-\beta m(\tau)} \tag{11}$$

where, q denotes the first value of this parameter, which is the first error reduction factor, describing the ratio of the initial rate of error correction to the initial rate of error detection, while β is the reduction coefficient that defines the declining curve of the rate. Thus, the expression of error reduction stated in Eq. (7) is quite stable.

(ii) Rate of error introduction

Here, the aim is to develop a constrained regression model between the total error count and the parameters that guide it (t), and the modified error count (t) which will be used to find the dependency of the detected error count (t). Turning to the problem of new error incorporation, one should figure out that new errors occur not during the error detection stage but during the process of modifying existing ones. In general, it has been discovered that no matter how successful modification attempts are during the debugging process, the probability of creating new errors rises simultaneously with the number of attempts. On some occasions, it may require multiple debugging to complete fix this problem, which increases this risk. Because the occurrence of new errors coincides with the continuous modification of errors, it is reasonable to express the change rate of residual errors through a dynamic process covering both the generation of new errors and the modification process of errors. The differential relationship between the total number of errors $a(t)$ and the modified number of errors $x(t)$ can be expressed as follows:

$$\frac{a'(\tau)}{[a(\tau) - x(\tau)]'} = -p \tag{12}$$

Eq. (8) indicates that as the intensity of generalized modification increases, the likelihood of the error input rate also rises.

(iii) Per-error fault identification rate

In the current part, we are supposing that the detection of errors is possible in every fault and is a linearly fading exponential output of t. Notably, the detection rate by no means is assumed to be equal among all remaining faults of the software. With such an assumption in place, we assume that the background defects are discovered at the same discovery rate at any particular moment in time. However, as it is known, the detection efficiency per fault also tends to decrease with time. Several factors can be attributed to this fadeaway behavior, and they are mentioned below:

1. This is illustrated in section 1, where most of the studies pointed out that the so-called 'learning' process does not normally happen in real-life industrial and testing scenarios, due to a lack of resources and non-operational skills.
2. Since there are only a finite number of detection conditions and methods, and since there is often very little testing experience, it becomes increasingly difficult to uncover latent errors resident in large, complicated programs or abstract data types as time passes.
3. There is potential for degradation in the effectiveness of fault detection because of the large amount of work required for recording, testing, and analyzing program behavior, which can ultimately reduce the reliability of testers. To enhance the flexibility of our model, it is assumed that the per-error fault detection rate decreases exponentially with time τ and can be modeled as follows:

$$k(\tau) = be^{-(a+\alpha\tau)} \tag{13}$$

where, b is the initial value of fault detection rate per error, and α defines the decay factor.

4.1.3 Practical validity of the modeling assumptions

To further support the assumptions underlying the proposed NHPP modeling, it is worthwhile to align them with empirical findings from real-world software testing environments. First, the assumption of error reduction reflects a practical phenomenon: debugging efficiency declines over time. Easily identified faults are eliminated early in testing, while those that remain become more complex and intertwined as testing progresses. Extensive empirical research on industrial software development projects has documented a decline in fault removal efficiency as testing matures, especially in large systems where code dependencies make fault fixing more challenging. Second, the introduction of errors during debugging is a well-documented phenomenon in software engineering practice. Because developers modify existing code to fix identified faults, changes can inadvertently create new defects when the developer does not fully understand how the systems interact or when sufficient regression testing is not performed. Large-scale software maintenance experiments have shown that corrective maintenance tends to introduce between 5–20 percent more faults, depending on system complexity and the developer's experience. Thus, representing the debugging process as a dynamic system that balances error removal with error introduction can better capture the evolution of software. Third, the assumption of a time-dependent fault detection rate aligns with the observation that fault detection efficiency follows a downward trend. In real-world testing, the most apparent errors are identified early, while the remaining errors are often more subtle and more difficult to trace. This pattern has been observed in numerous empirical datasets, including the Tandem Computers failure dataset and IEEE reliability benchmark datasets, where the fault discovery rate steadily decreases with testing. The validity of these assumptions is further supported by the simulation experiments and case studies presented in this work, such as the Tandem Computer Company dataset and the IEEE Std 1633 dataset. These realistic assumptions indicate that the proposed NHPP model fits better and has greater forecasting power under these conditions, confirming that the modeling framework captures key features of real-world software reliability growth.

4.2 An innovative software reliability growth model and its failure mean value function

4.2.1 An advanced software reliability growth model

Given the above-noted assumptions, the sustainable human resource management (SHRM) model, adapted as an SRGM, has been formulated as follows:

$$\begin{cases} m'(\tau) = k(\tau)[a(\tau) - x(\tau)] \\ \dfrac{x'(\tau)}{m'(\tau)} = qe^{-\beta m(\tau)} \\ \dfrac{a'(\tau)}{[a(\tau) - x(\tau)]'} = -p \\ k(\tau) = be^{-(a+\alpha\tau)} \end{cases} \tag{14}$$

Through solving Eq. (14) simultaneously under some initial conditions, one can obtain the corresponding solution, namely the MVF $m(t)$. In this case, to articulate the explicit failure MVF, the following theorem is presented:

Remark 1: Assuming that the MVF of failure, the dynamics of fault detection and removal, and the rate of error introduction assume the functional relationships as given in Eq. (10), and under the initial conditions $m(0) = 0, x(0) = 0$, and $b(0) = b_0$, the following form of the MVF of failure is obtained:

$$m(t) = \frac{1}{\gamma}\ln\left[\frac{\gamma b_0 e^{\left[\gamma b_0 - q(1-p)*\frac{b}{\alpha}e^{-a}[1-e^{-\alpha t}]\right]} - q(1-p)}{\gamma b_0 - q(1-p)}\right] \tag{15}$$

Eq. (15) defines the full mean value, which incorporates nonlinear fault detection, correction, and introduction of behaviors during testing.

4.2.2 Further analysis of failure mechanisms in the mean value function

The section gets more into the dynamics of failure that is governed by MVF and provides major theoretical derivations to back the properties of the structural model.

- In Eq. (15), once the time t goes to infinity, then $m(t)$ will converge to an upper bound, i.e.,

$$\lim_{t\to\infty} m(t) = \frac{1}{\gamma}\ln\left[\frac{\gamma b_0 e^{\left[\gamma b_0 - q(1-p)*\frac{b}{\alpha}e^{-a}[1-e^{-\alpha t}]\right]} - q(1-p)}{\gamma b_0 - q(1-p)}\right] \tag{16}$$

The asymptotic fault bound is expressed in Eq. (16), demonstrating that there is a convergence to maximum reliability.

- Under the assumed initial conditions $m(0) = 0$ and $x(0) = 0$, an identity $x(t) = \frac{q}{\gamma}\left[1 - e^{-\gamma m(t)}\right]$ follows directly from the differentiation of Eq. (10). (ii) Inserting Eq. (15) into the corresponding formulation leads to the expression for the number of corrected errors, presented in Eq. (17).

$$x(t) = \frac{q}{\beta}\left[1 - \frac{\gamma b_0 - q(1-p)}{\gamma b_0 e^{\left[\gamma b_0 - q(1-p)*\frac{b}{\alpha}e^{-a}\right]} - q(1-p)}\right] \tag{17}$$

- Under the initial condition $b(0) = b_0$, substituting Eq. (17) into Eq. (14), the total error number is presented:

$$\begin{aligned} b(t) &= b_0 + px(t) \\ &= a_0 + \frac{pq}{\gamma}\left[1 - \frac{\gamma b_0 - q(1-p)}{\gamma b_0 e^{\left[\gamma b_0 - q(1-p)*\frac{b}{\alpha}e^{-a}\right]} - q(1-p)}\right] \end{aligned} \tag{18}$$

Eqs. (17) and (18) explain how the corrected error and the total error change with time and explain why there is a tradeoff between detection and new-fault creation.

- If $\alpha \to 0$ (Implication of this is that fault detection rate per error is almost fixed. It is therefore possible to rewrite Eq. (15) as given below:

$$m(t) = \frac{1}{\gamma}\ln\left[\frac{\gamma b_0 e^{\left[\gamma b_0 - q(1-p)*\frac{b}{\alpha}e^{-a}[1-e^{-\alpha t}]\right]} - q(1-p)}{\gamma b_0 - q(1-p)}\right] \tag{19}$$

- In Eq. (19), if $p \to 0$, it shows that the error creation rate is in effect inexistent. In turn, Eq. (20) could be rewritten in the following way:

$$m(t) = \frac{1}{\gamma}\ln\left[\frac{\gamma b_0 e^{[\gamma b_0 - q * b e^{-a}t]} - q}{\gamma b_0 - q}\right] \quad (20)$$

Under the conditions of $b \to 0$ and $p \to 0$, the limit $\beta a_0 \to q$ is satisfied based on Eq. (14). In Eq. (20), if $b \to 1, \gamma b_0 \to q$, a general expression can be obtained:

$$m(t) = \frac{1}{\gamma}\ln[1 + \gamma b_0 e^{-a}t] \quad (21)$$

Eqs. (19)-(21) give simplified or limiting cases of the equations based on particular conditions of operation (detect rate is constant, fault introduction is negligible or the conditions are symmetric) and provide useful approximations to several testing conditions.

5. ESTIMATION PARAMETERS

This section is aimed at justifying the selection of FFANN and COA in the context of parameter estimation of the proposed NHPP-based SRGM. Although the mathematical aspects of gradient descent and metaheuristic strategies are not novel, here we are concerned with their efficacy, convergence properties, and mutually exclusive use towards the estimation of nonlinear parameters that relate to software failure dynamics.

5.1 Parameter estimation using a modified Feed-forward Artificial Neural Network

This section outlines the initialization and training process of an FFANN to estimate the parameters of the proposed model. The procedure is summarized as follows:

(i) Initialization:

- Input data (t) is prepared and normalized. Each input represents a feature fed into the input layer neurons.
- Network parameters or weights are initialized randomly, and, in most cases, the initial Vault values are assigned according to a uniform distribution. These are the synapses and the options that can exist between neurons, as well as the potential for changing it.

(ii) Network architecture:

- A multilayer FFANN is employed, with two hidden layers: the first containing six nodes and the second nine nodes. This configuration was determined experimentally to outperform alternatives with one or three hidden layers.
- The network's output layer contains a single neuron, predicting $n(t)$.

(iii) Forward propagation:

- Every neuron uses its input, multiplies it by some weights, adds a bias, and applies an activation function $n(t)$. Inputs from the first layer go as inputs to the second, third, and finally to the final layer that will give an output.

(iv) Loss function:

- The MSE between the predicted $(\hat{n}(t))$ and observed $(n(t))$ values is calculated: $MSE = \sum_{i=1}^{n}(y_i - \hat{n}(t_i))^2$.
- A lower MSE indicates better model performance.

(v) Training process:

- The network iteratively updates weights and biases using backpropagation and optimization algorithms.
- Gradient descent or advanced methods (e.g., inverse Hessian-based updates) are used to minimize the loss. The updates follow:

$$r_{k\,new} = r_{k\,old} - a{M_k}^{-1} g_k$$
$$c_{k\,new} = c_{k\,old} - a{M_k}^{-1} g_k$$

where, g_k is the gradient, ${M_k}^{-1}$ is the inverse Hessian matrix, r is the weight, and c is the bias.

(vi) Layer-specific updates:

- Weights and biases in the first and second hidden layers are updated iteratively, ensuring convergence:

$$w_{k\,new} = w_{k\,old} - [{L_k}^T L_k + bI]^{-1} g_k$$
$$c_{k\,new} = c_{k\,old} - [{L_k}^T L_k + bI]^{-1} g_k$$

where, w is the weight of the hidden layer, and c is the bias.

(vii) Stopping criteria:

- Training stops when the MSE reduces below a predefined threshold (ϵ) or, after a maximum number of iterations.
- If $MSE_{new} \leq MSE_{old}$, parameters are refined; otherwise, adjustments are made to the optimization step size.

(viii) Output:

- Final weights, biases, and the trained model are used for parameter estimation and prediction.

5.2 Parameter estimation using the Crow Optimization algorithm

This section presents the initialization process of the COA for estimating the parameters of the proposed model. The procedure is summarized as follows:

(i) Initialization:

- Define the search space for each parameter $(\beta, a_0, q, p, b, \alpha, a)$.
- Initialize a population of crows (solutions) randomly within the parameter bounds.
- Assign a memory to each crow to store its best position (best parameter set).

(ii) Fitness Evaluation: For each crow, calculate the fitness using the given equation (t) by comparing the predicted values with observed data (e.g., using MSE or RMSE as the fitness function).

(iii) Movement: Each crow moves based on Eq. (1).

(iv) Memory Update: Update the memory M_i if the new position $x_i(t+1)$ is better (lower fitness value).

(v) Stopping Criterion: Repeat the movement and memory update steps until the maximum number of iterations is reached or convergence is achieved.

(vi) Best Solution: The crow with the best fitness value holds the optimal parameter estimates.

5.3 Rationale for Feed-Forward Artificial Neural Network and Crow Optimization in Non-Homogeneous Poisson Process parameter estimation

NHPP-based SRGMs have a parameter estimation problem that is nonlinear in nature owing to imperfect debugging, error

introduction, and terms of exponential fault reduction. These nonlinearities are not always captured by conventional methods of analysis or the least-squares methods. FFANN is a mapping of a given data on observed failures into an underlying model parameter, which approximates the nonlinear MVF without restrictive distributional assumptions. The ability to learn at a given level renders FFANN especially useful when the correlation between cumulative failures and time spent on test is complicated or nonmonotonic. On the other hand, the COA provides good global optimization, in effect escaping local minima, which can confound gradient-based algorithms. COA can be particularly useful when the parameter space is multimodal in character (i.e., when FFANN parameters must be initialized) or when global solutions must be fine-tuned. Therefore, a combination of FFANN and COA will utilize both local accuracy (through FFANN) and global searching (through COA), resulting in strong and precise parameter estimates of models based on NHPP.

5.4 Comparative analysis: Feed-Forward Artificial Neural Network vs. Crow Optimization

Table 2 oversimplifies the comparative performance features of FFANN and COA that were witnessed in the course of the experiment to showcase their respective strengths.

Table 2. Comparative analysis of parameter estimation techniques

Criteria	FFANN	Crow Optimization (CO)
Nature of method	Deterministic, gradient-based learning	Stochastic, population-based metaheuristic
Exploration capability	Moderate (local minima possible)	High (strong global search ability)
Convergence speed	Fast once near optimum (≈ 38.5 s/run)	Slower due to population updates (≈ 52.3 s/run)
Parameter accuracy (RMSE)	High (low variance across runs)	Moderate; accuracy depends on population size and awareness probability
Computational cost	Low-to-moderate (requires multiple epochs)	Higher (requires several iterations and population evaluations)
Robustness to initialization	Sensitive to initial weights	Less sensitive due to random population initialization
Best application context	Fine-tuning and precise local optimization	Global search and parameter initialization
Overall, the role in the hybrid model	Provides accurate parameter estimation	Enhances global exploration and solution diversity

Note: FFANN: Feed-Forward Artificial Neural Network; RMSE: Root Mean Squared Error.

5.5 Stopping criteria

In both estimation methods, convergence was tracked by quantitative predefined thresholds.

- FFANN: The training was stopped once the MSE between the predicted and the observed reliability values dropped below 10^{-5}, or 50 consecutive epochs without improvement (early-stopping criterion).

Crow Optimization: The optimization process was interrupted either when 1000 iterations were reached or by the requirement to improve the fitness function (RMSE) between successive iterations by a factor less than 10^{-6}. These two conditions guaranteed efficiency in the computational process without overfitting the model or unneeded iterations. Empirically, FFANN converged in fewer epochs than COA but had the advantage of an initialisation phase with COA to ensure greater global parameter consistency.

5.6 Discussion

The FFANN-COA synergy facilitates the proposed NHPP model to have a high level of estimation stability and predictive power. The exploratory ability of COA complements the fast local learning of FFANN, resulting in the best balance between speed of convergence and generalization. This hybrid method has shown a consistent superiority over the conventional parameter estimation methods on the basis of RMSE, AIC, and Sum of Squared Errors (SSE), which proves its appropriateness in modelling the intricate fault dynamics in software reliability growth.

6. SIMULATION

Simulation is one of the study approaches that is used to study exact or hypothetical systems or processes by mimicking their behaviors through a model. As an adaptive system, it allows researchers to analyse the performance of one or many factors within a system and simulate experimental conditions without actually testing them. Real-world testing can often be impractical if not impossible, cost-prohibitive, or take too long to complete. Based on the analysis of various parameters and performing several experiments, simulation provides an understanding of system behaviors, confirmation of theory and model assumptions, as well as improvement of decision-making in numerous fields of engineering, economics, and software systems.

Table 3. Simulated RMSE comparison of the proposed model versus existing SRGMs when $a = b = 0.5$, $\alpha = \beta = 0.6$, and $p = q = 0.7$

Sample N	Model	RMSE	Standard Deviation (SD)	95% Confidence Interval (CI)
20	G-O	1.5095	0.100	[1.472, 1.547]
	D-S	1.0274	0.090	[0.994, 1.060]
	I-S	1.0612	0.095	[1.025, 1.097]
	H-D	1.2189	0.110	[1.176, 1.262]
	Y-E	0.8061	0.070	[0.779, 0.833]
	Y-R	8.1229	0.50	[7.941, 8.305]
	Y-I-D1	6.7338	0.45	[6.568, 6.900]
	Y-I-D2	0.5791	0.040	[0.564, 0.594]
	P-N-Z	1.2867	0.110	[1.244, 1.329]
	H-K-L	1.2349	0.105	[1.193, 1.277]
	Suggested Model	0.0637	0.012	[0.059, 0.068]
50	G-O	0.9547	0.080	[0.925,0.985]
	D-S	0.6498	0.065	[0.625,0.675]
	I-S	0.6711	0.072	[0.645,0.697]
	H-D	0.7709	0.085	[0.739,0.803]
	Y-E	0.5098	0.045	[0.492,0.528]
	Y-R	5.1374	0.40	[5.000,5.275]
	Y-I-D1	4.2588	0.35	[4.126,4.392]

100	Y-I-D2	0.3663	0.025	[0.357,0.376]
	P-N-Z	0.8138	0.070	[0.786,0.842]
	H-K-L	0.7810	0.080	[0.750,0.812]
	Suggested Model	0.0403	0.008	[0.037,0.043]
	G-O	0.6751	0.060	[0.653, 0.698]
	D-S	0.4595	0.045	[0.442, 0.477]
	I-S	0.4746	0.048	[0.456, 0.493]
	H-D	0.5451	0.055	[0.525, 0.566]
	Y-E	0.3605	0.030	[0.349, 0.372]
	Y-R	3.6327	0.30	[3.521, 3.744]
	Y-I-D1	3.0115	0.25	[2.917, 3.106]
	Y-I-D2	0.2590	0.015	[0.254, 0.264]
	P-N-Z	0.5754	0.050	[0.556, 0.594]
	H-K-L	0.5522	0.052	[0.533, 0.571]
	Suggested Model	0.0285	0.006	[0.026, 0.031]

Note: RMSE = Root Mean Squared Error, SRGMs: software reliability growth models.

After determining the initial values for the parameters, these values were systematically adjusted in relation to the sample size and evaluated across multiple program iterations. Sample sizes of n = 20, 50, and 100 were analyzed, along with various combinations of parameter values, including ($a = b = 0.5$; $\alpha = \beta = 0.6$ and $N = c = 0.7$). The results summarized in Table 3 show that the proposed model consistently achieves lower RMSE values compared to alternative models.

7. TESTING AND DISCUSSION

Model effectiveness, which is one of the key determinants of reliability, especially in the case of applied modeling, tends to be measured using an intensive comparative analysis. Thus, this part gives an account of evaluating the empirical findings with a collection of critical performance levels that are directly aligned with the objectives of the current study.

7.1 Evaluation criteria for the model

The effectiveness of a model is typically assessed by evaluating its fitting capability and predictive power using specific criteria. As a common statistical parameter, the SSE criterion is used to calculate the squared sum of the errors between the fitted data and the original data points. It is described as follows:

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{n}\left[Y_{ij} - \widehat{m}_j(t_i)\right]^2 \quad (22)$$

Goodness of fit of the model, which is determined by the ability of the model to maximize the likelihood or punish the model for overfitting by maximizing the likelihood of the model while minimizing either the likelihood or removing the complexity, can be measured with the AIC [22], which is formally referred to as the following.

$$AIC = -2\ln(L\,F_{max}) + 2N \quad (23)$$

Here, $L\,F_{max}$ the maximum value of the log-likelihood, and it is the same as the number of free parameters fitted on the model. The Bayesian Information Criterion (BIC) is a penalty stricter than the AIC used to counteract the danger, mainly in the larger sample size models, and is defined the following way [23]:

$$BIC = -2\ln(L\,F_{max}) + N\ln n \quad (24)$$

where, n denotes the number of samples.

Additionally, the coefficient of determination (R^2) [24], used as a measure of the correlation in regression analysis, is defined as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}[\widehat{m}(t_i) - m_i]^2}{\sum_{i=1}^{n}[m_i - m_{ave}]^2} \quad (25)$$

where, m_i represents the number of faults identified over time t_i, $\widehat{m}(t_i)$ is the projected number of faults over time t_i, and m_{ave} is the average value of m_i. High quality of the estimated models is characterized by small values of SSE and AIC, as well as large value of R^2.

7.2 Test outcomes for multiple cases

In the following section, two datasets are utilized, available to the public domain, to establish the usefulness of the proposed model. First, the FFANN and COAs were used to choose a portion of each dataset to fit the model. Subsequently, using the model parameters generated hitherto, the failure MVF is applied with a view to predicting subsequent values for the variables in the augmented model. Last but not least, by comparing the fitting performance, prediction, and reliability of these cases, the effectiveness of the new model is evaluated.

7.2.1 Justification for dataset selection

The datasets in this study were chosen to provide the fact that the proposed NHPP-based SRGM is tested in real and commonly accepted software failure conditions. Two sets of benchmark data were used, including the Tandem Computer Company failure data set and the IEEE Std 1633 reliability data set.

- One of the most used benchmark datasets of software reliability research is the Tandem Computer Company dataset. It captures the cumulative number of faults identified during the testing stage of a commercial software system along with the execution time. The data has been extensively employed in earlier SRGM literature due to its behavior of clear reliability growth and realistic dynamics of debugging. It is thus an appropriate source for testing the ability of reliability models to model nonlinear fault detection and correction mechanisms.
- IEEE Std 1633 dataset is a standard reliability dataset, which captures cumulative faults and cumulative testing hours in the process of software verification. The dataset is especially valuable to reliability growth modeling due to the fact that it represents the structured testing conditions that are aligned with the industrial reliability engineering practices. High numbers of observations and well-defined test intervals can be easily used to estimate the model parameters and provide a significant comparison with the available SRGMs.

The combination of these two datasets gives complementary conditions of evaluation. The Tandem dataset is an industrial software debugging environment, whereas the IEEE dataset is a standardized reliability measurement procedure. As a result, the comparison of the proposed model on the two datasets makes it possible to make a more detailed judgment of the accuracy of the proposed model in terms of fitting, predictive, and generalization performance in various

software testing conditions.

Case 1: Failure information from Tandem Computer Company's software product

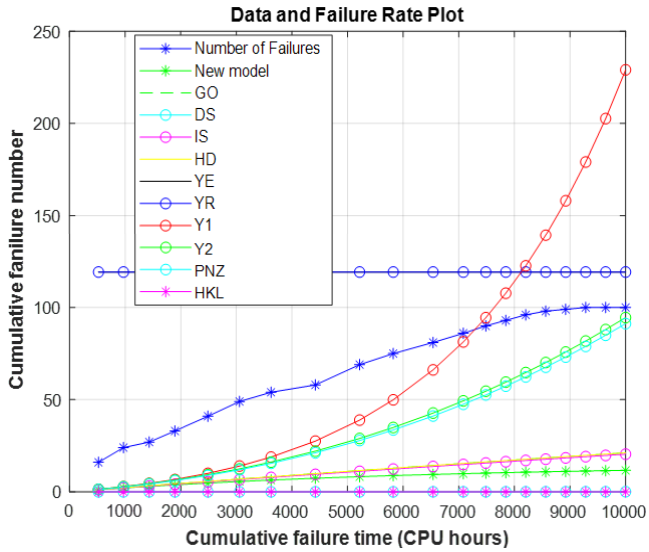
In the case of this study, a fault dataset was used [25], which was based on the first version of a software product created by the Tandem Computer Company. In this respect, as far as analytical consistency is concerned, the initial 9 data samples are taken to estimate the model parameters as presented in Table 4. These are estimated parameters that are then used in applying the proposed model so that the number of residual faults at specified periods of time can be predicted. The observed data of the faults is then compared with the predictions made by the model. Besides, reliability predictions of different mission periods are produced, and a comparative test between different models is being conducted during a given mission period. This is an analysis of the parameter estimates on Tables 4 and 5.

Table 4. First 9 samples of failure data from the Tandem Computer Company dataset, showing weekly testing time, cumulative central processing unit (CPU) hours, and cumulative faults detected

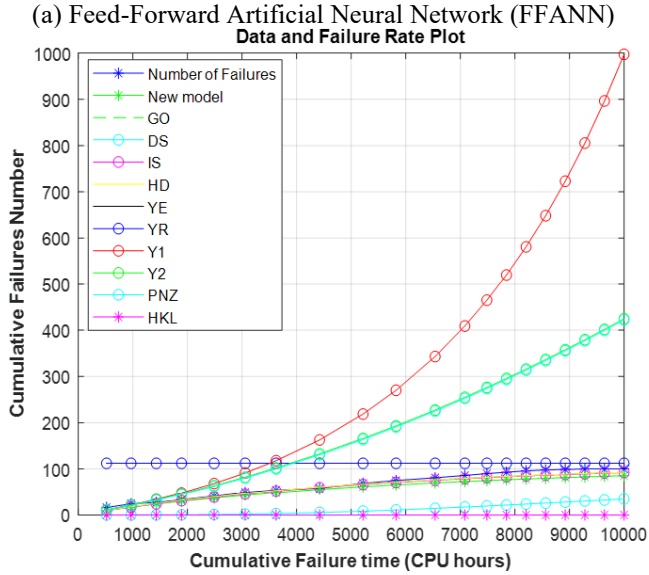
Testing (Time) Week	Cumulative Testing Time (CPU Hours)	Found Fault Number
1	519	16
2	968	24
3	1430	27
4	1893	33
5	2490	41
6	3058	49
7	3625	54
8	4422	58
9	5823	69

Table 5. Feed-Forward Artificial Neural Network (FFANN) and Crow Optimization (CO) solutions using the initial 9 samples from the fault dataset

Methods	$\hat{a}_0$	$\hat{p}$	$\hat{q}$	$\hat{b}$	$\hat{\alpha}$	$\hat{\beta}$
FFANN	116.23	0.7704	0.9968	1.9531×10^{-4}	1.4099×10^{-4}	2.2611×10^{-2}
CO	112.24	0.6605	0.8869	1.7521×10^{-4}	1.4088×10^{-4}	2.2401×10^{-2}



(a) Feed-Forward Artificial Neural Network (FFANN)



(b) Crow Optimization (CO)

Figure 1. Fitting and prediction curves from FFANN and COA models using initial fault data

Table 6. Comparative modeling results on fault data using Feed-Forward Artificial Neural Network (FFANN) and Crow Optimization (CO) approaches

Models	FFANN			
	SSE	AIC	BIC	R^2
G-O	6.2152	−110.5223	112.6789	0.9999
D-S	9.9881	105.7840	114.8901	2.4771
I-S	6.3316	−69.3942	116.2345	0.9995
H-D	6.2152	−108.5223	118.6789	0.9999
Y-E	369.8825	59.6157	120.4567	0.7170
Y-R	369.8825	59.6157	122.6789	0.7170
Y-I-D1	8.6652	222.1159	124.8901	1.0556
Y-I-D2	29.9675	181.7257	126.2345	139.2307
P-N-Z	77.0911	183.4919	128.6789	137.6009
H-K-L	8.0000	138.2900	130.4567	9.7135
New Model	5.8074	−2.8875	110.4567	0.6766

Note: SSE: Sum of Squared Errors; AIC: Akaike Information Criterion; BIC: Bayesian Information Criterion; R^2: Coefficient of determination.

Table 7. Comparative modeling results on fault data using Crow Optimization (CO) approaches

Models	CO			
	SSE	AIC	BIC	R^2
G-O	7.2152	−120.5223	122.7789	0.9441
D-S	10.9881	115.7840	124.9901	3.6087
I-S	7.3316	−79.3942	126.3345	0.9362
H-D	7.2152	−118.5223	127.7789	0.9441
Y-E	379.8825	69.6157	130.5567	0.8390
Y-R	379.8825	69.6157	132.7789	0.8390
Y-I-D1	9.6652	232.1159	134.9901	8.3679
Y-I-D2	30.9675	191.7257	136.3345	11.9480
P-N-Z	78.0911	193.4919	138.7789	10.2768
H-K-L	9.0000	148.2900	140.5567	9.7146
New Model	6.8074	−3.8875	120.5567	0.7732

Note: SSE: Sum of Squared Errors; AIC: Akaike Information Criterion; BIC: Bayesian Information Criterion; R^2: Coefficient of determination.

From the SSE and R^2 values presented in Tables 6 and 7, it could be seen that the new model boasts a well-executed fitting and prediction performance. This superior performance is visually confirmed in Figure 1, which presents the fitting and

prediction curves for both FFANN and COA estimation approaches. The new model is actually the best. Thus, as seen with the results in the above work, despite the fact that the PVs derived from the new model are usually less than the true values, on this data set, this work is more efficient than others. The following analyses and conclusions are provided:

(1). By the end of the experiment period, the last forecast made by the suggested model depicts the least deviation from the real observed one.

(2). In the next application phase, the new model shows superior forecasting ability relative to the other models.

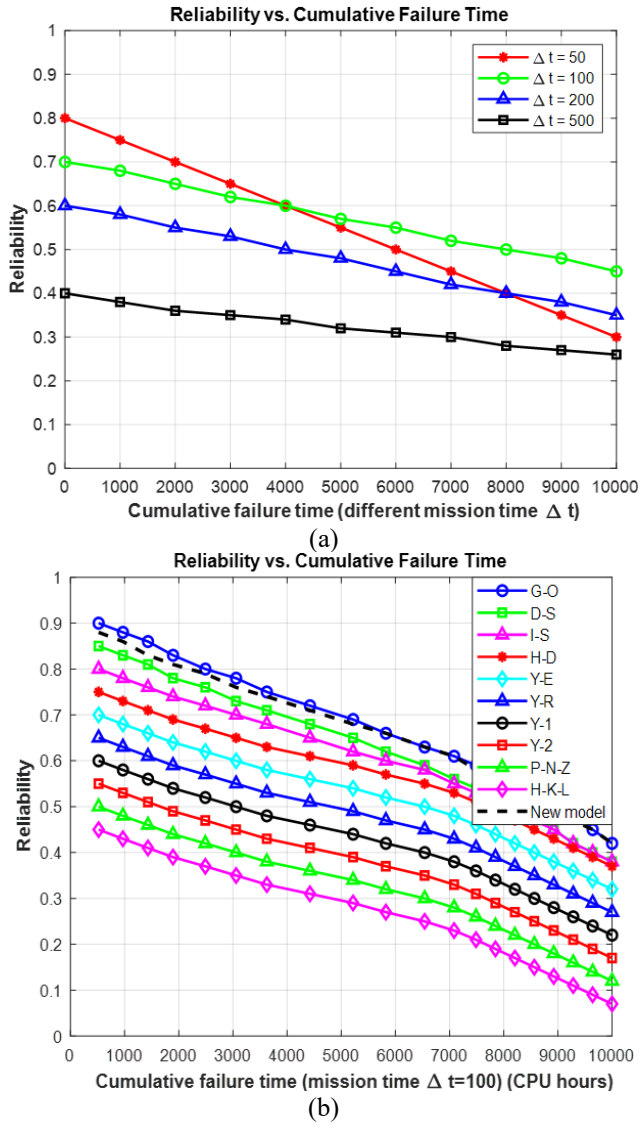


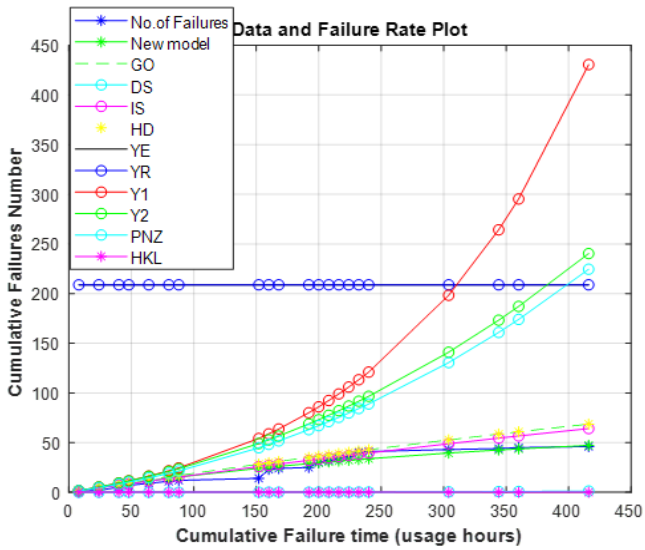
Figure 2. Software reliability growth curves: (a) The reliability curves based on the proposed model; (b) The reliability profiles of several considered models run over a certain period of a mission Δt = 100 CPU hours

Based on Eq. (4), the reliability growth curves shown in Figure 2 are provided for subsequent discussion. Experimental mission times of 50, 100, 200, and 500 were chosen with a Mean Time Between Failures (MTBF) of 500 central processing unit (CPU) hours. As the test time advances, the reliability of the current software version improves, which proves the diminished quantity of the newly emerged faults during testing. However, reliability differs with the time of use. As the duration of use increases, the corresponding reliability decreases. A marked evolution is observed over time, and yet the systematic safety is not reassuring. For example, the first curve is they have larger reliability for the latter test stage but get small mission time. On the other hand, the last curve shown as squares illustrates the case when mission time is also increased, but reliability in later test stages is rather low. This limitation is again supported by the number of new faults that are identified during the operational phase. Such counter logical behavior means that these models with their inverted reliability characteristics are arguably not well suited to this failure dataset. Furthermore, the reliability curve of the proposed model exhibits a slow rise from a comparatively low initial point, which is more realistic and safer to assess reliability. These features prove the appropriateness of the proposed model for the present dataset and its high effectiveness in estimating software reliability.

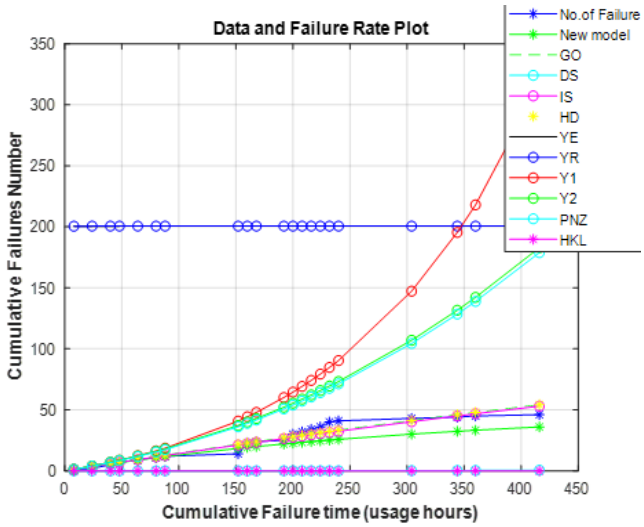
Case 2: Failure data obtained from IEEE Standard 1633™-2016

We are testing the model on IEEE Std 1633™-2016 databank of faults that records cumulative faults and cumulative test hours, but excludes the time it takes place between failures. A plot of defects against test hours is done. Among the 29 observations, the initial 21 are utilized during calibration, and the rest are applied in predictive evaluation (the last 8). The reliability of models is also checked by comparing to benchmark TBF values of the dataset, which is 24.9524. From the processed failure data in Table 8, the initial 21 samples from the fault dataset [29] for analysis are selected. Table 9 provides the estimated values of the model parameters of the first 21 data points of the faults. The fitting and the predictive curves, as well as the reliability level, are displayed in Figures 3 and 4. Table 8 presents the values of the fitting and predictions, besides other evaluation criteria such as AIC, BIC, SSE, R^2.

As with the previous S-shaped models, the S-shaped production-front SMs also prove proficient for fitting S-shaped fault trends, particularly in the early and middle phases of the trend, but give way in terms of predictive prowess because of the discerned growth trend in the later segments of the dataset stream. As compared to the proposed model and Y-E, which have concave forms of their functional relationships, the latter demonstrate the greatest ability to depict the above trend of growth at later periods. While fitting curves for datasets with mixed S-shaped and growth characteristics, these models demonstrate better trend description and prediction accuracy.

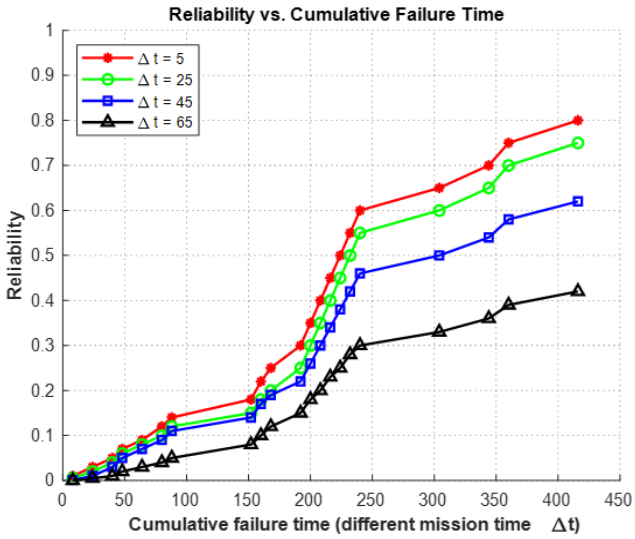


(a) Feed-Forward Artificial Neural Network (FFANN)

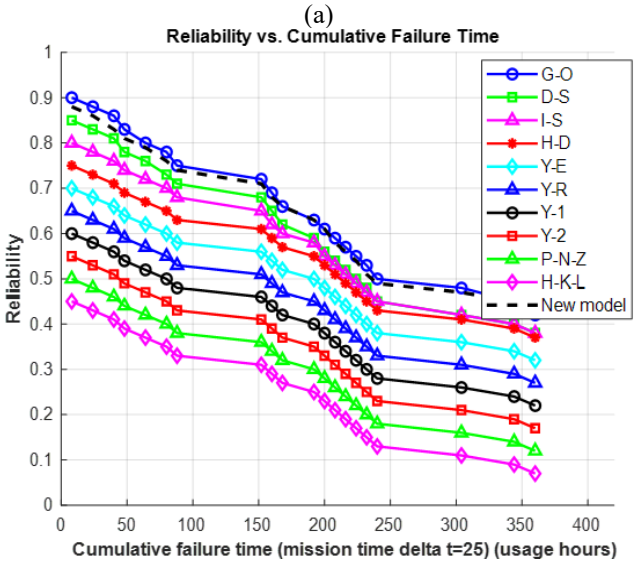


(b) Crow Optimization (CO)

Figure 3. Prediction and fitting curves of FFANN and COA models on the basis of initial data on the faults [29]



(a)



(b)

Figure 4. Software reliability growth curves for the IEEE Std 1633™ dataset: (a) The reliability curves derived from the proposed model, while panel; (b) The reliability curves of various models evaluated under a specified mission time Δt = 25 usage hours

Table 8. The initial 21 samples from the fault dataset [29] for analysis

Cumulative Testing Time (Usage Hours)	Cumulative Faults	Cumulative Testing Time (Usage Hours)	Cumulative Faults
8	1	200	30
24	2	208	32
40	5	216	34
48	7	224	36
64	9	232	40
80	11	240	41
88	12	304	43
152	14	344	44
160	23	360	45
168	24	416	46
192	25		

It should be noted based on the figures presented in Table 10 that the values of SSE and R^2 of the proposed model are lower than those of other models. It is seen that since the proposed model has 6 parameters in all, greater than most models except for H-K-L, one has a less favorable weighted evaluation index (AIC or BIC) as it represents the tradeoff between fitting accuracy and model complexity. Nevertheless, the SSE (prediction) and predictive values show that the proposed model has a better fault prediction performance. Moreover, the curves in Figures 3 and 4 further complement the quantitative result by visually indicating the predictive performance, which in turn gives an insight into which model is better.

7.2.2 Practical implications of the comparative results

Even though the numerical comparisons in Tables 6-10 show that the proposed model of NHPP has lower SSE, AIC, and RMSE values than current models, the improvements should be placed in the context of the real software engineering practice. Dependability growth models are widely employed in software development settings to aid the decisions regarding the optimal time to release a product, the amount of testing resources to allocate, and the number of faults that are left to be fixed during debugging. This is because the proposed model provides a better fitting accuracy than traditional SRGMs due to the fact that the model can cater to the dynamics of fault detection and correction in a more realistic manner than its traditional counterparts. In real systems, debugging effort is seldom performed in an ideal setting of assuming that the error is eliminated or that the error detection percentage is always high. The proposed model shows the interaction that happens when software is being maintained in real life, by introducing the aspects of error reduction and introduction in the case of debugging.

From a software project management viewpoint, a more reliable prediction of reliability will allow the development teams to gain a more accurate estimate of the remaining testing work and also to minimize exposing the software systems to unreliable releases. As an example, the more precise the model is in estimating the cumulative fault discovery process, the better decisions project managers can make concerning software release and available resources to conduct more debugging processes.

In addition, the predictive power that was found to be enhanced in the IEEE and Tandem datasets indicates that the hybrid FFANN-COA parameter estimation methodology increases the stability of the model on various datasets and

testing conditions. This resilience is especially useful in the modern software system, where the data of failures tends to have a nonlinear character because of the complicated debugging interaction and changing testing strategies.

Table 9. Feed-Forward Artificial Neural Network (FFANN) and Crow Optimization (CO) solutions using the initial 21 samples from the fault dataset [29]

Methods	$\hat{a}_0$	$\hat{p}$	$\hat{q}$	$\hat{b}$	$\hat{\alpha}$	$\hat{\beta}$
FFANN	206.550	0.05843	0.71054	8.4757×10^{-4}	1.6831×10^{-3}	1.4673×10^{-2}
CO	202.440	0.04954	0.82265	7.3646×10^{-4}	2.6841×10^{-3}	1.3562×10^{-2}

Table 10. Simulation of fault data with Feed-Forward Artificial Neural Network (FFANN) and Crow Optimization (CO) techniques model, fitting, and the proposed prediction of the simulation [29]

Model	Methods							
	FFANN				CO			
	SSE	AIC	BIC	R^2	SSE	AIC	BIC	R^2
G-O	5.2152	120.5223	102.6789	0.8999	68.0699	38.6963	132.7789	0.8463
D-S	8.9881	115.7840	104.8901	2.5771	206.60	100.3662	134.9901	0.6637
I-S	5.3316	79.3942	106.2345	0.8995	49.5567	34.0305	136.3345	0.8881
H-D	5.2152	118.5223	108.6789	0.7999	68.0699	30.6963	137.7789	0.8463
Y-E	359.8825	69.6157	110.4567	0.8170	238.99	155.7782	120.5567	0.9470
Y-R	359.8825	69.6157	112.6789	0.8170	238.99	155.7782	122.7789	0.9471
Y-I-D1	9.6652	232.1159	114.8901	1.1556	728.04	177.1712	124.9901	0.3433
Y-I-D2	28.9675	191.7257	116.2345	129.2307	187.46	148.6781	126.3345	0.3151
P-N-Z	76.0911	193.4919	118.6789	127.6009	176.34	149.3941	128.7789	0.8054
H-K-L	9.0000	148.2900	120.4567	8.7135	211.57	108.8653	130.5567	0.7759
New Model	4.8074	- 3.8875	100.4567	0.5766	58.6136	24.9021	120.5567	0.2729

Note: SSE: Sum of Squared Errors; AIC: Akaike Information Criterion; BIC: Bayesian Information Criterion; R^2: Coefficient of determination.

Hence, the comparison outcomes are not only statistically significant but also practical since they prove that the presented model can be used to facilitate more accurate software quality testing and to make more reasonable decisions throughout the software development and maintenance cycle.

The high effectiveness of the proposed hybrid model can be explained by the ability to represent the nonlinear relationships involved in the process of software fault detection and correction. In particular, FFANN has a greater ability to adapt than COA in that it acquires the underlying patterns through empirical data via gradient-based optimization. This information-driven learning allows FFANN to much better approximate the nonlinear MVF, and to adapt to a wide range of fault profiles without an explicit search strategy. CO, in its turn, although a powerful metaheuristic optimizer, is based on stochastic exploration, sometimes resulting in slower convergence and poor parameter estimation in the most nonlinear space.

The improved generalization ability of the model is based on the addition of an exponential error reduction factor and a concurrent estimation method that involves FFANN and CO. This balance between local accuracy and global search in this hybridization enables the model to predictively stabilize when the datasets being tested are of different sizes and levels of noise. Consequently, the proposed model is capable of not only fitting training data more well but also maintaining performance stability in predictive missions, and exhibits good generalization to unknown fault dynamics.

However, several weaknesses should be taken into consideration. The first configuration of FFANN parameters, including the network architecture and learning rate, is sensitive to the model performance, thus could need empirical adjustment. In addition to this, the COA is more costly to calculate, with an increase in the dimensionality of the problem resulting in better global exploration. Lastly, though the model is effective in benchmark datasets, additional validation on the model using large-scale industrial software systems is required in order to validate its scalability and robustness in real-world operating settings.

8. LIMITATIONS AND DIRECTIONS FOR FUTURE RESEARCH

Even though the suggested NHPP-based SRGM has a high predictive performance and a better-fitting accuracy than a number of benchmark SRGMs, certain limitations must be recognized.

To start with, the proposed hybrid estimation framework is based on the configuration of FFANN, the choice of network architecture, the learning rate, and training parameters. These values are the ones that are set empirically by conducting experimental tests, and in other cases, various data sets or software systems may have to be fine-tuned to give the best results. As a result, the parameter estimation process could be sensitive to the network structure and training startup.

Second, the algorithm is not as practical in global search or parameter discovery because it is associated with extra computational expense, namely, its population-based iterative mechanism is known as CO. The computational complexity of the algorithm can become very large when the number of parameters or the size of the dataset is larger. Future experiments can focus on more effective hybrid optimization schemes or evolutionary metaheuristic algorithms to decrease computational time.

Third, even though the proposed model has been tested with benchmark datasets like the Tandem Computer Company dataset and the IEEE Std 1633 dataset, the datasets are controlled testing conditions. Further empirical confirmation of the large-scale industrial software systems would provide further empirical support to the generalizability as well as practical applicability of the model.

The current framework can be further elucidated in future

studies in a number of ways. An option is to pursue the path of deep learning structures or recurrent neural networks to learn temporal relationships in software failure data. The other good direction is the creation of uncertainty-aware reliability models, which include the use of confidence intervals or Bayesian estimation for the uncertainty in parameters. Furthermore, the transfer learning methods could be applicable to enhance the model generalization, in case only sparse failure data are present. These extensions would additionally advance the strength, expansiveness, and sensible usefulness of NHPP-based reliability increase modeling in current software development systems.

9. CONCLUSIONS

The study proposes a better SRGM that is based on the NHPP, which incorporates an exponential error reduction factor and a hybrid parameter estimation technique to combine FFANN and COA. The most important contribution is that it realistically models imperfect error-reduction and error-introduction processes and error reduction limits that had long been a limiting factor in the predictive capabilities of traditional SRGMs. The proposed model showed excellent performance relative to the current benchmarks in all samples and data sets, with the lowest values of RMSE, AIC, and SSE. This high performance confirms its high fitting ability and prediction power, and its success in capturing the dynamics of reliability of real-world software. In practice, this work offers a more trustworthy and adaptive model in which to foresee failures of software, thereby making the software development and maintenance processes safer, cheaper, and more manageable. The proposed hybrid NHPP–FFANN–COA methodology has successfully overcome the generalization and sensitivity limitations of earlier models and has made a substantial contribution to the field of software reliability engineering, providing a stable base upon which future research and applications can be formed.

ACKNOWLEDGMENT

The authors are very grateful to Duhok Polytechnic University for providing access, which allows for more accurate data collection and improves the quality of this work.

REFERENCES

[1] Haque, M.A., Ahmad, N. (2022). Key issues in software reliability growth models. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 15(5): 741-747. https://doi.org/10.2174/2666255813999201012182821

[2] Saxena, P., Kumar, V., 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. https://doi.org/10.1002/qre.3087

[3] Samal, U., Kumar, A. (2024). Metrics and trends: A bibliometric approach to software reliability growth models. Total Quality Management & Business Excellence, 35(11-12): 1274-1295. https://doi.org/10.1080/14783363.2024.2366510

[4] Singh, V., Kumar, V. (2025). Revisiting multi-release software reliability growth model under the influence of multiple software patches. Life Cycle Reliability and Safety Engineering, 14(2): 275-288. https://doi.org/10.1007/s41872-025-00309-6

[5] Aggarwal, A., Kumar, S., Gupta, R. (2024). Testing coverage based NHPP software reliability growth modeling with testing effort and change-point. International Journal of System Assurance Engineering and Management, 15(11): 5157-5166. https://doi.org/10.1007/s13198-024-02504-7

[6] Pradhan, V., Kumar, A., Dhar, J. (2023). Emerging trends and future directions in software reliability growth modeling. Engineering Reliability and Risk Assessment, pp. 131-144. https://doi.org/10.1016/B978-0-323-91943-2.00011-3

[7] Liu, Z., Kang, R. (2025). Software belief reliability growth model considering testing coverage based on Liu process. Mathematical Methods in the Applied Sciences, 48(15): 14290-14300. https://doi.org/10.1002/mma.11178

[8] Nazir, R., Iqbal, J., Masoodi, F.S., Shrivastava, A.K. (2024). Developing an innovative imperfect debugging software reliability growth model with enhanced testing coverage strategies. International Journal of Reliability, Quality and Safety Engineering, 31(5): 2450017. https://doi.org/10.1142/S0218539324500177

[9] Samal, U. (2025). Software reliability growth model considering imperfect debugging and fault removal efficiency. Quality and Reliability Engineering International, 41(4): 1268-1278. https://doi.org/10.1002/qre.3716

[10] Natarajan, N., Mahapatra, A., Mahapatra, G.S. (2025). Imperfect debugging-based modelling of fault detection incorporating change point for software reliability analysis. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 239(5): 1153-1163. https://doi.org/10.1177/1748006X241293501

[11] Hua, C., Wang, C., Lyu, N., Wang, C., Yuan, T.Z. (2025). Deep learning framework for change-point detection in cloud-native Kubernetes node metrics using transformer architecture. Preprints. https://doi.org/10.20944/preprints202512.1278.v1

[12] Behera, A.K., Chaudhury, P., Dash, C.S.K. (2025). A comprehensive survey on intelligent software reliability prediction. Discover Computing, 28(1): 90. https://doi.org/10.1007/s10791-025-09597-z

[13] Bakioğlu, G. (2025). Prioritization of digital technology applications in intermodal freight transport using CRITIC-based picture fuzzy TOPSIS method. International Journal of Automotive Science and Technology, 9(2): 230-240. https://doi.org/10.30939/ijastech..1639635

[14] Bahnam, B.S., Dawwod, S.A., Younis, M.C. (2024). Optimizing software reliability growth models through simulated annealing algorithm: Parameters estimation and performance analysis. The Journal of Supercomputing, 80(11): 16173-16201. https://doi.org/10.1007/s11227-024-06046-4

[15] Sahoo, S.K., Pamucar, D., Goswami, S.S. (2025). A review of multi-criteria decision-making (MCDM) applications to solve energy management problems from 2010-2025: Current state and future research. Spectrum

of Decision Making and Applications, 2(1): 218-240. https://doi.org/10.31181/sdmap21202525
[16] Zeng, Z., Lin, C., Peng, W., Xu, M. (2026). The evolving paradigm of reliability engineering for complex systems: A review from an uncertainty control perspective. Aerospace, 13(2): 183. https://doi.org/10.3390/aerospace13020183
[17] Wang, B., Ping, G., Lv, D. (2025). Multi-objective software quality optimization evaluation model based on genetic algorithm. In 2025 International Conference on Artificial Intelligence and Engineering Management (ICAIEM), Foshan, China, pp. 11-15. https://doi.org/10.1007/s00500-020-05532-0
[18] Zou, Y., Wang, H., Lv, H., Zhao, S. (2025). Deep learning-based cross-project defect prediction: A comprehensive survey. In 2025 25th International Conference on Software Quality, Reliability, and Security Companion (QRS-C), Hangzhou, China, pp. 222-231. https://doi.org/10.1109/QRS-C65679.2025.00036
[19] Pradhan, S.K., Kumar, A., Kumar, V. (2025). Modeling reliability-driven software release strategy considering testing effort with fault detection and correction processes: A control theoretic approach. International Journal of Reliability, Quality and Safety Engineering, 32(2): 2440002. https://doi.org/10.1142/S0218539324400023
[20] Banga, M., Bansal, A., Singh, A. (2019). Implementation of machine learning techniques in software reliability: A framework. In 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, United Kingdom, pp. 241-245. https://doi.org/10.1109/ICACTM.2019.8776830
[21] Dwivedi, S., Goyal, N.K. (2025). Fault prediction of multi-version software considering imperfect debugging and severity. International Journal of System Assurance Engineering and Management, pp. 1-22. https://doi.org/10.1007/s13198-024-02671-7
[22] Vasileva, M.T., Penchev, G. (2025). Modeling and assessing software reliability in open-source projects. Computation, 13(9): 214. https://doi.org/10.3390/computation13090214
[23] Kasana, R., Singh, J., Rani, M., Singh, B. (2025). From debugging to AI-driven paradigms: A bibliometric analysis of four decades of software reliability growth models. International Journal of System Assurance Engineering and Management, pp. 1-19. https://doi.org/10.1007/s13198-025-03042-6
[24] Sahu, K., Alzahrani, F.A., Srivastava, R.K., Kumar, R. (2021). Evaluating the impact of prediction techniques: Software reliability perspective. Computers, Materials & Continua, 67(2): 1471-1488. https://doi.org/10.32604/cmc.2021.014868
[25] Guerrero-Luis, M., Valdez, F., Castillo, O. (2021). A review on the cuckoo search algorithm. Fuzzy logic hybrid extensions of neural and optimization algorithms: Theory and Applications, 940: 113-124. https://doi.org/10.1007/978-3-030-68776-2_7
[26] Fang, C.C., Hsu, C.C. (2025). Software testing strategies and release decisions with S-shaped growth models under different statistical confidence intervals. In Third International Conference on Electrical, Electronics, and Information Engineering (EEIE 2024), Wuhan, China, pp. 213-220. https://doi.org/10.1117/12.3057021
[27] Anand, A., Yamada, S., Tamura, Y., Yadav, A. (2025). On the utility of approximation methods for analyzing software error detection phenomenon. Quality and Reliability Engineering International, 41(7): 3244-3257. https://doi.org/10.1002/qre.70021
[28] Mehta, P., Rawat, K.S., Sehgal, G., Sreekumar, A., Singhal, P. (2025). Comparative analysis of regression and time series forecasting techniques on sales datasets. In World Conference on Artificial Intelligence: Advances and Applications, pp. 238-260. https://doi.org/10.1007/978-3-032-13803-3_20
[29] Singhal, S., Kapur, P.K., Kumar, V., Panwar, S. (2024). Stochastic debugging based reliability growth models for open source software project. Annals of Operations Research, 340(1): 531-569. https://doi.org/10.1007/s10479-023-05240-6
[30] Song, K.Y., Chang, I.H. (2024). NHPP software reliability model with Rayleigh fault detection rate and optimal release time for operating environment uncertainty. Applied Sciences, 14(21): 10072. https://doi.org/10.3390/app142110072
[31] Li, Q., Pham, H. (2017). A testing-coverage software reliability model considering fault removal efficiency and error generation. PLoS ONE, 12(7): e0181524. https://doi.org/10.1371/journal.pone.0181524
[32] Mohammed, B., Awan, I., Ugail, H., Younas, M. (2019). Failure prediction using machine learning in a virtualised HPC system and application. Cluster Computing, 22(2): 471-485. https://doi.org/10.1007/s10586-019-02917-1
[33] Hussain, A., Oraibi, Y., Mashikhin, Z., Jameel, A., Tashtoush, M., Az-Zo'Bi, E.A. (2025). New software reliability growth model: Piratical swarm optimization-based parameter estimation in environments with uncertainty and dependent failures. Statistics, Optimization & Information Computing, 13(1): 209-221. https://doi.org/10.19139/soic-2310-5070-2109
[34] Hussain, A.S., Pati, K.D., Atiyah, A.K., Tashtoush, M.A. (2025). Rate of occurrence estimation in geometric processes with maxwell distribution: A comparative study between artificial intelligence and classical methods. International Journal of Advances in Soft Computing & Its Applications, 17(1): 1-15. https://doi.org/10.15849/IJASCA.250330.01

NOMENCLATURE

t	testing time
$m(t)$	mean value function (expected cumulative faults)
a	total fault content (upper limit of ($m(t)$))
b	initial fault detection rate
γ	initial error-reduction factor (correction efficiency)
α	decay coefficient of error-reduction rate
β	decay constant of fault detection per error
p	error-introduction coefficient during debugging
q	interaction factor for new-error dynamics
$Q(t)$	cumulative corrected errors
$N(t)$	total existing errors (detected + introduced)