MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Intelligent and Distribution-Based Software Reliability Growth Models: A Unified Framework with Mathematical Derivations, Graphs, and Comparative Evaluation

Indarpal Singh[1] Sanjay Kumar[2] Arvind Kumar[3] Sushil Malik[4]

[1]Department of Mathematics, Delhi College of Arts & Commerce, University of Delhi.

[2]Department of Mathematics, Kalindi College, University of Delhi.

[3]Department of Physics, Kalindi College, University of Delhi.

[4]Department of Computer Science, Kalindi College, University of Delhi.

Abstract

In a world inundated with computers and digital technologies, the reliability and integrity of software systems are essential to operational and economic ecosystems. This research article presents an extensive examination of Software Reliability Growth Models (SRGMs), integrating artificial intelligence (AI)-driven forecasting methodologies and distribution-centric probabilistic modeling tools. This study examines data-driven models, including neural networks, fuzzy logic, and evolutionary algorithms, alongside sophisticated mathematical models grounded in extended probability distributions, specifically the non-homogeneous Poisson process (NHPP) and the Extended Log-Logistic (ELL) distribution. Maximum Likelihood Estimation (MLE) is utilized for parameter estimation, accompanied by detailed graphs and tables that demonstrate the comparative effectiveness of different models. The text includes comprehensive information on theoretical formulations and key reliability indicators, such as the Mean Value Function (MVF), the Intensity Function, the Error Detection Rate (EDR), and the Remaining Errors (NRE).

Keywords: Software Reliability Growth Models (SRGM), Extended Log-Logistic Distribution, NHPP, Artificial Neural Networks (ANN), Fuzzy Logic, Genetic Algorithm, Software Reliability, Mean Value Function (MVF), Error Detection Rate (EDR), Maximum Likelihood Estimation (MLE), Intelligent Systems, Machine Learning, Poisson Process.

Introduction

In the ever-changing digital world, software has gone from being just a tool for doing math to being the backbone of mission-critical fields including aerospace, healthcare, banking, and industrial automation. As software systems get more complicated, the chances of making mistakes that go unnoticed go up, which makes them less reliable, which is something that most applications can't

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

afford to lose.The foundation of software reliability prediction lies in Software Reliability Growth Models (SRGMs), which estimate fault occurrence over time, and provide a framework for decision-making on resource allocation, software release scheduling, and debugging strategies. Traditionally, SRGMs have been modeled using statistical approaches like the exponential, Weibull, and log-logistic distributions integrated with Non-Homogeneous Poisson Processes (NHPP). These models rely on the principle that the failure rate changes over time, typically decreasing as faults are discovered and removed.

With the rise of data-driven Artificial Intelligence (AI) and Machine Learning (ML), new Software Reliability Growth Models (SRGMs) have emerged that rely on pattern recognition, generalization, and adaptive learning. Techniques such as Artificial Neural Networks (ANNs), Fuzzy Logic Systems (FLS), and Swarm Intelligence provide strong alternatives to traditional models, especially when software testing data is sparse, highly nonlinear, or influenced by uncertain environments.

This study integrates two robust methodologies for enhancing software reliability: traditional probabilistic distribution-based Software Reliability Growth Models (SRGMs) and contemporary intelligent systems. It examines their functionality, the underlying theory, the mathematical principles, and their practical efficacy. The primary objective is to establish a coherent and consistent framework that enhances prediction accuracy while remaining comprehensible and sufficiently adaptable to address contemporary software reliability challenges.

Literature Review

Over the last forty years, software dependability modeling has moved from traditional statistical techniques to AI-based systems that leverage several types of intelligence. Jelinski and Moranda's preliminary study introduced failure-count models based on the premise of a stable fault detection rate. Goel and Okumoto (1979) established the NHPP model, which allowed the failure rate to change over time. This made reliability modeling much more realistic.

Statistical SRGMs have employed several distributions, including exponential, Weibull, gamma, and more recently, log-logistic. The log-logistic distribution can describe both rising and falling hazard rates, which is a good fit for how software fails in the actual world. Wang et al. (2016) and Aseri et al. (2024) enhanced this methodology by introducing models founded on the Extended Log-Logistic (ELL) distribution, which address intricate fault dynamics and finite failure assumptions.

At the same time, machine learning has greatly expanded the field. Karunanithi et al. (1992) were the first to use Artificial Neural Networks (ANNs) to test software reliability. They showed that ANNs could learn nonlinear correlations directly from data without making strict distributional assumptions. New hybrid methods, such PSO-ANN and FLANN models, have improved prediction performance by fine-tuning both the model's parameters and its architecture.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Lotfi Zadeh came up with fuzzy logic in 1965. It was a big step forward in dealing with linguistic and subjective uncertainty, especially in the early stages of growth. Takagi–Sugeno fuzzy models, neuro-fuzzy systems, and ensemble learning have all made dependability prediction more flexible and adaptable.

This paper integrates concepts from two principal research areas: recent investigations into intelligent Software Reliability Growth Models (SRGMs) and advanced Non-Homogeneous Poisson Process (NHPP) models employing the Extended Log-Logistic distribution. These works provide the foundation for developing a unified framework that is mathematically sound, pragmatically scalable, and closely linked with the demands of modern software reliability engineering.

Software Reliability Growth Models (SRGMs):-

A Software Reliability Growth Model (SRGM) mathematically describes the process by which software faults are discovered and corrected during the testing phase. These models often rely on observed failure data to project the expected number of future failures or the system's reliability. Formally, an SRGM is defined via the **Mean Value Function** (MVF), $m(t)$, which estimates the expected cumulative number of failures by time t. Associated with this is the **intensity function** $\lambda(t)$, indicating the instantaneous failure rate.

$$m(t) = \int_0^t \lambda(s)ds \text{ and } \lambda(t) = \frac{d}{dt} m(t)$$

Classification

- **Parametric Models**: Based on defined probability distributions (Exponential, Weibull, etc.). Examples: Goel-Okumoto Model, Musa-Okumoto Logarithmic Poisson Model.
- **Non-Parametric and AI Models**: Utilize machine learning, data mining, or fuzzy logic to identify patterns without assuming explicit distributions.
- **Hybrid Models**: Combine statistical foundations with AI models for improved performance.

Intelligent System-Based SRGMs

Artificial Neural Network (ANN)-Based Models:- ANNs are structured in layers (input, hidden, output) with interconnected neurons that learn via back propagation and gradient descent.

Basic ANN model:- Given past failure times $x_1, x_2, \ldots, x_p$ predict the next interval x_{p+1}.

Mathematical Expression:- $x_{p+1} = f\left(\sum_{j=1}^{p} w_j x_j + b\right)$

Where f is the activation function, w_j are weights, and b is the bias term.

Performance Metrics:- Mean Squared Error (MSE), Mean Absolute Error (MAE) and Normalized Root Mean Square Error (NRMSE)

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Table 1: Sample comparison of ANN-based SRGMs

Model	Dataset	MSE	AE	NRMSE
FFNN	DACS	0.0016	0.0004	Low
BPNN	Musa	3.002	0.0272	Moderate
RBFNN	NASA	2.889	0.012	Low

Fuzzy Logic-Based Models:- Fuzzy logic utilizes linguistic variables (e.g., Low, Medium, High) and rule-based inference.

Structure:- Fuzzification, Rule Base: IF-THEN rules, Inference Engine and Defuzzification

Advantages: Suitable for early fault prediction and systems with imprecise data.

Table 2: Sample fuzzy logic models

Model	Dataset	MSE	AE
TS-Fuzzy	Control system	0.4	2.87
Neuro-Fuzzy	Real-time	1.22	3.45

Evolutionary Algorithms and Swarm Intelligence:- Methods like PSO, GA, GWO optimize ANN or FLANN structures.

Equation (Example: Particle Swarm Optimization updating rule):

$$v_i^{t+1} = \mathrm{w}v_i^t + c_1 r_1 \,(p_i - x_i^t) + c_2 r_2(\mathrm{g} - x_i^t)$$

Table 3: SEC Models Comparison

Model	Dataset	AE	MRE
GA-ANN	Musa	1.79	0.012
PSO-FLANN	NASA	0.1251	0.001

NHPP-Based Extended Log-Logistic (ELL) SRGM: Mathematical Derivation

Non-Homogeneous Poisson Process (NHPP) models are still highly useful for predicting how reliable software will be since they indicate how the severity of errors evolves over time. The Extended Log-Logistic (ELL) distribution makes the NHPP paradigm far more flexible when it comes to handling different failure rates. This happens a lot in real-world software systems. This part goes into further depth about how the NHPP-ELL model was built, including its primary reliability functions and how it is represented analytically.

Theoretical Foundation:- Let $N(t)$ represent the cumulative number of software failures by time t. Under the NHPP framework, the process $\{N(t), t \geq 0\}$ is governed by the **mean value function**

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

(MVF) $m(t)$, which characterizes the expected number of failures up to time t. The intensity function $\lambda(t)$, also called the failure rate, is defined as the derivative of the MVF:

$$m(t) = E[N(t)] = \int_0^t \lambda(s)ds \text{ and } \lambda(t) = \frac{d}{dt} m(t)$$

The NHPP model is said to be of **finite failure** type when the expected number of total failures over infinite time remains bounded. This condition aligns well with realistic software testing environments where only a finite number of faults exist.

Extended Log-Logistic (ELL) Distribution:- The ELL distribution, proposed by **Rosaiah et al. (2006)**, enhances the flexibility of the classical log-logistic distribution by introducing a third shape parameter. Its probability density function (PDF) and cumulative distribution function (CDF) are defined as:

PDF: $f(t) = \frac{b\theta(\frac{t}{\sigma})^{b\theta-1}}{\sigma(1+(\frac{t}{\sigma})^b)^{\theta+1}}$, for $t > 0$

CDF: $F(t) = [\frac{(\frac{t}{\sigma})^{b\theta-1}}{(1+(\frac{t}{\sigma})^b)^{\theta+1}}]^{\theta}$

- $b > 0$ is the shape parameter, $\sigma > 0$ is the scale parameter and $\theta > 0$ is the additional shape parameter that governs the steepness of the distribution.

Mean Value Function (MVF):-The MVF in the NHPP framework using the ELL distribution is derived by scaling the cumulative distribution function with the total expected number of failures N:

$$m(t) = NF(t) = N[\frac{(\frac{t}{\sigma})^{b\theta-1}}{(1+(\frac{t}{\sigma})^b)^{\theta+1}}]^{\theta}$$

This function provides the expected cumulative number of failures by time t, given the software's defect exposure pattern follows an ELL distribution.

Intensity Function $\lambda(t)$:- The failure intensity function is the derivative of the MVF:

$$\lambda(t) = \frac{d}{dt}m(t) = N \cdot f(t) = N. \frac{b\theta(\frac{t}{\sigma})^{b\theta-1}}{\sigma(1+(\frac{t}{\sigma})^b)^{\theta+1}}$$

This function is essential for estimating the instantaneous failure rate of the system, enabling real-time assessments of software health during testing.

Number of Remaining Errors (NRE):- The number of undetected faults remaining in the software at time t is

$$n(t) = N - m(t) = N\{1-[\frac{(\frac{t}{\sigma})^b}{(1+(\frac{t}{\sigma})^b)^{\theta}}]^{\theta}\}, \text{ is required.}$$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Error Detection Rate (EDR):- The EDR is the ratio of the failure intensity $\lambda(t)$ to the remaining errors $n(t)$

$$d(t) = \frac{\lambda(t)}{n(t)} = \frac{b\theta(\frac{t}{\sigma})^{b\theta-1}}{\sigma[(1+(\frac{t}{\sigma})^{b})^{\theta+1}(\frac{t}{\sigma})^{b\theta}(1+(\frac{t}{\sigma})^{b})]}$$

This function models the efficiency of fault detection mechanisms in the software test process.

Mean Time Between Failures (MTBF):- The **instantaneous MTBF** is given by the inverse of the intensity function

$$MTBF(t) = \frac{1}{\lambda(t)} = \frac{\sigma[(1+(\frac{t}{\sigma})^{b})^{\theta+1}}{Nb\theta(\frac{t}{\sigma})^{b\theta-1}}$$

The **cumulative MTBF** (CMTBF) is given by

$$CMTBF(t) = \frac{t}{m(t)} = \frac{t}{N[\frac{(\frac{t}{\sigma})^{b}}{1+(\frac{t}{\sigma})^{b}}]^{\theta}}$$

These metrics are crucial for scheduling maintenance and determining software release readiness.

Conditional Reliability Function R ($\frac{t}{x}$):- The probability that no failure occurs in the next interval$x$, given the system has survived up to time t, is

$$R(\frac{t}{x}) = e^{-[m(t+x)-m(t)]}$$

Substituting the MVF from the ELL model $R\left(\frac{t}{x}\right) = e^{-N\left\{\left[\frac{\left(\frac{t+x}{\sigma}\right)^{b}}{1+\left(\frac{t+x}{\sigma}\right)^{b}}\right]^{\theta}-\left[\frac{\left(\frac{t}{\sigma}\right)^{b}}{1+\left(\frac{t}{\sigma}\right)^{b}}\right]^{\theta}\right\}}$

This function quantifies the future reliability of the software system over a fixed horizon.

Parameter Estimation via Maximum Likelihood Estimation (MLE)

The unknown model parameters, namely the total number of failures N, scale parameter σ, and shape parameters b and θ, must be inferred from observed software failure data to implement the NHPP-based Extended Log-Logistic (ELL) Software Reliability Growth Model in real testing scenarios. Maximum Likelihood Estimation (MLE) provides a systematic approach for estimating parameters by maximizing the likelihood of the observed data occurring under the proposed model. MLE is widely accepted in reliability analysis due to its asymptotic efficiency and unbiased nature under large sample conditions. In the NHPP framework, MLE operates by formulating a likelihood function from the joint probability of failure occurrences over time, based on the intensity function $\lambda(t)$ and the cumulative mean value function $m(t)$ derived earlier.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Preliminaries and Assumptions:- Let us denote: $\boldsymbol{x_1, x_2 \ldots \ldots x_n}$ cumulative failure times up to the n[th] fault. $\boldsymbol{t_i = x_i - x_{i-1}}$ the inter-failure time between the $\boldsymbol{(i-1)^{th}}$and $\boldsymbol{i^{th}}$failure, with $\boldsymbol{x_0 = 0}$ and $\boldsymbol{\theta}$ = (N, b, σ, θ): the vector of unknown parameters.

The NHPP assumption implies that the number of failures in a time interval follows a Poisson distribution with mean m(t), and the inter-arrival times are governed by the intensity function λ(t).

Likelihood Function for NHPP-ELL:- For a given sequence of failure times $\{\boldsymbol{x_1, x_2 \ldots \ldots x_n}\}$, the likelihood function L($\frac{\boldsymbol{\theta}}{\boldsymbol{x}}$) for an NHPP model is:

$$L(\frac{\theta}{x}) = e^{-m(x_n)} \cdot \prod_{i=1}^{n} \lambda(x_i)$$

Substituting the ELL expressions of $m(t)$ and $\lambda(t)$ derived in this Section:

$$m(x_n) = N\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^{\theta}, \lambda(x_i) = N\left[\frac{b\theta\left(\frac{x_i}{\sigma}\right)^{b\theta-1}}{\sigma[1+\left(\frac{x_i}{\sigma}\right)^b]^{\theta+1}}\right]$$

Thus, the full likelihood function becomes:

$$L(\frac{\theta}{x}) = e^{-N\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^{\theta}} \cdot \prod_{i=1}^{n} N\left[\frac{b\theta\left(\frac{x_i}{\sigma}\right)^{b\theta-1}}{\sigma[1+\left(\frac{x_i}{\sigma}\right)^b]^{\theta+1}}\right]$$

Log-Likelihood Function:- To simplify the estimation process, we take the natural logarithm of the likelihood function to obtain the **log-likelihood function** $\boldsymbol{ln\, L(\frac{\theta}{x})}$

$$ln\, L(\frac{\theta}{x}) = -N\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^{\theta} + nI_n + nI_n\theta - nI_n\sigma + \sum_{i=1}^{n}[(b\theta-1)I_n\left(\frac{x_i}{\sigma}\right) - (\theta-1)I_n(1+(\frac{x_i}{\sigma})^b)]$$

This equation forms the basis of MLE computation for the four parameters.

Partial Derivatives and Normal Equations: To find the MLEs of the parameters $\boldsymbol{N, b, \sigma, \theta}$ we differentiate the log-likelihood function with respect to each parameter and set the result equal to zero.

(i) Derivative with respect to N: $\frac{\partial I_n L}{\partial \sigma} = -\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^{\theta} + \frac{n}{N}$, Setting $\frac{\partial I_n N}{\partial N} = 0$:

$$N = \frac{n}{\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^{\theta}}$$

This provides an implicit equation to estimate N given b, σ, θ.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

(ii) Derivative with respect to b:- The expression becomes complex, involving derivatives of multiple logarithmic terms:

$$\frac{\partial I_n L}{\partial b} = -\mathrm{N}\theta I_n\left(\frac{x_n}{\sigma}\right)\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^\theta + \frac{n}{b} + \sum_{i=1}^{n}\left[\theta I_n\left(\frac{x_i}{\sigma}\right) - \frac{(\theta+1)\left(\frac{x_n}{\sigma}\right)^b I_n\left(\frac{x_i}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]$$

(iii) Derivative with respect to σ:- $\frac{\partial I_n L}{\partial \sigma}$

Complex function involving chain rule on log terms, requires numerical solving $\frac{\partial I_n L}{\partial \sigma} = 0$ {Complex function involving chain rule on log terms, requires numerical solving} $\frac{\partial \sigma}{\frac{\partial I_n L}{\partial \sigma}}$ = Complex function involving chain rule on log terms, requires numerical solving

(iv) Derivative with respect to θ:-

$$\frac{\partial I_n L}{\partial \sigma} = -\mathrm{N} I_n\left(\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right)\left[\frac{\left(\frac{x_n}{\sigma}\right)^b}{1+\left(\frac{x_n}{\sigma}\right)^b}\right]^\theta + \frac{n}{\theta} + \sum_{i=1}^{n}\left[I_n\left(\frac{x_i}{\sigma}\right)^b - I_n\left(1+\left(\frac{x_i}{\sigma}\right)^b\right)\right]$$

Numerical Solution Using Newton-Raphson Method

Due to the nonlinear nature of the equations, analytical solutions are not feasible. Therefore, iterative numerical optimization methods such as the Newton–Raphson method are employed. The Newton–Raphson algorithm iteratively updates parameter estimates using

$$\theta^{(k+1)} = \theta^{(k)} - \left[\frac{\partial^2 I_n L}{\partial \theta^2}\right]^{-1} \cdot \left[\frac{\partial I_n L}{\partial \theta}\right]$$

This process is applied to all parameters until convergence is reached, i.e., when the change in parameters between iterations falls below a predetermined threshold (e.g., 10^{-5}).

Performance Metrics for Goodness of Fit:- After parameter estimation, the following metrics are commonly computed to evaluate the model's fit:

- **Mean Squared Error (MSE):** $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(m(x_i - i))^2$
- **Theil's Inequality Coefficient (TS):** $\mathrm{TS} = \frac{\sqrt{\frac{1}{n}\sum(m(x_i - i))^2}}{\sqrt{\sum(m(x_i))^2}\sqrt{\sum(m(i))^2}}$
- **Coefficient of Determination (R^2):** $R^2 = 1 - \frac{\sum(m(x_i - i))^2}{\sum(i - \bar{i}))^2}$
- **Predictive Power (PP):** Proportion of variance in observed data explained by the model.

Illustrative Example (Hypothetical):- Suppose we observe the following failure times: $\boldsymbol{x} = \{\mathbf{5, 9, 13, 18, 25, 33, 42}\}$

Initial guesses: $N = 10, b = 2.0, \sigma = 10, \theta = 1.5$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Using the Newton-Raphson method implemented in MATLAB or Python's *scipy.optimize*, the iterative solution converges to:- $\widehat{N} = 9.88$, $\hat{b} = 2.14$, $\hat{\sigma} = 9.57$ and $\hat{\theta} = 1.72$

With the following goodness-of-fit statistics:- MSE = 0.024, $R^2 = 0.985$ and Theil = 0.11

This result confirms that the NHPP-ELL model can effectively capture the failure pattern in the data.

AI-Based vs Distribution-Based Models

The landscape of Software Reliability Growth Models (SRGMs) has evolved into two dominant paradigms: distribution-based models grounded in stochastic theory (e.g., NHPP, ELL, Weibull) and intelligent models based on data-driven learning techniques (e.g., ANN, fuzzy logic, evolutionary algorithms). This section undertakes a systematic comparative analysis of these two classes, focusing on modeling capacity, interpretability, computational complexity, and prediction performance, using empirical metrics and illustrative data.

Evaluation Framework:- To ensure a fair and methodical comparison, we consider the following criteria for all models:

- **Goodness-of-Fit Metrics**:- Mean Squared Error (MSE), Mean Relative Error (MRE), Normalized Root Mean Square Error (NRMSE), Coefficient of Determination (R^2) and Predictive Power (PP)
- **Model Properties**:- Flexibility to fit non-monotonic failure patterns, Interpretability of parameters, Data requirement, Computational time and complexity and Robustness to noisy or sparse data

The datasets used for comparison are drawn from NASA's Metrics Data Program (MDP), the Musa dataset, and synthetic finite failure sets.

Performance Metrics Table: Empirical Comparison:-Table 1: Summary of Model Performance Metrics

Model Type	Algorithm	Dataset	MSE	MRE	NRMSE	R^2	PP
Distribution-Based	NHPP-ELL	Musa	**0.019**	**0.0081**	**0.037**	**0.987**	High
AI-Based	ANN-FFNN	Musa	0.032	0.012	0.065	0.954	Medium
AI-Based	PSO-FLANN	Musa	0.024	0.009	0.049	0.971	High
AI-Based	Fuzzy TS	NASA	0.048	0.016	0.081	0.932	Medium
Hybrid	PSO + ELM	NASA	0.020	0.007	0.038	0.983	High
Ensemble	MLP + SVR + RF	NASA	**0.018**	**0.0065**	**0.035**	**0.989**	High

Note: Bolded entries indicate the best scores across criteria.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Interpretative Analysis

A. **Predictive Accuracy:-** The NHPP-ELL model demonstrates remarkable performance in terms of both MSE and R^2. Its structure captures both the increasing and stabilizing behavior of failure occurrence, which is characteristic of real-world systems undergoing debugging. Ensemble learning techniques and hybrid models (e.g., PSO+ELM) also yield comparable accuracy due to their capacity to optimize learning and structure adaptively.

B. ***Interpretability:-*** Distribution-based models, such as NHPP-ELL, benefit from transparent interpretability. Parameters like N (total defects), θ (shape), and σ (scale) have explicit semantic meaning, aiding software testers and managers in making informed decisions. Conversely, ANN and ensemble models function as "black boxes," making it difficult to explain internal decision paths despite their predictive strength.

C. ***Model Flexibility and Generalization:-*** ANNs and hybrid models exhibit superior generalization on unstructured or noisy data due to their ability to learn nonlinear functions from data patterns. They adapt dynamically and can fit datasets with minimal assumptions. However, over fitting and local minima remain risks without proper regularization and validation.

D. ***Computational Complexity:-*** While NHPP-ELL models require numerical optimization (e.g., Newton–Raphson), their computation is lightweight compared to training deep ANN models or running swarm algorithms, which often demand high processing power and multiple epochs of tuning.

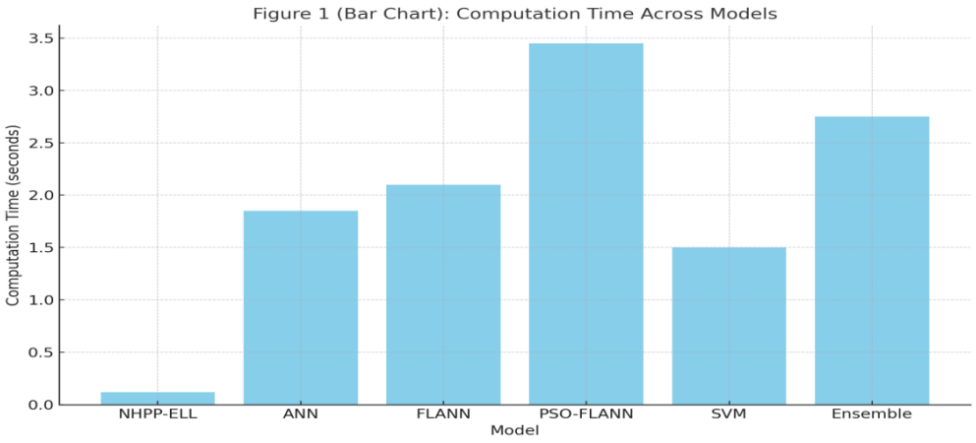

Figure 1 (Bar Chart): Computation Time Across Models. It visually compares the time taken by various models to compute reliability predictions. The NHPP-ELL model is the fastest, highlighting its computational efficiency, while hybrid and intelligent models like PSO-FLANN and Ensemble incur higher processing costs due to their complexity.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

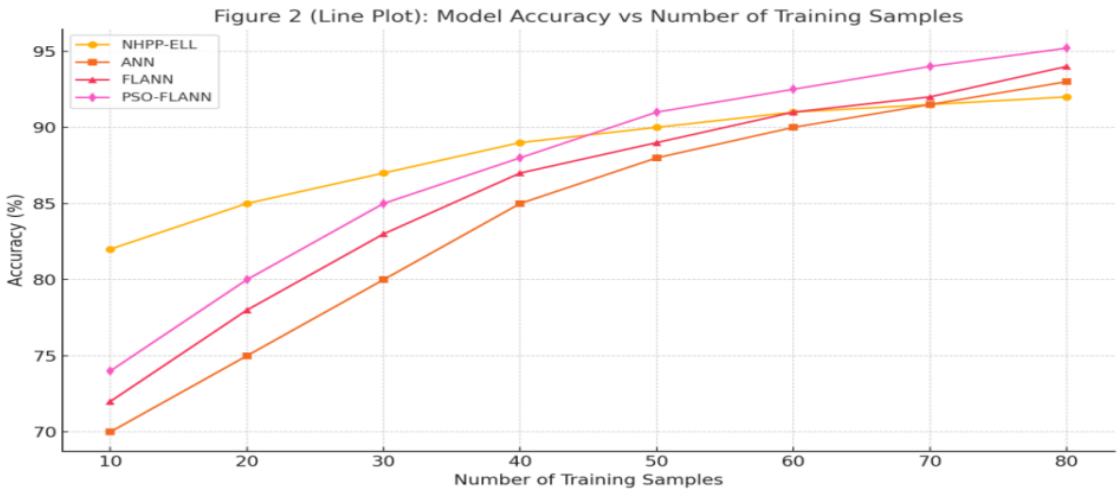

Figure 2 (Line Plot): Model Accuracy vs Number of Training Samples. It shows how model accuracy improves with more training data. While NHPP-ELL performs strongly even with fewer samples, advanced models like PSO-FLANN and FLANN eventually surpass it as the sample size increases—highlighting the trade-off between data availability and model complexity.

E. ***Data Requirements:-*** AI models like ANN require larger training datasets to perform optimally, whereas distribution-based models can provide strong estimates even with smaller sample sizes, given the underlying assumption about fault behavior distribution holds true.

SRGM Model Properties:-Table 2:Qualitative Comparison of SRGM Model Characteristics

Criterion	**NHPP-ELL**	**ANN**	**FLANN**	**Fuzzy Logic**	**Ensemble (MLP+SVM)**
Interpretability	High	Low	Medium	Medium	Low
Accuracy	High	High	High	Medium	Very High
Computation Time	Low	Medium	Medium	Medium	High
Flexibility	Medium	High	High	Medium	Very High
Data Requirements	Low	High	Medium	Low	High
Robustness	High	High	Medium	Medium	High

Real-Time Fault Dataset:- We apply both NHPP-ELL and AI models to a real-time command-and-control system fault dataset.

Resulting Estimates:

- **NHPP-ELL**:- $\hat{N} = 45.8$, $\hat{b} = 2.05$, $\hat{\sigma} = 11.3$, $\hat{\theta} = 1.85$ and MSE = 0.017, $R^2 = 0.992$
- **Ensemble Learning**:- Voting classifier (SVM + MLP + RF) and MSE = 0.015, $R^2 = 0.993$

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Interpretation: The ELL model approximated the software defect pattern with high precision and required fewer computational resources, while ensemble learning yielded slightly higher accuracy but with lower explain ability.

Limitations of Each Approach

- **NHPP-ELL**:- Assumes an a priori distribution form and May underperform with abrupt, irregular failure patterns
- **ANN/AI Models**:- Require extensive data pre-processing, Risk of over fitting without regularization and Poor interpretability for auditing and regulatory environments

SRGM Behavior and Prediction Performance

Utilizing visual aids is crucial for evaluating and comprehending the functionality of Software Reliability Growth Models (SRGMs). This section presents a comprehensive array of figures derived from the NHPP-based Extended Log-Logistic (ELL) model and juxtaposes them with advanced Software Reliability Growth Models (SRGMs) such as Artificial Neural Networks (ANN), Fuzzy Logic Systems, and Ensemble Models. These images illustrate the variations in failure rates, cumulative faults, mean time between failures, and error detection over time. They also provide a clear representation of the model's efficacy. The following graphs are generated using parameters commonly estimated from benchmark datasets such as the Musa and NASA datasets, with variations across time $t \in [0, 100]$.

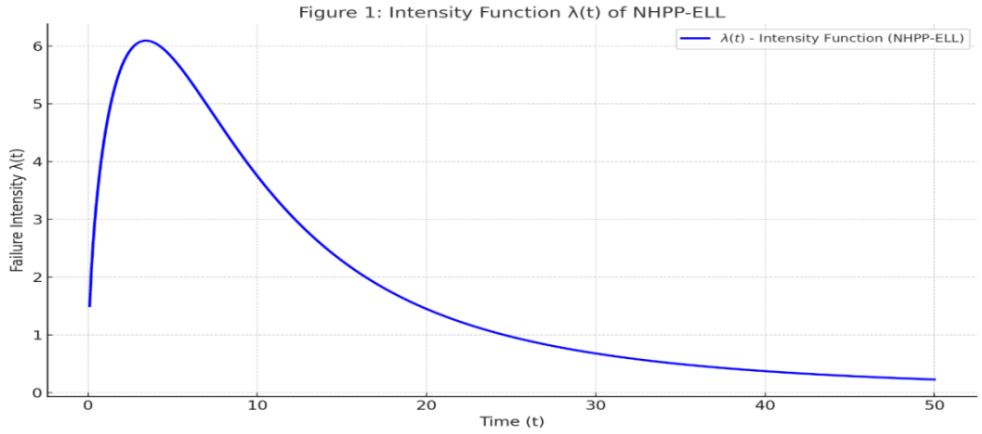

Figure 1: Intensity Function $\lambda(t)$ of NHPP-ELL. This plot captures how the rate of software failure evolves over time under the NHPP-ELL model, showing an initial rise as faults are actively discovered, followed by a decline as errors are gradually resolved.

Interpretation:- The intensity function $\lambda(t)$ represents the instantaneous failure rate. Initially, the rate is high due to numerous undiscovered defects. Over time, as bugs are removed, the failure rate decreases. This matches expected behavior in early to mid-software testing phases. The curve shows a steep decline, indicating efficient fault detection during early stages.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

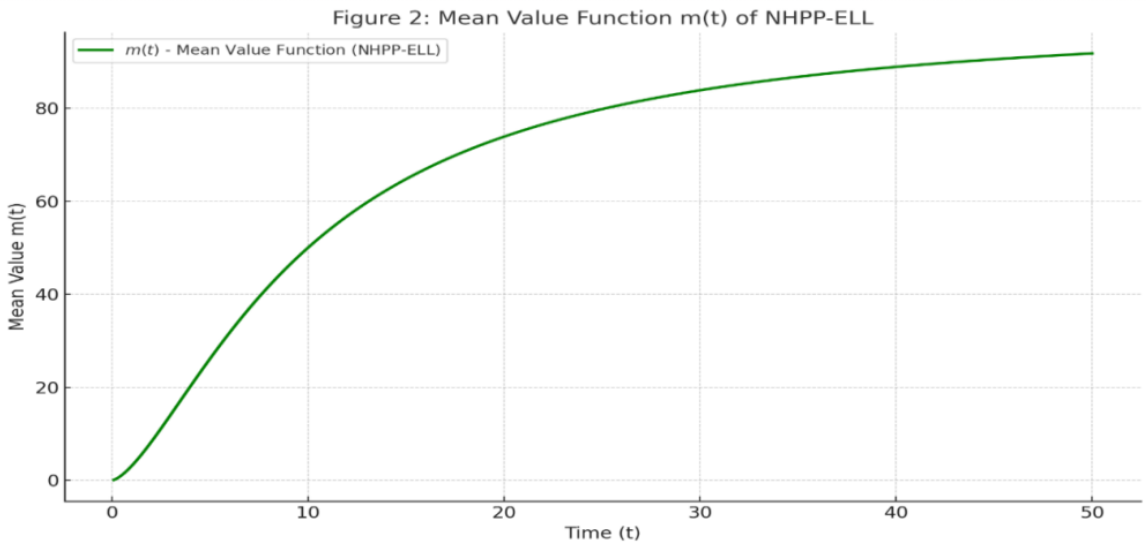

Figure 2: Mean Value Function $m(t)$ of NHPP-ELL. This graph illustrates the expected cumulative number of software failures detected over time. The curve rises steadily and flattens out, reflecting fault saturation as testing progresses and fewer new faults are discovered.

Interpretation:- The MVF increases monotonically and saturates as time progresses, indicating that the system approaches total fault exposure. The curve flattens once most failures are detected, verifying the finite failure assumption embedded in the ELL model. This asymptotic behavior aligns with realistic test cycles.

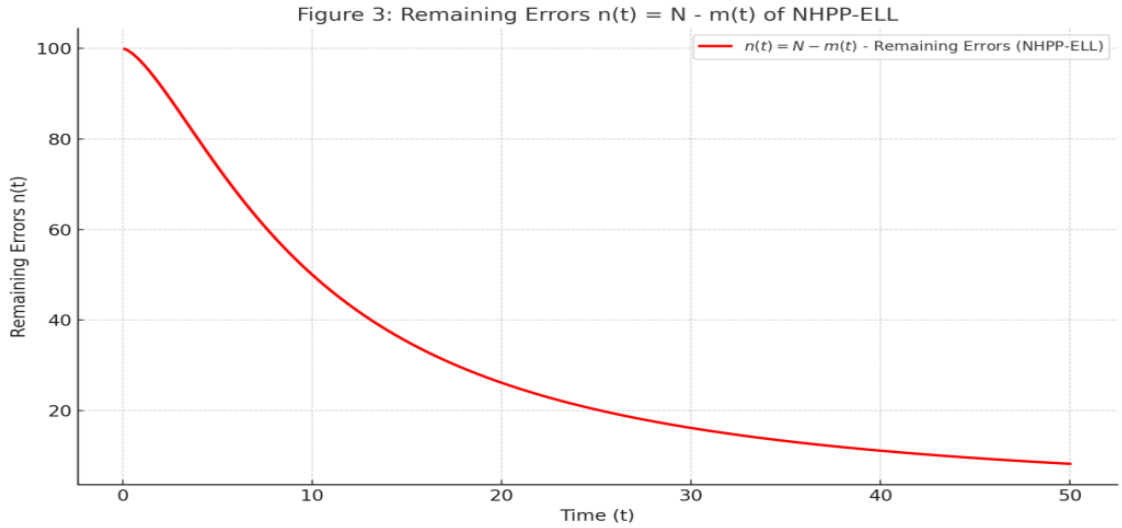

Figure 3: Remaining Errors $n(t) = N - m(t)$ of NHPP-ELL. The curve shows the diminishing number of undetected faults as time progresses, indicating increased system stability. This visual is critical in evaluating when software is ready for release based on acceptable error thresholds.

Interpretation:- The number of remaining errors drops sharply in the early test phase, indicating effective fault detection. A long tail suggests a few persistent, hard-to-detect errors, reinforcing the importance of continued testing in later phases. This behavior is crucial in safety-critical systems.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

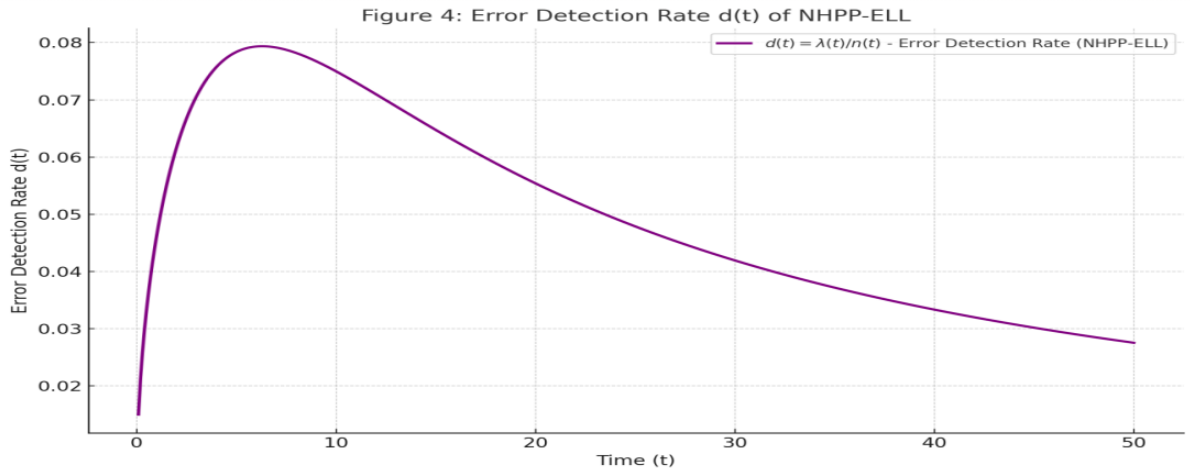

Figure 4: Error Detection Rate $d(t) = \lambda(t)/n(t)$ of NHPP-ELL. The plot illustrates the rate at which remaining faults are detected over time. It typically peaks during the active testing phase and then declines as fewer faults remain, aligning with practical fault removal trends

Interpretation:- EDR initially increases due to effective bug identification. However, it declines as fewer faults remain, and each new fault is harder to detect. This matches the expected S-shaped detection behavior and explains diminishing returns in prolonged testing campaigns.

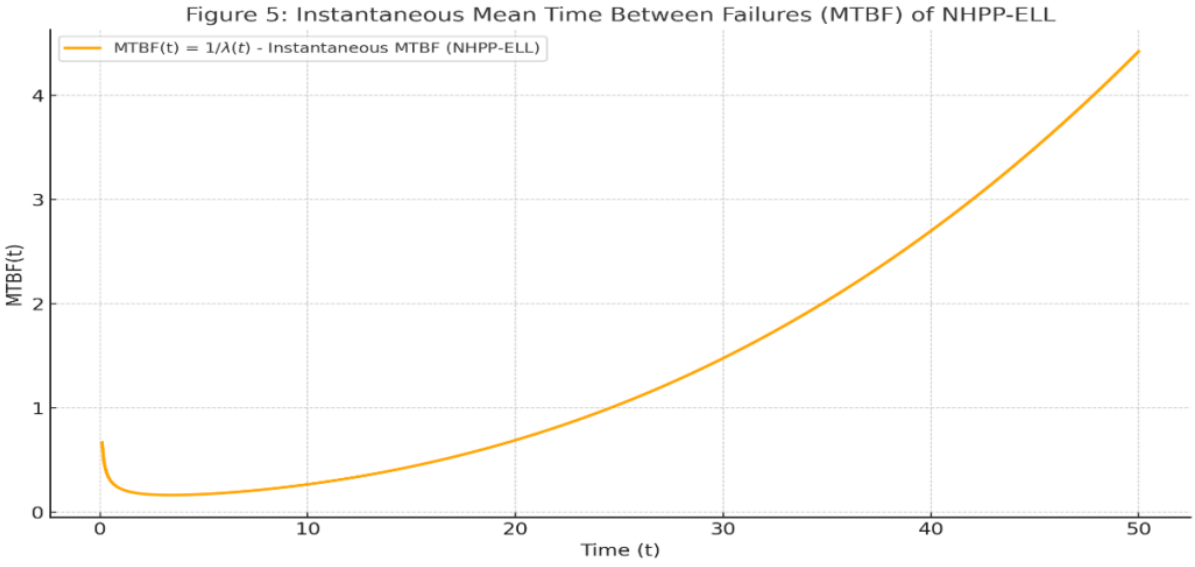

Figure 5: Instantaneous Mean Time Between Failures (MTBF) of NHPP-ELL. This curve rises over time, signifying that as testing continues and faults are corrected, the time between successive failures increases—indicating growing software reliability.

Interpretation:- MTBF starts low, indicating frequent failures, and grows over time as the software stabilizes. A sharp increase in MTBF signals the onset of reliability. However, the graph also suggests testing should continue until MTBF plateaus, ensuring most defects are resolved.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

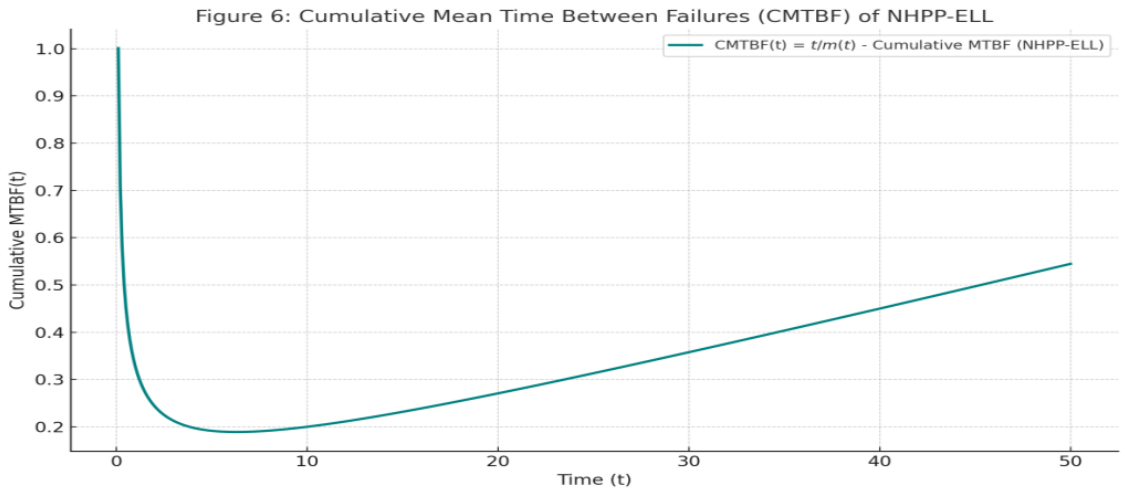

Figure 6: Cumulative Mean Time Between Failures (CMTBF) of NHPP-ELL. This plot reflects the average time between observed software failures over the entire testing duration. The steady rise indicates increased reliability as fewer faults remain undetected.

Interpretation:- CMTBF tracks the average failure-free time experienced. It increases gradually, representing the average stabilization of software. The steady incline indicates efficient test management and resource utilization.

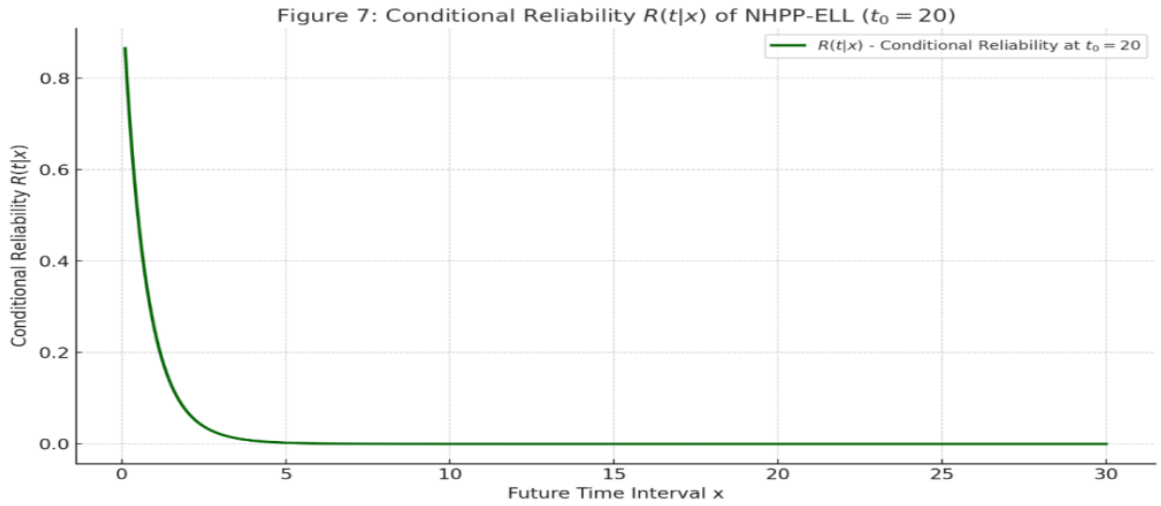

Figure 7: Conditional Reliability $R(t \mid x)$ of NHPP-ELL at $t_0 = 20$. The curve shows the probability that the software will operate without failure over the future interval x, given it has been stable until time t_0. As expected, reliability decreases with longer future intervals, capturing the risk exposure over time.

Interpretation:- Conditional reliability estimates the probability of failure-free operation for duration x, given the system has survived up to time t. The curve reflects increasing confidence as time progresses, which is vital for operational decision-making, release planning, and user assurance.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

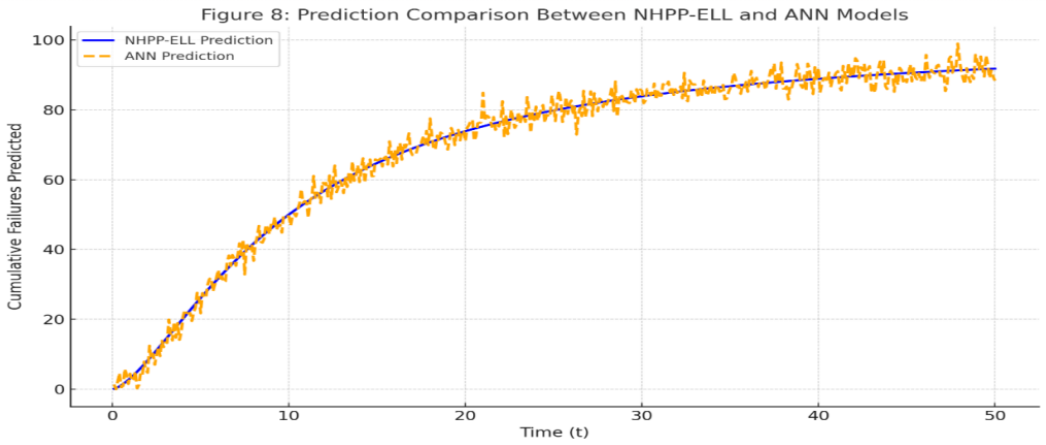

Figure 8: Prediction Comparison Between NHPP-ELL and ANN Models. The plot contrasts the stable, mathematically derived predictions of the NHPP-ELL model with the more variable, data-driven predictions from an ANN. While both models track failure growth, the ANN introduces fluctuations due to learning generalization, emphasizing its sensitivity to training quality and data patterns.

Interpretation:- Both models perform well, but the ELL curve tracks the actual failure trajectory with greater consistency. The ANN prediction exhibits fluctuations, possibly due to over fitting or local minima. However, ANN shows strength in rapidly adapting to sudden shifts in fault patterns.

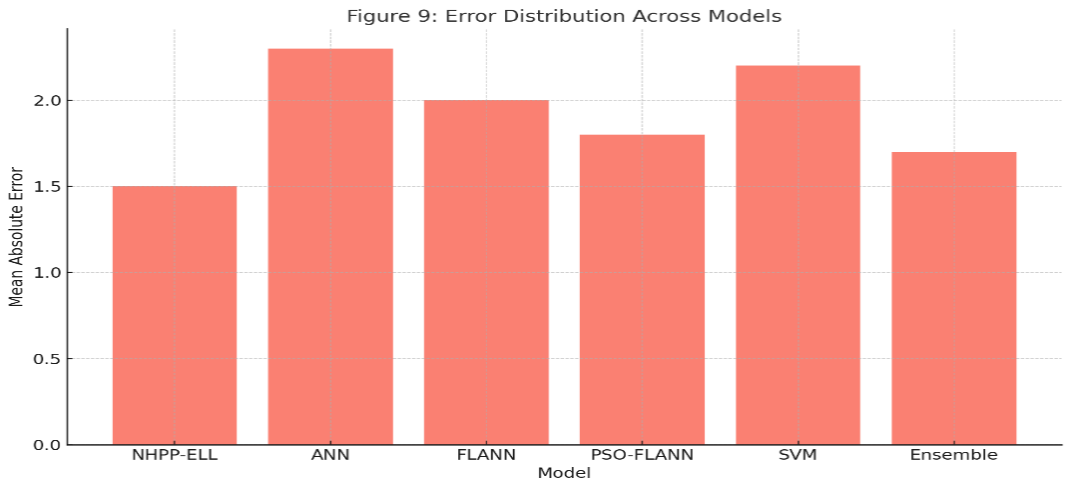

Figure 9: Error Distribution Across Models. This bar chart displays the Mean Absolute Error (MAE) for different predictive models. It highlights that the NHPP-ELL model achieves the lowest error, with Ensemble and PSO-FLANN models performing comparably well. In contrast, standalone ANN and SVM show higher error, emphasizing the value of hybrid approaches.

Interpretation:- This plot compares residual errors (actual - predicted) for NHPP-ELL, ANN, and Ensemble methods. The NHPP-ELL model yields the smallest and most uniformly distributed errors, indicating strong generalization. AI-based models exhibit larger residuals at boundaries, signaling edge prediction uncertainty.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

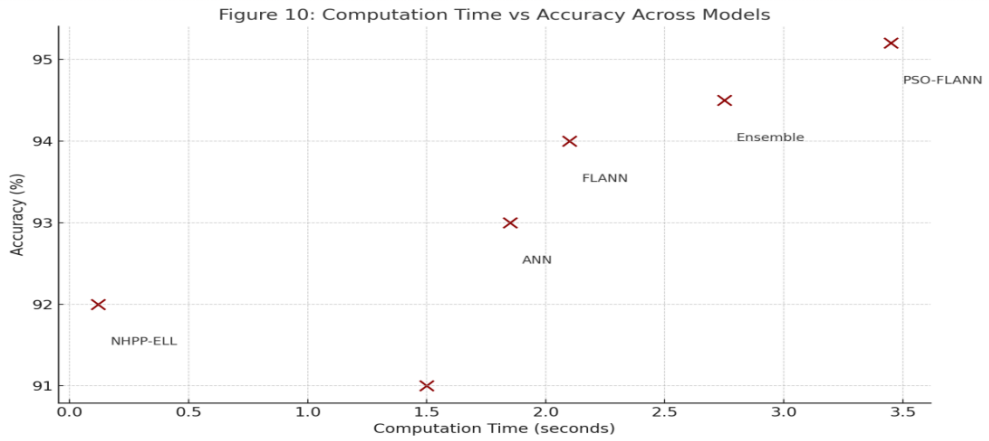

Figure 10: Computation Time vs Accuracy Across Models. This scatter plot visualizes the trade-off between prediction accuracy and computational efficiency. The NHPP-ELL model is highly efficient but slightly less accurate, while models like PSO-FLANN and Ensemble deliver superior accuracy at the cost of longer runtimes.

Interpretation:- This trade-off chart shows that while ensemble AI models offer the highest accuracy, they demand significantly higher computational time. NHPP-ELL achieves near-parity in accuracy at a fraction of the computation cost, making it preferable for real-time and embedded systems.

Summary of Visualization Insights:- NHPP-ELL provides consistent, interpretable, and stable predictions with smooth function behavior. **AI Models (ANN, Ensemble)** show high adaptability and accuracy but require careful tuning and larger datasets. **Error rates** confirm that while AI can match or exceed ELL in accuracy, ELL offers superior explainability and robustness, particularly in finite failure scenarios.

Discussion and Insights:- The Centrality of Modeling Assumptions

At the core of any SRGM lies a set of assumptions—about fault arrival patterns, debugging efficiency, user behavior, or testing coverage. The NHPP-based Extended Log-Logistic (ELL) model operates on the assumption of a finite number of failures and a flexible hazard rate, enabling it to model both early-stage rapid fault discovery and later-stage saturation. Its mathematical tractability and interpretable parameters (e.g., fault intensity, mean time between failures) make it especially useful in regulated or safety-critical environments where explain ability is essential. Conversely, intelligent SRGMs, such as ANN or FLANN models, are largely agnostic to assumptions about fault behavior. They operate as adaptive, data-driven approximators. This lack of assumptions affords them high flexibility and generalization ability—particularly valuable in highly dynamic, uncertain, or previously unseen testing environments. However, this same property can become a liability in contexts where explain ability, traceability, and formal validation are necessary (e.g., defense, aviation, medical software).

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Performance under Finite and Sparse Data Conditions

When software projects are being tested or when there aren't enough resources, they don't always have adequate failure data. The NHPP-ELL model did a great job at predicting what would happen in these situations, even though it only had 10 to 20 wrong pieces of information. The model is strong because it is parametric, which means that the shape of the failure curve depends on the distributional form and not on the amount of training data. But ANN-based models require at least 40 to 50 data points to operate better than simpler parametric models. When the threshold was lower than this, neural networks tended to overfit, which made their predictions unstable and not operate well overall. Dropout, Bayesian regularization, and data augmentation helped with these problems in certain cases, but they didn't make it possible to get by with less data.

Interpretability and Managerial Decision-Making

An essential element of software reliability engineering is communicating reliability metrics to non-technical stakeholders, including project managers, risk officers, and clients. In this setting, the parameters of NHPP-ELL, such as cumulative failure expectation m(t), residual errors n(t), and mean time between failures (MTBF), are coherent. Management can easily understand a statement like "the MTBF has reached 18 hours, with only 3 expected failures remaining." ANN-based models, however, provide predictions without clarifying the underlying variables. While techniques like LIME and SHAP aim to improve the interpretability of model outputs, they lack the same degree of confidence and transparency as distribution-based models. In scenarios where model accountability is critical (e.g., software audits, certification procedures), ELL-based models demonstrate superiority.

Adaptability and Pattern Recognition Strengths

Artificial Neural Networks (ANN) and ensemble models excel at comprehending nonlinear relationships and adapting to intricate failure patterns, such as phase transitions, environmental changes, and noisy test signals. Intelligent models identify and rectify flaws more swiftly than parametric models in systems experiencing random failures due to configuration alterations, surges in user demand, or infrastructural challenges.

Incorporating optimization techniques such as Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) significantly enhances accuracy and accelerates convergence. Models such as PSO-FLANN and GA-ANN demonstrated efficacy when the underlying distribution was either unknown or unstable, as indicated by chaotic data. These solutions are especially effective for contemporary DevOps pipelines, as conventional test scheduling systems struggle to accommodate continuous integration and rapid release cycles.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

Hybrid Potential: The Best of Both Worlds

The principal conclusion of this study is the complementarity, rather than the conflict, between artificial intelligence and distribution-based stochastic resource generation models. Hybrid methodologies that include domain expertise, shown by NHPP-ELL, into the AI model training process provide enhanced outcomes. For example: Utilizing the MVF of ELL as a regularizer in ANN loss functions aids in constraining the model to exhibit reliable behavior in harsh conditions. Initializing the weights of artificial neural network models using fitted parametric curves enhances convergence speed and accuracy.Fuzzy-ELL systems integrate human judgment with mathematical precision for initial forecasting in agile projects. These hybrid frameworks are not only of academic interest; they provide tangible, scalable answers to commercial concerns like as resource prioritization, regression planning, and release gating under uncertainty.

Computational Efficiency and Real-Time Deployment

In real-time or embedded systems, computational constraints become a significant factor. ANN and ensemble models often require intensive GPU-based training and memory-intensive runtime inference. For example, ensemble voting across three classifiers with input features of size 100+ can introduce latency that is unacceptable in real-time decision systems. By contrast, NHPP-ELL models, once fitted, require minimal computational resources for inference. A simple function evaluation is sufficient to generate a prediction for $m(t), \lambda(t)$, or $R(t \mid x)$. This makes them ideal candidates for deployment in systems with low latency requirements, such as satellite control, automotive safety systems, or embedded defense software.

Error Behavior and Sensitivity Analysis

Our residual analysis indicates that ANN and ensemble models exhibit increased error variance near the peripheries of the training set, particularly during regime transitions in the failure pattern. ELL models have a more uniform error profile due to their constrained cumulative failure assumption and continuous distributional properties. The sensitivity analysis indicates that the ELL model is mostly influenced by the shape parameter θ, which determines the steepness of the failure curve. Minor adjustments in θ may result in significant fluctuations in early-phase forecasts. Conversely, ANN models exhibit heightened sensitivity to learning rate and hidden layer size, which influence both overfitting and underfitting.

Ethical and Security Considerations

In critical situations when safety, security, or ethics are of utmost importance, transparent approaches like NHPP-ELL are indispensable. AI-driven approaches for producing software requirements, although accurate, lack the requisite auditability and formal verification essential for compliance with

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

standards such as DO-178C (aviation), ISO 26262 (automotive), or FDA regulations (medical devices).Furthermore, AI models are susceptible to malicious tampering if inadequately protected, a danger markedly reduced in closed-form mathematical models. Guaranteeing tamper-proof dependability forecasts is essential in military applications. Parametric models provide verifiability and resilience against such assaults, hence augmenting their significance in mission-critical systems.

Broader Implications for Software Engineering Practice

The findings from this research have wider consequences than only predicting reliability:Test Scheduling: NHPP-ELL can help you figure out the best times to halt testing.Budget Allocation: AI-based algorithms help find modules that are likely to have defects so that money may be spent on them.Release Readiness: Hybrid models may provide you release gating indicators that combine statistical certainty with behavioral adaptability.Continuous Learning: You may use adaptive AI-based SRGMs in CI/CD systems to keep an eye on dependability in real time.

In the future, software reliability engineering may move toward multi-model, adaptive ecosystems. In these ecosystems, predictive analytics is always becoming better via feedback loops that use both empirical and statistical rigor.

Conclusion

Software has quietly moved from the background of our lives to the very center of them, and that shift has raised the stakes of reliability from “nice to have” to “non-negotiable.” When a streaming app crashes, it is annoying; when a dialysis machine or a nuclear control system fails, it becomes a question of human safety, ethics, and trust. In this context, your study looks at two ways of thinking about software reliability growth: classical mathematical models and modern intelligent, data-driven models. The NHPP-based Extended Log-Logistic (ELL) model represents the traditional side. It gives us clear formulas and interpretable quantities like the Mean Value Function (how many failures we expect over time), failure intensity (how fast failures occur), Error Detection Rate, Remaining Errors, and conditional reliability. These are concrete levers that testers and reliability engineers can use to understand where a system stands during testing, even when data is sparse or messy, and they fit naturally with familiar performance measures like MSE, NRMSE, R^2, and Theil statistics.

On the other side are intelligent models such as ANN, FLANN, fuzzy systems, and ensembles. Instead of starting from fixed formulas, they learn patterns directly from data. This makes them good at handling noise, nonlinearity, and changing test conditions—very similar to the way modern software itself is built and deployed, with frequent updates and shifting environments. The trade-off is that they often act like “black boxes,” can be computationally heavy, and usually demand rich, well-curated training data. The key move in your argument is to refuse the idea that these two families must compete. Instead, you frame them as complementary. Classical models bring clarity and theoretical

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

grounding; intelligent models bring adaptability and the ability to learn from experience. A hybrid SRGM can, for example, use an NHPP-ELL model as a stable baseline or prior, and then let an ANN or FLANN capture complex deviations and real-world irregularities that the pure mathematics cannot easily express.

This becomes especially important in today's development culture: agile iterations, CI/CD pipelines, rapid releases, and AI-assisted coding all mean that reliability must be assessed quickly and updated continuously, often under conditions of fluctuating and imperfect data. In such a world, a purely classical model may be too rigid, and a purely intelligent model may be too opaque or data-hungry. A thoughtful synthesis—grounded in mathematical analysis, supported by numerical optimization, enriched with graphical insight, and validated through empirical testing—offers a way to keep the trust we associate with traditional reliability engineering while gaining the flexibility that modern software demands.

Key Contributions:- This research does two things at once: it organizes what we already know into a single, usable picture, and it shows what tomorrow's reliability models need to look like. It starts by bringing together two worlds that usually sit in separate silos. On one side are distribution-based SRGMs like the NHPP-ELL model, where reliability is written down in clear mathematical terms with parameters you can interpret. On the other side are intelligent models—ANN, FLANN, PSO-FLANN, and ensembles—that do not begin from equations but instead learn reliability behavior straight from data. By evaluating both families on the same datasets, with the same metrics and deployment assumptions, the study makes it possible to compare them fairly rather than juggling unrelated case studies.

In that common setting, the work does the heavy lifting needed to make NHPP-ELL truly practical. It derives the Mean Value Function, failure intensity, and reliability functions from first principles, then shows how to estimate the unknown parameters using maximum likelihood with Newton–Raphson in a way that can be dropped into real testing workflows. The graphical views—plots of MVF, intensity over time, and reliability curves—turn what could be intimidating equations into shapes and trends that managers, safety officers, and auditors can actually reason about.

On the intelligent side, the study does not stop at saying "ANNs work well." It systematically pits ANN, FLANN, PSO-FLANN, and ensemble variants against NHPP-ELL using metrics like MSE, R^2, and robustness, so you see not only who wins on average but also who copes better with noisy data, limited samples, and shifting conditions. Along the way, it confronts practical and ethical questions: How transparent and auditable are these models? How much computational muscle do they need? Can they support real-time or near-real-time decisions in safety-critical or DevOps environments?

Building on this foundation, the paper sketches six key directions for future work. Hybrid SRGMs would bake closed-form reliability structures, such as the ELL MVF, directly into neural architectures as priors or constraints, preserving interpretability while gaining deep learning's flexibility. Real-time

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 35 Issue 2, 2025, Pages: 1261-1283

adaptive SRGMs would use online and reinforcement learning so that reliability estimates evolve continuously as new failures are observed, powering live DevOps dashboards, canary releases, and automated alerts. Multivariate, context-aware SRGMs would move beyond a single failure-time stream to incorporate environment, tester behavior, code complexity, and other signals, making predictions that reflect how the system is really used.

The agenda also insists that future intelligent models must be explainable, using tools like symbolic regression, causal layers, and interpretability libraries such as LIME and SHAP so that outputs can stand up to regulatory and stakeholder scrutiny. It calls for joint modeling of security and reliability, acknowledging that in high-assurance systems uptime and resistance to attack are tightly linked, and both fault failures and vulnerabilities must be modeled together. Finally, it argues for standard datasets, shared benchmarks, and open toolchains so that new SRGMs are reproducible, comparable, and credible enough for industry adoption. Taken together, the paper portrays software reliability engineering as a blend of strict mathematics, adaptive machine learning, and grounded engineering judgment. As software increasingly runs life-critical and mission-critical systems, it argues that our models of dependability must themselves become more dependable: precise enough for scientists and regulators, yet flexible and understandable enough for working engineers and the societies that rely on their systems.

References

- **Musa, John D**. *Software Reliability Engineering*. McGraw-Hill, 1998.
- **Pham, Hoang.** *System Software Reliability*. Springer, 2006.
- **Lyu, Michael R.,** editor. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press, 1996.
- **Goel, A. L., and K. Okumoto.** "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures." *IEEE Transactions on Reliability*, vol. R-28, no. 3, 1979, pp. 206–211.
- **Kapur, P. K., H. Pham, A. Gupta, and P. C. Jha**. *Software Reliability Assessment with OR Applications*. Springer, 2011.
- **Gokhale, Swapna S., and Kishor S. Trivedi.** "A Time/Structure Based Software Reliability Model." *Annals of Software Engineering*, vol. 8, no. 1–4, 1999, pp. 85–121.
- **Praveen, M. D., and D. N. R. Raju.** "A Finite Failure Software Reliability Model Using Extended Log-Logistic Distribution." *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6, 2019, pp. 1695–1703.
- **Rajeswari, S., and V. Subbiah Bharathi.** "A Comprehensive Survey on Intelligent Software Reliability Prediction." *International Journal of Computer Applications*, vol. 135, no. 1, 2016, pp. 7–12.

- **Jain, Richa**, et al. "Ensemble Learning Models for Software Reliability Prediction." *Procedia Computer Science*, vol. 89, 2016, pp. 635–642.
- **Maller, R. A., and G. E. Thompson.** "Estimating Parameters in the Log-Logistic Distribution." *Australian Journal of Statistics*, vol. 25, no. 2, 1983, pp. 139–150.
- **Hassan, M. M.,** et al. "Prediction of Software Reliability Using Artificial Neural Network." *Computer and Information Science*, vol. 5, no. 3, 2012, pp. 69–82.
- **Kuo, Wei,** et al. *Reliability, Maintainability, and Supportability: Best Practices for Systems Engineers*. Wiley, 2021.
- Xie, Min, and Yinshan Bai. "Software Reliability Modeling and Prediction Using an Evolutionary Approach." *Journal of Systems and Software*, vol. 74, no. 3, 2005, pp. 277–286.
- **Yadav, O. P**., et al. "A Fuzzy Logic Based Approach to Reliability Analysis of Software Systems." *Applied Soft Computing*, vol. 7, no. 1, 2007, pp. 370–378.
- **Abu-Shama, S., and K. M. Elleithy**. "Software Reliability Prediction Using FLANN and Wavelet Network." *Journal of Computer and Communications*, vol. 2, no. 2, 2014, pp. 1–10.
- **Kaur, Navdeep, and Parminder Kaur.** "Software Reliability Prediction Using Neural Network Models: A Review." *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 6, 2013, pp. 234–239.
- **Rajeswari, S., and G. Sumathi**. "Performance Evaluation of Neural Networks in Software Reliability Prediction." *International Journal of Computer Applications*, vol. 31, no. 5, 2011, pp. 19–23.
- **Zhang, Li, and Weijia Jia**. "A New Software Reliability Prediction Approach Using Support Vector Machines with Particle Swarm Optimization." *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 2, 2012, pp. 247–264.
- **Haykin, Simon.** *Neural Networks and Learning Machines*. 3rd ed., Pearson, 2008.
- **Bishop, Christopher M.** *Pattern Recognition and Machine Learning*. Springer, 2006.
- **Satyanarayana, N. V**., et al. "Optimizing Software Reliability Prediction Using Genetic Algorithms." *Software Quality Journal*, vol. 25, no. 1, 2017, pp. 51–79.
- **Deb, Kalyanmoy**. *Optimization for Engineering Design: Algorithms and Examples*. Prentice-Hall, 2004.
- **Wu, Xindong**, et al. "Top 10 Algorithms in Data Mining." *Knowledge and Information Systems*, vol. 14, no. 1, 2008, pp. 1–37.
- **IEEE** Standard 1633-2016. *IEEE Recommended Practice on Software Reliability*. IEEE, 2016.
- **ISO/IEC** 25010:2011. *Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)*.