ENHANCED SOFTWARE RELIABILITY GROWTH MODELLING USING MODIFIED NEURAL NETWORKS AND CROW OPTIMIZATION FOR NHPP PARAMETER ESTIMATION

Adel S. Hussain[*], Rikan Abdulazeez Ahmed[**], Shaimaa W. Mahmood[**], Mohammad A. Tashtoush[***,****], Emad A. Az-Zo'bi [*****1]
[*]University of Duhok Polytechnic, Iraq.
[**]University of Mosul, Iraq.
[***]Al-Balqa Applied University, Jordan.
[****]Sohar University, Sohar, Oman.
[*****]Mutah University, Jordan.

ABSTRACT:
Non-Homogeneous Poisson Process (NHPP)-based Software Reliability Growth Models (SRGMs) are crucial for estimating software failure behavior over time. However, existing models often suffer from sensitivity to specific datasets, limited generalization, and inaccuracies in fault detection and correction. This paper introduces a novel SRGM that incorporates a generalized NHPP framework with an imprecise fault detection factor while leveraging Feedforward Artificial Neural Networks (FFANN) and the Crow Optimization Algorithm (CO) for parameter estimation. The proposed model introduces an error reduction factor to improve fault prediction accuracy; models fault detection rates using an exponential function, and establish a dynamic relationship between error introduction and residual errors. Experimental results demonstrate that the proposed model outperforms existing SRGMs in both short-term fault prediction and overall reliability estimation. The study validates the model using multiple datasets and evaluates performance based on key statistical metrics, confirming its robustness and superior predictive capability in software reliability assessment.
Keywords: NHPP, SRGM, Mean Value Function, Crow Optimization Algorithm, Feedforward Artificial Neural Networks, Software Reliability, Error Reduction Factor, Simulation.

MSC: 60G55, 68M15, 62M05, 90C26, 68T07.

RESUMEN: Los Modelos de Crecimiento de la Confiabilidad del Software (SRGM) basados en el Proceso de Poisson No Homogéneo (NHPP) son fundamentales para estimar el comportamiento de fallos del software a lo largo del tiempo. Sin embargo, los modelos existentes suelen presentar sensibilidad a conjuntos de datos específicos, limitada capacidad de generalización e imprecisiones en la detección y corrección de errores. Este trabajo introduce un nuevo SRGM que incorpora un marco NHPP generalizado con un factor impreciso de detección de fallos, aprovechando Redes Neuronales Artificiales Feedforward (FFANN) y el Algoritmo de Optimización por Cuervos (CO) para la estimación de parámetros. El modelo propuesto introduce un factor de reducción de errores para mejorar la precisión en la predicción de fallos, modela la tasa de detección de errores mediante una función exponencial y establece una relación dinámica entre la introducción de errores y los errores residuales. Los resultados experimentales demuestran que el modelo propuesto supera a los SRGM existentes tanto en la predicción de fallos a corto plazo como en la estimación global de la confiabilidad. El estudio valida el modelo utilizando múltiples conjuntos de datos y evalúa su desempeño mediante métricas estadísticas clave, confirmando su robustez y superior capacidad predictiva en la evaluación de la confiabilidad del software.
Palabras Clave: NHPP, SRGM, Función de valor medio, Algoritmo de optimización Crow, Redes neuronales artificiales de avance, Confiabilidad del software, Factor de reducción de errores, Simulación.

1. INTRODUCTION

The Simple Homogeneous Poisson Process, or a simple Poisson process in general, was initially introduced for basic execution time modelling [1], and it has later been applied for describing fault detection processes [2], making the capability of the Non-Homogeneous Poisson Process (NHPP) in software reliability analysis strong. This model is especially important because of flexibility as a way of handling different data sets. Generally, researchers and software reliability engineers focus on two key questions: It has (a) the time taken to gain a specified number of faults, and (b) reliability of the software at a certain time of use. The NHPP model fits right and answers both of these questions right [3-5].

The main work of the NHPP model is to estimate the Mean Value Function (MVF) which is the function that estimates the aggregate (cumulative) number of faults detected by time t in the software testing process. It is denoted as $m(t)$ in this study. Consequently, to obtain the

[1] eaaz2006@mutah.edu.jo

MVF for failures it is pertinent to analyse the error data of a special software component (a module or code, for example) to estimate its parameters. For the last few decades, scholars have proposed numerous important methods to assess the MVF of failure, where some of these are as follows [6-14]. For better understanding, table 1 below shows outline of key SRGMs according to the framework of NHPP.

No.	Model	$m(t)$
1	GO [2]	$a(1-e^{-bt})$ $a(t)=a\,, b(t)=b$
2	DS [6]	$a(1-(1+bt)e^{-bt})$ $a(t)=a\,, b(t)=\frac{b^2t}{1+bt}$
3	IS [26]	$\frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$ a(t) = a; b(t) =$\frac{b}{1+\beta e^{-bt}}$.
4	H-D/G-O [27]	$\log\left[\frac{(e^{a}-c)}{(e^{ae^{-bt}}-c)}\right]$
5	YE [10]	$a(1-e^{-\gamma\alpha(1-e^{-bt})})$ $a(t)=a; b(t)=r\alpha\beta e^{-\beta t}$
6	YR [10]	$a(1-e^{-\gamma\alpha(1-e^{-bt^2/2})})$ $a(t)=\ a; b(t)=r\alpha\beta t e^{\frac{-\beta t^2}{2}}$
7	Y-I-D1 [28]	$\frac{ab}{b-a}(e^{-at}-\ e^{-bt})$ $a=ae^{\alpha t}\,; b(t)=b$
8	Y-I-D2 [28]	$a(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha at$ $a(t)=a(1+\alpha t)\,; b(t)=b$
9	P-N-Z [12]	$\frac{a}{1+\beta e^{-bt}}\left[(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha t\right]$ $a(t)=a(1+\alpha t)\,; b(t)=\frac{b}{1+\beta e^{-bt}}$.
10	H-K-L [15]	$\frac{a}{1-\beta}\left[1-e^{-b(1-\beta)w^*(t)}\right]$ $a(t)=a+\beta n(t); b(t)=b$

Table 1. Establish MVF for the suggested model and compare it with other existing NHPP Software Reliability Growth Model

However, practical experience reveals that it is almost impossible to identify a general model capable of addressing various types of fault configurations. In other words, several models work well only for given datasets or systems, which restrict them from applications across the board. While it has become relatively routine to consider one or another Mean Value Function (MVF) for failures relying on the NHPP, there are still many difficulties associated with constructing widely applicable models in a number of spheres.

(i) Indeed, the vast majority of assumptions are not global in nature. For example, the S-shaped models can reduce the ability to capture cases where there are elements that grow and develop in a step-wisely or in other words, non-monotonous manner, such as [5-7, 11, 16] represent the outcome of a 'learning' process aimed at dynamically improving testing efficiency. In real-world testing scenarios, the 'learning' mechanism is rarely engaged, largely because of resource limitations and the dependence on predefined, inactive profiles for shaping tests and business strategies [17].

(ii) Certain assumptions lack practical validity and evidence not available in the actual circumstance. When analyzing prior generalized imperfect debugging models [10-12], it is usually assumed that the overall number of errors obey some given increasing functions in terms of time. Furthermore, the error introduction rate is assumed to bear

a relationship with the error correction procedure. But as to how this relation can best be described, this proposed typology leaves much to be debated.

(iii) Overall, most of the existing models can be classified and referred to as the least error models. Though the number of latent errors that are buried in code is still limited by the coding rules and constraints, the practical experience shows that the code fixing can generate new errors with given probability [18]. Consequently, a small number of errors could generate an endless number of errors after applying modifications. Thus, it is possible to suggest the creation of an effective model, which may be considered as one with infinite errors but targeting on the characteristics of the limited errors.

(iv) A primary concern in the NHPP model is to estimate the MVF for the failures. However, in many cases, the estimator can only be obtained when a closed-form solution for the MVF is available for a limited number of special cases. This limitation makes it very difficult to make correct assessment of the model parameters of the model. Thus, the necessity to invest more time in developing a new model, which will help to define the MVF of failures using the closed-form solution and with access to which will not take much time, cannot be overestimated.

To address these challenges, we propose a new SRGM in this study, based on the NHPP framework for generalized imperfect debugging. First, we establish a typical constraint relationship between the error introduction rate and the rate of change of generalized residual errors, grounded in reasonable assumptions consistent with the error introduction process. Additionally, we propose an exponential linear function to model the error detection rate per error and introduce an error reduction factor to quantify the effectiveness of error correction. Based on these assumptions, we derive a closed-form expression for the MVF of failures. Furthermore, we present a detailed discussion of the MVF's properties and demonstrate its consistency with traditional models under specific conditions. Experimental results indicate that our suggested model demonstrates superior performance in error fitting and short-term prediction within a specific range. Finally, we summarize the key contributions of this paper as follows:

(i) Unlike current models, we aim to incorporate an error reduction factor, which takes the form of an exponential linear function of the number of existing errors, as a measure of error removal (modification) efficiency within the framework of generalized imperfect debugging.

(ii) Considering the rate of change in the remaining errors, which can be utilized to dynamically reflect the efficiency of error removal and infer the likelihood of error introduction, we propose a new perspective: the error introduction rate is directly related to the rate of change in the generalized remaining errors.

(iii) Thus, in view of the possibility that the efficiency of the error detection is lower in conditions close to real testing, instead of the traditional S-curve, we will use an exponential linear function to assess the rate of error detection per each error.

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 Reference [18] including maintenance strategies; multi criteria decision making techniques for analysing SRGMs was discussed by Reference [19]. Recent research has also employed machine learning and deep learning to software reliability analysis. [20-22].

This paper proposes a novel NHPP-based Software Reliability Growth Model with an original error reduction mechanism and a dynamic error introduction process, representing a significant algorithmic advancement in software reliability modelling.

To ensure the topic of SRGM is explored to the fullest, the paper is divided into several major parts. The first chapter, which is the chapter on 'Introduction' appreciates the role of software reliability and establishes the fact that existing models have their drawbacks.
The Methodology section presents the proposed NHPP-based model which demonstrates its parameter estimation through the application of Crow Optimization Algorithm. The Results section contains data from model evaluation which demonstrates the superiority of the proposed technique through comparisons with other methods. 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 Crow Search Algorithm (CSA) is a met-heuristic algorithm inspired by crow intelligent behaviour to store food, hiding, and recalling the place to get it back at the right time. Crows are also capable of memorizing positions or locations in which other crows hid which in turn leads to inspiration for a solution in optimization problems [19].
Each crow in CSA is a potential solution to an optimization problem and the position of a crow in the search space corresponds to a set of decision variables. The algorithm cycles the crows through the search space, and in their attempt to optimize their positions, use either the memory of known solutions or move at random so as not to alert others. The basic concept is to guarantee the exploration of new areas as well as the exploitation of the existing solutions, in an aim of arriving at the best solution [20].

- **(i) Position of the Crow**: Each crow is considered as a solution vector, where the elements will be one of the model parameters.
- **(ii) Memory:** Self-organized crow: they remember or at least the best position found so far, which is hiding spots. These are stored as potential solutions and changed during an optimization process.
- **(iii) Awareness Probability:** Other possibilities may include crows understanding that other crows are following them. In such cases they go to other parts of the forest in order to hide themselves from human beings. This awareness probability is useful to overcome the problem of stagnation and monotony in the search.
- **(iv) Movement:** In each iteration, crows either move to their optimal position (best colouring hiding position) or a random position determined by the awareness probability. The position update for crow i is determined by:

  $$x_i^{(t+1)} = x_i^{(t)} + r * f * (m_i^{(t)} - x_i^{(t)}) \quad (1)$$

  where:
  $x_i^{(t)}$: is the current position of crow i
  r : is a random number between 0 and 1
  f : is the flight length (a control parameter for step size).
  $m_i^{(t)}$: is the crow's memory (best-known solution).
- **(v) Termination:** The algorithm continues until a stopping criterion is met (such as the maximum number of iterations or a threshold for minimal improvement).

3. FEED FORWARD ARTIFICIAL NEURAL NETWORKS

The model is classified as a simple feed forward neural network model, in which information flow is strictly unidirectional and sees no recurrent pathway between the layers. The coupled architecture of FFANNs makes them suitable for all sorts of tasks including regression, classification, and pattern recognition.

(i) Input Layer:

- The first is a layer that defines and specifies the input features, or data on which the network will act. Every node (or neuron) in this layer will correspond with a single feature.
- Finally, in practical application, these inputs may be of numerical type or pixels of an image, or of time series type and so on.

(ii) Hidden Layers:

- They are between the input and the output though they are more complex layers. In simpler terms, hidden layer applies transforms to the received inputs through weights and biases which formulate further processing.
- Each neuron in the layers of the hidden layer will sum product of weight value of each extracted feature the data is received from the prior layer and processed through an activation function.
- Explaining the effect of having more hidden layers and neurons the author notes that it enables the network to learn more complex dependencies in the data in question.

(iii) Output Layer:

- This layer gives the final output or the result or even the forecasted values. The number of neurons in the output layer depends on the type of problem:

(a) In regression, there can be only one output neuron.
(b) In classification, there can be several output neurons depending on the classification's various possibilities.

(iv) Weights and Biases:

- Every synapse between neurons has a weight that determines the signal passing between the neurons.
- Every neuron has a weight assigned to it and there is also a bias which is stirred with the inputs to be launched through an activation function. This helps the network to translate the activation function curve and enable it make betters predictions.

(v) Activation Functions:

- New layers add non-linearity in the network so as to enable it model more complex relations.
- Examples of activations functions are:
  a. Sigmoid: Normalises the input values to be in between 0 and 1.
  b. ReLU (Rectified Linear Unit): Returns 0 if the input is negative while returns the input as it is if the input is positive.
  c. Tanh: They scale inputs of the neurons to a range of -1 to 1.
- If non-linear activation functions are not used, then the given network resembles the simple linear model which is not capable of solving sophisticated problem.

3.1 Limitations of FFANNs

- Fixed Input and Output Size: A weakness of FFANNs lies in the predictability of the number of input and output neurons which make 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 NHPP FRAMEWORK AND ITS ASSOCIATED KEY TASKS

From the glossary shared above it is seen that there are profound distinctions between software error, fault and failure and that their correlations can be intricate. For example, one error may result in several faults, whereas, several errors may be the source of one and the same fault. To simplify the modeling process, we make a key assumption in this paper: each fault (or failure) is thus in one-to-one correlation with one or another error.

When a software product is under the testing phase, or whenever it is operative, the manifestation of fault that refers to inherent errors in the program is random. This randomness is due to the execution environment or Profile of the Software. On this account, the non-aftereffect of the Markov process allows count detected faults and remaining errors at time t, presumably, the complexity of the fault discovery process is mainly influenced by the frequency of residual errors and the operational profile involved of the software [20]. Fundamentally, the NHPP, a type of counting process, is built upon the following four assumptions:

- Characterized by independent increments, the failure process ensures that the number of failures in each time period is independent of prior occurrences $(t + t + \Delta t]$ only depends on a current time parameter t and interval parameter length Δt.
- For the interval (t; t + △t], the probability of occurrence of more than one failure is $o(\Delta t)$, i.e., $p\{N(t + t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$.
- For the interval (t; t + △t], the probability of occurrence of one failure is $\lambda(u)\Delta t + o(\Delta t)$, i.e., $p\{N(t + t + \Delta t) - N(t) \geq 1\} = \lambda(u)\Delta t + o(\Delta t)$.

The NHPP, commonly assumed in most SRGMs, is represented by the following equation:

$$p[N(t) = y] = \frac{[m(t)]^y e^{-m(t)}}{y!}, \quad y = 1,2,3, \ldots \tag{2}$$

Using $m(t)$ [23], the NHPP-based dependability function can be expressed as follows. The probability of no failures occurring within the time interval (0, t) is represented by the reliability function (t), which is given by:

$$R(t) = p\{N(t) = 0\} = e^{-m(t)} \tag{3}$$

Reliability $R(y/t)$ typically represents the probability of no failures occurring during the interval.$[t, t + \Delta t]$ is given by:

$$R\left(\frac{\Delta t}{t}\right) = p\{N(t + \Delta t) - N(t) = 0\} = e^{-[m(t+\Delta t) - m(t)]} \tag{4}$$

Eq. (4) forms a cornerstone for analysis of software failures, which constantly appear in modern systems. It can therefore be used to examine useful parameters for $m(t)$, including software reliability at Time t, the number of residual errors up to time t and the time it will take to achieve a given number of failures. Therefore, the failure MVF $m(t)$ is important to the NHPP mechanism.

Generally, the failure Mean Value Function (MVF) $m(t)$ encompasses numerous unknown parameters. Therefore, estimating these parameters is a critical task in software reliability analysis.

4.1 Utilizing a new method to analyze software reliability growth models

In the NHPP framework, the derivation of the traditional failure Mean Value Function (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: The proportion of failure detection, above mentioned, is postulated as depending on time, t. i.e. $k(t)$; (ii) By correction of Read errors, other errors may

be produced in the process or in other words, the total number of errors, $a(t)$, 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 t is corrected probabilistically. The generalized imperfect debugging NHPP model, which accounts for error removal efficiency, is expressed as follows: [18]:

$$m'(t) = k(t)\,[\,a(t) - Qm(t)], \tag{5}$$

Q is the measurement of the likelihood of error correction while (t) is the number of errors that has been corrected by the time (t). For generalization, the proposed new Software Reliability Growth Model (SRGM) is primarily derived from the following differential equation:

$$m'(t) = k(t)\,[\,a(t) - x(t)] \tag{6}$$

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

4.1.1 Three important assumptions

(i) Modeling Assumptions for Exponential Error Reduction

In this context, therefore, we adopt the error reduction factor as the measure of differential constraint applicable to the observable number of errors (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 sections that is 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 the efficiency of it against 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 is modeled as a decreasing exponential function of (t). This formulation is derived from the defined characteristics of the error reduction factor and the number of modified errors (t) can be described as follows [20]:

$$\frac{x'(t)}{m'(t)} = qe^{-\beta m(t)} \tag{7}$$

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 this formula (7) is quite stable. Under this assumption, if $t \to 0, m(t)$ tends to be 0 and $\frac{x'(t)}{m'(t)}$ tends to be q.

(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 in order 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. In 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. Therefore, it would be wrong to describe the change rate of residual errors as $x'(t)$; rather, it must be defined as $[a(t) - x(t)]'$ If the total number of errors is taken to be constant and equal to $a(t) = a_0$, then $x'(t)$ and $[a(t) - x(t)]'$ essentially convey the same mathematical and physical significance. We call $[a(t) - x(t)]'$ as the first derivative of the generalized MSD, containing an estimate of the speed at which error is being reduced.

Given that the number of residual errors typically decreases over time, we assume that the average ratio between the error introduction rate and the rate of change of generalized residual errors is $-p$, the differential relationship between the total number of errors $a(t)$ and the modified number of errors $x(t)$ can be expressed as follows:

$$\frac{a'(t)}{[a(t)-x(t)]'} = -p \tag{8}$$

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

(iii) Fault detection rate per error

In this section, we propose that the error detection rate for each error follows an exponential linear function of time t. However, it should be noted that the detection rate for each residual error in the software is not uniform. Consistent with our current assumption, we also consider that all errors associated with the remaining defects will be detected at the same discovery rate at a given time. However, we further recognize that the detection rate for each error may diminish over time. The underlying causes of this decline are as follows:

1. This is illustrated in Section 1 where most of the authors pointed out that the so called 'learning' process does not normally happen in real life industrial and testing scenarios, 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, we posit that the fault detection rate per error follows an exponential decay with respect to time t, which can be expressed as follows:

$$k(t) = be^{-(a+\alpha t)} \tag{9}$$

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

4.2 An Innovative SRGM and Its Failure Mean Value Function

A. An Advanced SRGM

Taking into account the assumptions outlined earlier, our proposed model for sustainable human resource management can be summarized as follows:

I Physical errors in the software are: (i) Each fault corresponds to a single error; (ii) incorporate the NHPP as a model of the occurrence of faults (or failures) in the software; (iii) it is assumed that debugging (or modification) is initiated immediately once a fault (or failure) is detected; (iv) Fault detection rate is directly proportional to the number of generalized residual errors still in the software where the proportionality factor is the generalized direction, $k(t)$, towards fault detection for each error. This factor is a function of time t, and its behavior decreases exponentially as the value of time increases. (v) The efficiency of error removal is expressed using an error reduction factor in the form of an exponential linear function of the number of detected errors $m(t)$. (vi) The error introduction rate is directly proportional to the rate of change of generalized residual errors hidden within the software, with the proportionality factor being $-p$.

According to the aforementioned assumptions, our proposed SRGM is expressed as follows:

$$\begin{cases} m'(t) = k(t)\,[\,a(t) - x(t)] & (i) \\ \dfrac{x'(t)}{m'(t)} = qe^{-\beta m(t)} & (ii) \\ \dfrac{a'\,(t)}{[a(t)-x(t)]'} = -p & (iii) \\ k(t) = be^{-(a+\alpha t)} & (iv) \end{cases} \tag{10}$$

That is, through solving Four Equations above simultaneously under some initial conditions, one can obtain the corresponding solution, namely the mean value function (MVF) $m(t)$. In our case, we follow that, to articulate the explicit failure MVF, we can present the following theorem:

Remark 1: Given that the failure MVF, fault removal efficiency, error introduction rate, and fault detection rate per error conform to the relationships expressed in Eq. (10), under the initial conditions,$n(0) = 0, x(0) = 0\ and\ a(0) = a_0$, the failure MVF is derived as follow:

$$n(t) = \frac{1}{\beta}\ln\left[\frac{\beta a_0\, e^{\left[\beta a_0 - q(1-p)*\frac{b}{\alpha}e^{-a}[1-\,e^{-\alpha t}]\right]} - q(1-p)}{\beta a_0 - q(1-p)}\right] \tag{11}$$

B. Further Analysis of Failure Mechanisms in MVF

In this section, we further explore the failure mechanisms of MVF and present several deductive results.

- In (11) once the time t goes to infinity, then $n(t)$ will converge to upper bound, $i.e.$,

$$\lim_{t\to\infty} n(t) = \frac{1}{\beta}\ln\left[\frac{\beta a_0\, e^{\left[\beta a_0 - q(1-p)*\frac{b}{\alpha}e^{-a}\right]} - q(1-p)}{\beta a_0 - q(1-p)}\right] \tag{12}$$

- Under the assumed initial conditions $n(0) = 0\ and\ x(0) = 0$, an identity $x(t) = \frac{q}{\beta}\left[1 - e^{-\beta m(t)}\right]$ can be followed by differentiation, directly from (10). (ii) Substituting Eq. (11) into followed identify, we can get the number of modified errors shown in Eq. (13).

$$x(t) = \frac{q}{\beta}\left[1 - \frac{\beta a_0 - q(1-p)}{\beta a_0\, e^{\left[\beta a_0 - q(1-p)*\frac{b}{\alpha}e^{-a}\right]} - q(1-p)}\right] \tag{13}$$

- Under the initial condition $a(0) = a_0$ substituting Eq. (13) into (10) (iii), we can get the total error number:

$$a(t) = a_0 + px(t) = a_0 + \frac{pq}{\beta}\left[1 - \frac{\beta a_0 - q(1-p)}{\beta a_0\, e^{\left[\beta a_0 - q(1-p) * \frac{b}{a} e^{-a}\right]} - q(1-p)}\right] \quad (14)$$

- If $\alpha \to 0$ (This implies that the fault detection rate per error is approximately constant. Therefore, Eq. (11) can be rewritten as:

$$n(t) = \frac{1}{\beta} \ln\left[\frac{\beta a_0\, e^{[\beta a_0 - q(1-p) * be^{-a}t]} - q(1-p)}{\beta a_0 - q(1-p)}\right] \quad (15)$$

- In Eq. (15), if $p \to 0$ (This indicates that the error introduction rate is approximately zero). Therefore, Eq. (16) can be rewritten as:

$$n(t) = \frac{1}{\beta} \ln\left[\frac{\beta a_0\, e^{[\beta a_0 - q * be^{-a}t]} - q}{\beta a_0 - q}\right] \quad (16)$$

- Under the conditions of $\alpha \to 0\ and\ p \to 0$, the limit $\beta a_0 \to q$ is satisfied based on (10) (ii). So, in Eq. (16), if $b \to 1$, $\beta a_0 \to q$ we can obtain a general expression:

$$n(t) = \frac{1}{\beta} \ln[1 + \beta a_0 e^{-a}t] \quad (17)$$

Eq. (17) represents the Logarithmic Poisson Execution Time Model (LPETM) [21]. In this context, our proposed model, derived from Eq. (10) and incorporating novel perspectives can be viewed as a generalized form of the LPETM. Specifically, Eq. (17) describes an infinite fault model. However, by considering factors such as error introduction, error reduction, and the decrease in fault detection, the new model derived in Eq. (11) combines the concave model with the infinite fault model. This new model retains the characteristics of the infinite fault model while converging to an upper bound.

5. ESTIMATION PARAMETERS

In this section, the parameters of the proposed model are estimated using two approaches: Modified FFANN and CO.

5.1. Parameter Estimation Using Modified Feed-forward Artificial Neural Network (FFANN)

This section outlines the initialization and training process of a feed-forward artificial neural network (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, add a bias and apply an activation function $n(t)$.. Inputs from the first layer go as inputs to the second, third still to the final layer that will give an output.

(iv) Loss Function:

- The mean squared error (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:

$r_{k\,new} = r_{k\,old} - aM_k{}^{-1}g_k,$

$c_{k\,new} = c_{k\,old} - aM_k{}^{-1}g_{k,}$

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:

$w_{k\,new} = w_{k\,old} - \left[L_k{}^T L_k\ + b\ I\right]^{-1} g_k,$

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

where w is the weight of 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 Crow Optimization algorithm (CO)

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

(i) Initialization:

1. Define the search space for each parameter $(\beta, a0, q, p, b, \alpha, a)$.

2. Initialize a population of crows (solutions) randomly within the parameter bounds.

3. 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 Mean Squared Error 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.

6. SIMULATION

Simulation is one of the study approaches that are used to study exact or hypothetical systems or processes by mimicking their behaviour 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 it out. 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 understanding of system behaviour, confirmation of theory and model assumptions, as well as improvement of decision-making in numerous fields of engineering, economics, and software systems.

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 analysed, along with various combinations of parameter values, including ($a = b = 0.5;\ \alpha = \beta = 0.6\ \&\ N = c = 0.7$). The results, summarized in Table 2, show that the proposed model consistently achieves lower RMSE values compared to alternative models.

No.	Model	Sample size(n)	$RMSE$	Sample size(n)	$RMSE$	Sample size(n)	$RMSE$
1	G-O	20	1.5095	50	0.9547	100	0.6751
2	D-S		1.0274		0.6498		0.4595
3	I-S		1.0612		0.6711		0.4746
4	H-D		1.2189		0.7709		0.5451
5	Y-E		0.8061		0.5098		0.3605
6	Y-R		8.1229		5.1374		3.6327
7	Y-I-D1		6.7338		4.2588		3.0115
8	Y-I-D2		0.5791		0.3663		0.2590
9	P-N-Z		1.2867		0.8138		0.5754
10	H-K-L		1.2349		0.7810		0.5522
11	Suggested Model		0.0637*		0.0403*		0.0285*

Table 2. The Simulated RMSE of each model listed in table 1. Estimating parameters with various sample sizes and estimation techniques when $a = b = 0.5;\ \alpha = \beta = 0.6\ \&\ p = q = 0.7;$

7. TESTING AND DISCUSSION

From an application perspective, model quality, which always is regarded to be vital, is normally determined by the following comparisons. Thus, in this section, we will evaluate the tests and their outcomes following several critical evaluation indices that form part of the research objectives.

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 Sum of Squared Errors (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:

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{n}\left[Y_{ij} - \hat{m}_j(t_i)\right]^2 \tag{19}$$

The ability of the model to maximize the likelihood function can be evaluated using the Akaike Information Criterion (AIC), which is defined as follows [22]:

$$AIC = -2\ln(L\,F_{max}) + 2N, \tag{20}$$

Here, $L\,F_{max}$ represents the maximum value of the logarithmic likelihood function, and N denotes the number of parameters in the model. The Bayesian Information Criterion (BIC), which imposes a larger penalty term than AIC to prevent model overfitting at high precision, is defined as follows [23]:

$$BIC = -2\ln(L\,F_{max}) + N\ln n \tag{21}$$

where, n denotes the number of samples.

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

$$R^2 = 1 - \frac{\sum_{i=1}^{n}[\hat{m}(t_i) - m_i]^2}{\sum_{i=1}^{n}[m_i - m_{ave}]^2}\ , \tag{22}$$

Were, m_i represents the number of faults identified over time t_i , $\hat{m}(t_i)$ 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, we utilize two datasets available to the public domain to establish the usefulness of the proposed model. First, we used the FFANN and CO algorithms 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 the least, by comparing the fitting performance, prediction and reliability of these cases, we evaluate the effectiveness of the new model.

Case 1: Failure Information from Tandem Computer Company's Software Product

In this study, we analyze a fault dataset [25] from the first version of a software product developed by Tandem Computer Company. To ensure consistency in our analysis, we select the first 9 samples from the dataset to estimate the model parameters outlined in Table 3. These estimated parameters are then used to apply the proposed model, allowing us to predict the expected number of residual faults in the software at specific time points, and we compare these predictions with the actual error counts. Additionally, we present reliability predictions for several mission intervals and assess the comparative reliability of various models across a defined mission time. The analysis is conducted based on the parameter estimates provided in Table 4.

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 3. Processed failure data we select the first 9 samples from the dataset [25]

Methods	$\widehat{a}_0$	$\widehat{p}$	$\widehat{q}$	$\widehat{b}$	$\widehat{\alpha}$	$\widehat{\beta}$
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}

Table 4. FFANN and CO solutions using the initial 9 samples from the fault dataset [25].

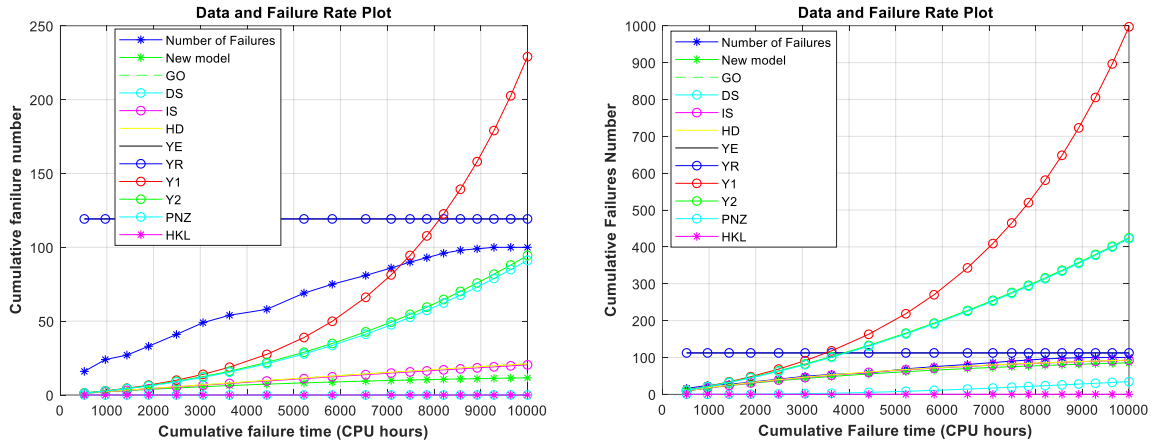
Models	Methods							
	FFANN				CO			
	SSE	AIC	BIC	R^2	SSE	AIC	BIC	R^2
G-O	6.2152	-110.5223	112.6789	0.9999	7.2152	-120.5223	122.7789	0.9441
D-S	9.9881	105.7840	114.8901	2.4771	10.9881	115.7840	124.9901	3.6087
I-S	6.3316	-69.3942	116.2345	0.9995	7.3316	-79.3942	126.3345	0.9362
H-D	6.2152	-108.5223	118.6789	0.9999	7.2152	-118.5223	127.7789	0.9441
Y-E	369.8825	59.6157	120.4567	0.7170	379.8825	69.6157	130.5567	0.8390
Y-R	369.8825	59.6157	122.6789	0.7170	379.8825	69.6157	132.7789	0.8390
Y-I-D1	8.6652	222.1159	124.8901	1.0556	9.6652	232.1159	134.9901	8.3679
Y-I-D2	29.9675	181.7257	126.2345	139.2307	30.9675	191.7257	136.3345	11.9480
P-N-Z	77.0911	183.4919	128.6789	137.6009	78.0911	193.4919	138.7789	10.2768
H-K-L	8.0000	138.2900	130.4567	9.7135	9.0000	148.2900	140.5567	9.7146
New Model	**5.8074**	**- 2.8875**	**110.4567**	**0.6766**	**6.8074**	**- 3.8875**	**120.5567**	**0.7732**

Table 5. Fitting and prediction results derived from different models using the fault dataset, including those using FFANN and CO approaches [25].

From the SSE and R^2 values presented in Table 5, it could be seen that the new model boasts a well-executed fitting and prediction performance. 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. Upon completing the testing phase, the final predicted value of the new model is nearest to the actual value.

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



A) FFANN **B)** CO

Figure 1. The fitting and prediction curves for various models, including those based on FFANN and CO, are constructed using the first 9 fault data points from the dataset [25].

Based on Equation (4), the reliability growth curves shown in Figure 2(a) are provided for subsequent discussion. Experimental mission times of 50, 100, 200, and 500 were chosen with an MTBF of 500 CPU hours. As the test time advances, the reliability of the current software version improves, which prove 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 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.

In Figure 2(b) the reliability curves of models D-S, Y-R and H-K-L are characterized by a steep drop during testing phase and rising steeply to one at different rates during some phase of testing 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.

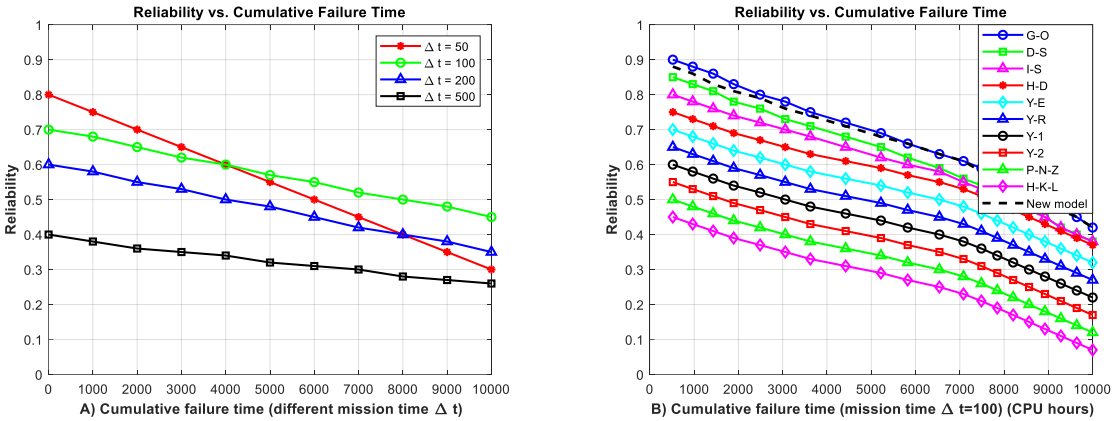


Figure 2. Reliability curves based on the proposed model are illustrated in (a), while (b) presents the reliability curves of various models under a mission time of Δt=100 CPU hours.

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

In this section, a standard fault dataset from IEEE Std 1633™-2016 [30] is used to evaluate the fitting and predictive capabilities of the proposed model. This dataset does not include fault times between failures; instead, the dataset provides the cumulative number of faults along with their corresponding test hours per day. Accordingly, we model the cumulative fault counts as a function of the cumulative test hours. For validation purposes, the first 21 out of 29 fault records are used for model fitting, while the remaining 8 records are reserved for prediction. Additionally, given the mean time between failures (TBF) of 24.9524 in this dataset, the reliability levels of several models are compared using selected TBF values close to this benchmark.

Table 7 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 the Figures 3 and 4. Also, Table 8 presents the values of the fitting and predictions besides other evaluation criteria such as AIC, BIC, SSE, R^2.

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		

Table 6. From the processed failure data, we select the initial 21 samples from the fault dataset [30] for analysis.

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 7. FFANN and CO solutions using the initial 21 samples from the fault dataset [30]

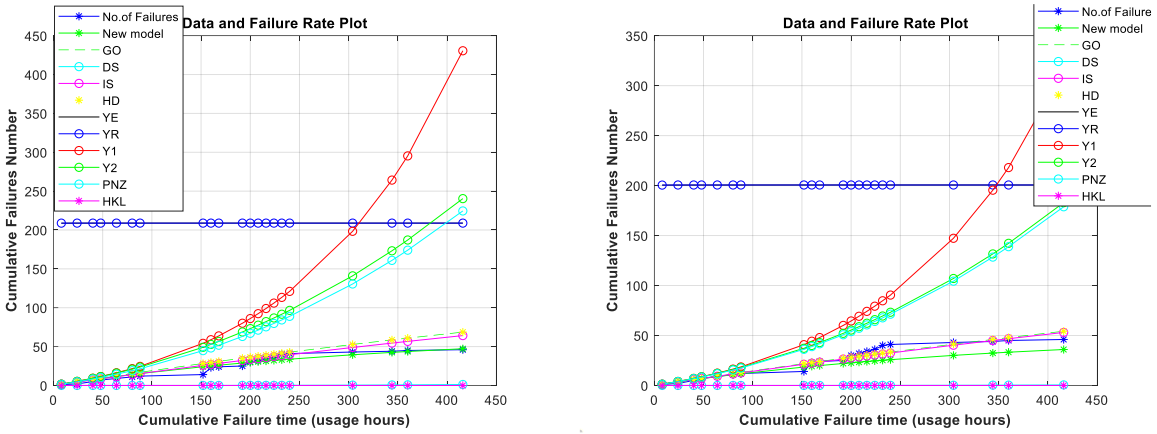
It should be noted based on figures presented in Table 8 that the values of SSE and R^2 of the proposed model is lower than other models. It is seen that since the proposed model has 6 parameters in all, greater than most models except for H-K-L, one has a less favorable weighted evaluation index (AIC 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 figure 3 further complement the quantitative result by visually indicating the predictive performance, that in turn gives an insight of which model is better.

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**

Table 8. Fitting and prediction results derived from different models using the fault dataset, including those using FFANN and CO approaches [30], are presented.

Prediction results can be grouped into two categories: Specifically, models such as G-O and H-D are likely to overestimate contingencies owing to their restricted parameters and less versatility as compared with additional models for instance D-S, I-S, Y-R and H-K-L that on the other hand underestimate contingency trends. In general, these latter models have many more parameters and are very variable with potential Overfitting, resulting in early convergence of the prediction function. The H-K-L model, which has the same number of parameters as the proposed model, has almost negligible changes in the predictions during the last stage.

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 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) FFANN **B)** CO

Figure 3. Fitting and predictive curves of several models, including those developed using FFANN and CO, are based on the first 21 fault data points in the fault dataset [30].

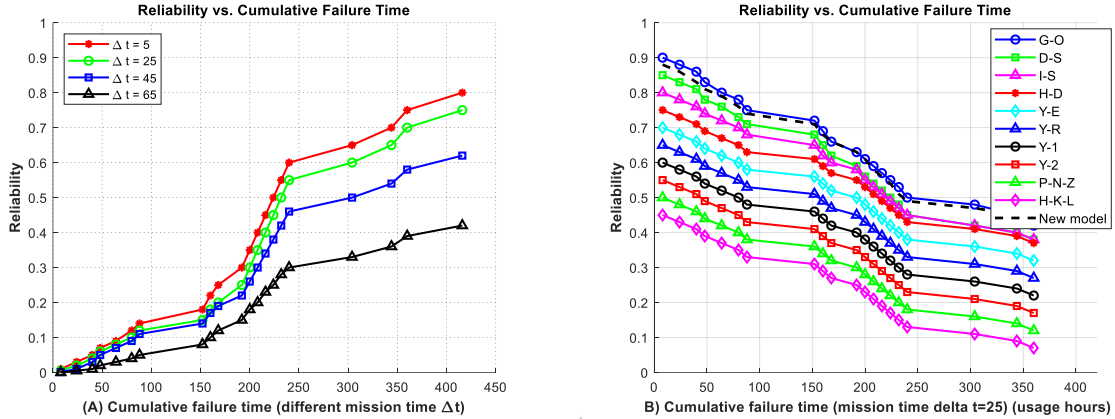


Figure 4. Panel (a) illustrates the reliability curves derived from the proposed model, while panel (b) depicts the reliability curves of various models evaluated under a specified mission time Δt=25 usage hours.

The gradual increase of this parameter as well as the low resultant reliability values shown in Figure 4(a) indicates that the reliability of the software is less than its optimum during the data collection phase of faults. It can be seen in the graph 4(b), the reliability values of the proposed model are at moderate and it is slightly close to the values of Y-E, Y-I-D1 and Y-I-D 2. However, other models such as the G-O and the H-D have comparatively lower reliability coefficients which are on a gradually rising platform.

On the other hand, there is abnormal behavior of D-S, I-S, Y-R and H-K-L models where the reliability values are rising steeply to 1 at the middle testing phase and then again constant at the later phases. Furthermore, some models: H-K-L, P-N-Z, Y-R, I-S, D-S, which are presented in both tables, demonstrate a rather low reliability during the initial test stage, despite a general progression in the subsequent stages.

8. FURTHER WORK

Inherently the nature of most software reliability models appears to have stringent assumptions that cannot perfectly accommodate an actual operating environment. Indeed, in our case, typical software processes, such as fault detection, error correction, and error introduction, exhibit distinct stochastic characteristics in practical applications. In this study, the proportionality coefficient (-p) which establishes the relationship between the error introduction rate and the rate of change of generalized residual errors is not flexibly and

exhaustively represented. In turn, we have identified several ideas for continuation of the current work, which are based on preemptive priority queuing models [31, 32] and queuing-based simulation strategies [33, 34] in particular concerning error modification, learning effect and utilization of testing resources. In particular, we will elaborate on a more rational and dynamic concept of an error introduction rate per a modified error, which would take into account both instantaneous error fluctuation rates and the rates of fault detection with respect to the specific error [35].

Succeeding studies will also explore how errors are generated and fixed (modified) and if the learning effect is experienced during debugging period; how it affects the total debugging time, skill of debuggers, and testing phase. We will discuss variation of faults with detectability and debug-ability, the fact that testing effort and testing time are not directly proportional and other factors. Mitigating these challenges will not only enhance our knowledge database but also inform future reliability model enhancements in subsequent phases of the study.

Furthermore, the parameters that were estimated using FFANN and CO in this study are all point estimates. However, in practical real life most practitioners are interested in interval estimates where the true parameter lies within a given margin of error at a certain level of significance. If the sample conditions are met, FFANN produces super consistent estimation and is asymptotically normally distributed and the above distributions can be used to make confidence intervals about the parameter estimates. In more than one parameter, interval estimation can be derived, at times, from the Fisher information matrix [36-41]. In its standard form, the application of this method is not possible for small or moderate sample sizes since the normality assumption is almost always violated. In similar scenarios, potential solutions will be explored in subsequent research endeavors in order to minimize this weakness effectively.

9. CONCLUSION

The Based on novel modeling perspectives, encompassing the methodology, structural analysis, parameter interpretation, and performance evaluation validation, and comparative evaluation within the scope of this research, several conclusions can be drawn:

(i) Different from ordinary generalized imperfect debugging NHPP models and the proposed LPETM, the present work enhances the LPETM to a superior generalization. It adequately defines the dynamics of an infinite number of faulting scenarios but remains ultimately bounded.

(ii) The developed failure MVF showcases a high capacity for effectively modeling fault data within a specified range and level of detail. Moreover, it demonstrates superior predictive accuracy compared to certain established models in short-term fault forecasting for targeted datasets.

(iii) The proposed model is a relatively effective SRGM. Its predictions generally align well with the actual operational trends of the software. Overall, the SRGM developed in this study enriches the diversity of existing models.

(iv) The FFANN parameter estimation method exhibits superior performance compared to the CO algorithm.

Declarations

- **Ethical Approval:** All the authors demonstrating that they have adhered to the accepted ethical standards of a genuine research study.
- **Competing Interests:** No conflict of interest is declared by authors.
- **Author contributions:** All authors have sufficiently contributed to the study and agreed with the results and conclusions.
- **Funding:** No funding source is reported for this study.

- **Availability of Data and Materials:** The data used to support the findings of this study are included in the article.

RECEIVED: JUNE 2025.
REVISED: FEBRUARY 2026.

REFERENCES

[1] HUANG, Y., CHIU, K., & CHEN, W. (2022): A software reliability growth model for imperfect debugging **Journal of Systems and Software**, 188(111267).

[2] LUO, H., XU, L., HE, L., JIANG, L., & LONG, T. (2023): A novel software reliability growth model based on generalized imperfect debugging NHPP framework **IEEE Access**, 11: 47356-47369.

[3] CHIU, K. C., HUANG, Y. S., & HUANG, I. C. (2019): A study of software reliability growth with imperfect debugging for time-dependent potential errors **International Journal of Industrial Engineering: Theory, Applications and Practice**, 26(3).

[4] GUPTA, R., JAIN, M., & JAIN, A. (2019): Software reliability growth model in distributed environment subject to debugging time lag **Performance prediction and analytics of fuzzy, reliability, and queuing models: Theory and applications** (pp. 105–118). Springer.

[5] PRADHAN, V., DHAR, J., & KUMAR, A. (2023): Testing coverage-based software reliability growth model considering uncertainty of operating environment **Systems Engineering**, 26(4): 449–462.

[6] PRADHAN, S. K., KUMAR, A., & KUMAR, V. (2023): A new software reliability growth model with testing coverage and uncertainty of operating environment **Computer Sciences & Mathematics Forum**, 7(1): 44.

[7] PRADHAN, S. K., KUMAR, A., & KUMAR, V. (2023): A testing coverage based SRGM subject to the uncertainty of the operating environment **Computer Sciences & Mathematics Forum**, 7(1): 44.

[8] HAQUE, M. A., & AHMAD, N. (2023): Software reliability modeling under an uncertain testing environment **International Journal of Modelling and Simulation**, 1–7. Advance online publication.

[9] CHATTERJEE, S., SAHA, D., SHARMA, A., & VERMA, Y. (2022): Reliability and optimal release time analysis for multi up-gradation software with imperfect debugging and varied testing coverage under the effect of random field environments **Annals of Operations Research**, 312(1): 65–85.

[10] LEE, D. H., CHANG, I. H., & PHAM, H. (2020): Software reliability model with dependent failures and SPRT **Mathematics**, 8(8): 1366.

[11] KIM, Y. S., SONG, K. Y., PHAM, H., & CHANG, I. H. (2022): A software reliability model with dependent failure and optimal release time **Symmetry**, 14(2): 343.

[12] RAHEEM, A. R., AKTHAR, S., & RAFI, S. M. (2021): An imperfect debugging software reliability growth model: Optimal release problems through warranty period based on software maintenance cost model **Rev. GEINTEC-GESTAO Inov. E Tecnol.**, 11(4): 4623–4631.

[13] MINAMINO, Y., INOUE, S., & YAMADA, S. (2019): Change-point–based software reliability modeling and its application for software development management. In H. Pham (Ed.) **Recent advancements in software reliability assurance** (pp. 59–92). CRC Press.

[14] KE, S., & HUANG, C. (2020): Software reliability prediction and management: A multiple change-point model approach **Quality and Reliability Engineering International**, 36(5): 1678–1707.

[15] 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.

[16] KUMAR, V., SAXENA, P., & GARG, H. (2021): Selection of optimal software reliability growth models using an integrated entropy–Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) approach **Mathematical Methods in the Applied Sciences**. Advance online publication.

[17] GARG, R., RAHEJA, S., & GARG, R. K. (2021): Decision support system for optimal selection of software reliability growth models using a hybrid approach **IEEE Transactions on Reliability**, 71(1): 149–161.

[18] ZHU, M. (2022): A new framework of complex system reliability with imperfect maintenance policy **Annals of Operations Research**, 312(1): 553–579

[19] YAGHOOBI, T. (2021): Selection of optimal software reliability growth model using a diversity index **Soft Computing**, 25(7): 5339–5353.

[20] SAN, K. K., WASHIZAKI, H., FUKAZAWA, Y., HONDA, K., TAGA, M., & MATSUZAKI, A. (2021): Deep cross-project software reliability growth model using project similarity-based clustering **Mathematics**, 9(23): 2945.

[21] LI, L. (2021): RETRACTED: Software reliability growth fault correction model based on machine learning and neural network algorithm **Microprocessors and Microsystems**, 82: 103894.

[22] AZ-ZO'BI, E., KALLEKH, R., RAHMAN, L., AKINYEMI, A., BEKIR, H., AHMAD, M., & TASHTOUSH, M. (2024): Novel topological, non-topological, and more solitons of the generalized cubic p-system describing isothermal flux **Optical and Quantum Electronics**, 56(1): 84.
[23] BANGA, M., BANSAL, A., & SINGH, A. (2019, April). Implementation of machine learning techniques in software reliability: A framework. **In 2019 International Conference on Automation Computational and Technology Management** (ICACTM) (pp. 241–245). IEEE.
[24] HUSSAIN, A., SULAIMAN, M., HUSSEIN, S., AZ-ZO'BI, E., & TASHTOUSH, M. (2025): Advanced parameter estimation for the Gompertz-Makeham process: A comparative study of MMLE, PSO, CS, and Bayesian methods **Statistics, Optimization & Information Computing**, 13(3): 950-963.
[25] ZUREIGAT, H., TASHTOUSH, M., JAMEEL, A., AZ-ZO'BI, E., & ALOMARE, M. (2023): A solution of the complex fuzzy heat equation in terms of complex Dirichlet conditions using a modified Crank-Nicolson method **Advances in Mathematical Physics**, 2023:, 6505227.
[26] PHAM, H., NORDMANN, L., & ZHANG, Z. (1999): A general imperfect-software-debugging model with S-shaped fault-detection rate **IEEE Transactions on Reliability**, 48(2): 169–175.
[27] PHAM, H. (2014): A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments **Optimization**, 63(10): 1481–1490.
[28] ANJUM, M., HAQUE, M. A., & AHMAD, N. (2013): Analysis and ranking of software reliability models based on weighted criteria value **International Journal of Information Technology and Computer Science**, 19(6): 1–14.
[29] ADEL, S. H., FATAH, K. S., & SULAIMAN, M. S. (2023): Estimating the rate of occurrence of extreme value process using classical and intelligent methods with application: Nonhomogeneous Poisson process with intelligent **Iraqi Journal of Science**, 64(6), 3054–3065.
[30] YANG, X. S., & DEB, S. (2009, December): Cuckoo search via Lévy flights **2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)** (pp. 210–214). IEEE.
[31] YAMADA, S., OHBA, M., & OSAKI, S. (1983): S-shaped reliability growth modeling for software error detection **IEEE Transactions on Reliability**, R-32(5): 475–484.
[32] GOEL, A. L., & OKUMOTO, K. (1979): Time-dependent error-detection rate model for software reliability and other performance measures **IEEE Transactions on Reliability**, R-28(3): 206–211.
[33] YAMADA, S., OHBA, M., & OSAKI, S. (1984): S-shaped software reliability growth models and their applications **IEEE Transactions on Reliability**, R-33(4): 289–292.
[34] HUSSEIN, H., HUSSEIN, S., HUSSAIN, S., & TASHTOUSH, M. (2025): Estimating parameters of software reliability growth models using artificial neural networks optimized by the artificial bee colony algorithm based on a novel NHPP **Mathematical Modelling of Engineering Problems**, 12(1): 25-36.
[35] YAMADA, S., TOKUNO, K., & OSAKI, S. (1992): Imperfect debugging models with fault introduction rate for software reliability assessment **International Journal of Systems Science**, 23(12): 2241–2252.
[36] PHAM, H., & ZHANG, X. (1997): An NHPP software reliability model and its comparison **International Journal of Reliability Quality and Safety Engineering**, 4(3): 269–282.
[37] CHANG, I. H., PHAM, H., LEE, S. W., & SONG, K. Y. (2014): A testing-coverage software reliability model with the uncertainty of operating environments **International Journal of Systems Science: Operations & Logistics**, 1(4): 220–227.
[38] MOHAMMED, B., AWAN, I., UGAIL, H., & YOUNAS, M. (2019): Failure prediction using machine learning in a virtualised HPC system and application **Cluster Computing**, 22(Suppl 1): 471–485.
[39] HUSSAIN, A., ORAIBI, Y., MASHIKHIN, A., JAMEEL, A., TASHTOUSH, M., & AZ-ZO'BI, E. (2025): New software reliability growth model: Piratical swarm optimization-base parameter estimation in environments with uncertainty and dependent failures **Statistics, Optimization & Information Computing**, 13(1): 209–221.
[40] HUSSAIN, A., MAHMOOD, K., IBRAHIM, I., JAMEEL, A., NAWAZ, S., & TASHTOUSH, M. (2025): Parameters estimation of the Gompertz-Makeham process in non-homogeneous Poisson processes: Using modified maximum likelihood estimation and artificial intelligence methods **Mathematics and Statistics**, 13(1): 1–11.
[41] HUSSAIN, A., PATI, K., ATIYAH, A., & TASHTOUSH, M. (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 and Its Applications**, 17(1): 1–15.