Google Scholar Crossref doi scopus

Impact factor 6.2

Geoscience Journal

ISSN:1000-8527

Indexing:

- Scopus
- Google Scholar
- DOI, Zenodo
- Open Access

www.geoscience.ac

Registered

Development of Hybrid Software Reliability Models Integrating S-Shaped and Logarithmic Growth with Dynamic Change Points

Anil Kumar Singh & Kaushal Kumar

University Department of Statistics and Computer Applications

T.M. Bhagalpur University, Bhagalpur, India

Abstract

Software reliability is a crucial component of overall software quality, particularly in mission-critical and commercial software systems. Classical software reliability models often fail to capture the multifaceted dynamics of real-world failure patterns, particularly in the presence of fluctuating testing efforts and environmental shifts. In practice, software projects experience fluctuating testing efforts, process improvements, shifts in operational environments, and evolving fault profiles over time. Such variations introduce complex, multi-phase failure behaviors that classical models struggle to capture, motivating the need for more flexible modeling frameworks capable of integrating diverse growth patterns and accounting for dynamic change points in the software testing lifecycle. This paper proposes a hybrid software reliability growth model (SRGM) that integrates the characteristics of delayed S-shaped and logarithmic growth models while incorporating dynamically identified change points. The suggested model aims to enhance the precision and adaptability of reliability estimates. Through mathematical formulation, parameter estimation techniques, and empirical validation, this study demonstrates the advantage of the hybrid model over conventional SRGMs.

Keywords: Software Reliability Growth Model (SRGM), Hybrid Reliability Model, Change Point Detection, S-Shaped and Logarithmic Growth, Maximum Likelihood Estimation (MLE)

1. Introduction

Software reliability, defined as the likelihood of failure-free performance over a designated timeframe under specific conditions, is crucial in software engineering.

Throughout the decades, various Software Reliability Growth Models (SRGMs) have been introduced, each aiming to fit observed failure data more accurately. Nonetheless, these models frequently encounter difficulties with actual datasets due to their inability to adapt to evolving test conditions or shifts in testing strategies. Notably, the delayed S-shaped model captures delayed fault detection effectively, while the logarithmic model excels at modeling early rapid fault detection.

Most SRGMs presented in the literature presume that the fault correction rate is invariant over time. However, in practice, the rate of fault rectification is affected by various factors, including the proficiency the debugging team, the complexity of the identified defects, the debugging environment, and the tools employed. Consequently, the fault correction rate is often neither constant nor smooth; rather, it may vary at specific points in time, known as change points (CPs) [Huang, 2005; Inoue and Yamada, 2008].

A change point represents the moment when a model parameter exhibits a discontinuity over time. Zhao (2003) suggested that change points may occur due to modifications in testing strategies or resource allocation. Additionally, variations in testing facilities and other random external factors can also trigger such changes. Over the years, several change-point-based models for software reliability have been proposed, including the Weibull change-point model, the Jelinski–Moranda de-eutrophication model with a change point, and the Littlewood model with one change point [Zhao, 2003].

A change point signifies the instance at which a model parameter displays a discontinuity over time. Zhao (2003) proposed that change points may arise from alterations in testing methodologies or resource distribution. Moreover, discrepancies in testing facilities and other arbitrary external factors may also induce such alterations. Numerous change-point-based models for software dependability have been introduced over the years, such as the Weibull change-point model, the Jelinski–Moranda de-eutrophication model using a change point, and the Littlewood model incorporating a single change point [Zhao, 2003].

This paper introduces a hybrid model combining the strengths of both and dynamically integrating change points to reflect shifts in testing policies, resource allocations, or environmental conditions. This research is timely and necessary, especially given the rise of adaptive, continuous deployment environments where static assumptions rarely hold.

2. Literature Review

The Jelinski-Moranda (J-M) model and the Goel-Okumoto (G-O) non-homogeneous Poisson processes (NHPP) model laid the foundation for SRGMs. The S-shaped model, introduced by Ohba (1984), acknowledges the learning curve of the testing team, making it suitable for systems with delayed fault detection. Conversely, the logarithmic model

proposed by Musa et al. (1987) reflects rapid initial fault detection, tapering off over time.

Kimura et al. (1992) introduced a novel approach to software reliability growth modeling by distinguishing between two classes of software errors: easily detectable and difficult-to-detect errors. By NHPPs to represent the error detection rates of these two classes, where the aim to provide a more realistic representation of software reliability growth during testing.

Shyur (2003) presented a SRGM that jointly incorporates the effects of imperfect debugging and the change-point problem within the software testing process.

Huang (2005) proposed a novel approach to constructing SRGMs using a NHPP. The model focussed on developing a method that integrates both testing effort and change-point considerations into software reliability modeling, addressing limitations in existing models.

Zhao et al. (2006) presented a novel SRGM called the generalized inflection S-shaped model. They have provided a finite Poisson process that offers great flexibility and possesses two distinct characteristics.

Lin & Huang (2008) developed SRGMs that incorporate multiple change-points into Weibull-type testing effort functions in order to better capture real-world testing-resource allocation dynamics.

Huang and Hung (2010) challenged a major limitation of traditional SRGMs- the assumption that detected faults are removed instantly. In practice, fault correction is a multi-step process involving failure reproduction, cause identification, fixing, and re-testing, all of which take time. Consequently, the fault-removal rate is not constant and may shift at specific stages of the development process.

Li et al. (2010) investigated the sensitivity of software release time in the context of Software Reliability Growth Models (SRGMs) that incorporate testing effort functions.

Li et al. (2011) highlighted that incorporating the time distribution of testing effort into SRGMs improves reliability estimation, as testing effort strongly shapes the reliability growth curve. They introduced two enhanced models—DSTEF-SRGM and ISTEF-SRGM—by integrating S-shaped testing effort functions into traditional SRGMs to better capture testing dynamics and improve prediction accuracy.

Huang and Lyu (2011) proposed a simple and effective method for forecasting and assessing software reliability across both the testing and operational phases, recognizing that debugging continues before and after release.

Ahmad et al. (2011) examined the way to incorporate the Log-logistic TEF into inflection S-shaped SRGMs based on NHPP in the context of imperfect debugging. The models parameters are estimated by least square estimation (LSE) and maximum likelihood estimation (MLE) methods. The methods of data analysis and comparison criteria are presented.

Ahmad et al. (2010) examined exponential Weibull (EW) testing-effort functions and developed inflection S-shaped SRGMs that incorporate EW-based effort to improve fault-detection accuracy. The models were evaluated under an incomplete-debugging environment, where some detected faults remain uncorrected.

Chatterji et al. (2012) introduced a SRGM formulated under the NHPP framework. The model emphasizes the joint effect of time-dependent fault introduction and fault removal rate while integrating the concept of a change point in the software testing phase.

Syuan et al. (2014) extended traditional SRGMs by incorporating a Parr-curve TEF with multiple change-points. This reflects real-world variations in testing intensity, producing more accurate reliability predictions than models with fixed Weibull or Rayleigh TEFs.

Sagar et al. (2016) employed Weibull distribution and inflection S-shaped SRGM to estimate and fix software problems. This article uses two-parameter Weibull distribution.

Chatterji and Shukla (2016) proposed a SRGM that incorporates a time-variant fault detection probability into an S-shaped coverage model to improve the accuracy of software reliability estimation.

Song et al. (2017) explored software reliability, especially given the societal repercussions of software failures. Software reliability has been a top priority in software development as it becomes more important in life.

Minamino et al. (2019) investigated the optimal software release problem to determine when to release software, balancing quality, cost, dependability, and user pleasure. Author mentions two techniques in this model: 1. Change–Point Model, which recognizes a software testing process shift like a failure rate change. 2. Multi-Attribute Utility Theory (MAUT) helps make judgments with conflicting goals.

Ke and Huang (2020) examined the mathematical features of the suggested model and evaluated its applicability and performance using real software failure data. They also presented a cost-reliability-based optimal software release approach to reduce software development costs while meeting reliability goals.

Shrivastava and Sharma (2022) created a novel SRGM that considers fault distribution functions before and after the change point. A hybrid SRGM with a change-point is

developed by considering alternative fault distribution functions before and after the change-point.

Fang and Hsu (2025) suggested developing SRGMs with C.I. using Ohba's Inflection S-shaped model and Yamada's delayed S-shaped model to help developers choose the best release time.

Many studies have incorporated change points, recognizing that failure intensity is not static throughout the testing lifecycle. However, few models simultaneously leverage different growth behaviors and dynamic adjustments. This gap motivates the current research.

3. Methodology

3.1 SRGM with Change Point

Most of the early SRGMs were developed on the assumptions that the fault detection occurs either at a constant rate or follow a monotonic function. Some of these model includes, J-M model (Jelinski-Moranda (1972)), G-O model (Goel and Okumoto (1979)), modified G-O model (Yamada et al. (1983)), logarithmic model (Musa et al. (1987)), etc. Failure being a random phenomenon it is expected that failure intensity is a continuous function of time. That's why the above discussed models were proposed with continuous failure intensity rate. As per the standard SRGM assumptions, the fault detection process is either smooth or monotonic, but the real processes have jumps. A change-point allows the model to reflect this shift. Since, during testing process, fault detection rate can be influenced by multiple factors including the testing strategy, testing resource allocations, etc., so including a change point enhances the efficiency of software reliability models in tracing closely the reality.

Chang (1997) considered the change-point problems in the NHPP SRGMs. The parameters if the NHPP with change-point models are estimated by the weighted least square method. Let the parameter τ be the change-point that is considered unknown and is to be estimated from the data. The fault detection rate function is defined as

$$b(t) = \begin{cases} b_1, & \text{for } t \leq \tau, \\ b_2, & \text{for } t \geq \tau. \end{cases}$$

By the assumptions, the mean value function, $m(t)$, can be derived as

$$m(t) = f(x) = \begin{cases} a(1 - e^{-b_1 t}), & \text{for } t \leq \tau, \\ a\left(1 - e^{-b_1 \tau - b_2 (t-\tau)}\right), & \text{for } t \geq \tau. \end{cases}$$

3.2 Hybrid Model Formulation

In software reliability analysis, the mean value function $m(t)$ plays a crucial role. It represents the expected cumulative number of software failures that will have occurred by time *t*. This function forms the basis of most SRGMs.

In real-world software testing, failure detection doesn't always follow a single pattern. Often, it changes over time due to learning by the testing team, shifts in resource allocation, change in testing strategies, new testing tools, etc. To capture this complex behavior more realistically, the hybrid model combines two well-known growth patterns such as delayed S-shaped growth before a change point, and Logarithmic growth after the change point. The Hybrid Model is given below:

$$m(t) = \begin{cases} A(1-(1+bt)e^{-bt}), & \text{for } t \le \tau \\ A\log\big(1+c(t-\tau)\big), & \text{for } t > \tau \end{cases}$$

where *A* is total expected number of failures in the software system, *b* and *c* are the parameters that control the growth rate before and after the change point, respectively, τ is the change point – the time at which the testing behavior changes, and *t* is the time since testing began.

Behavior before the Change Point ($t \le \tau$):

The model uses:

$$m(t) = A(1-(1+bt)e^{-bt})$$

This is an S-shaped model (similar to the Yamada-Ohba model), which is designed to reflect the initial learning curve of the testing process.

Behavior after the Change Point ($t > \tau$):

The model switches to:

$$m(t) = A\,log(1 + c(t - \tau))$$

This is a logarithmic model, often used when a significant update or improvement changes the detection capability or a new testing strategy (e.g., automation) is deployed or resources are ramped up, enabling faster yet steady fault detection.

3.2 Model Curve Visualization

The chart below illustrates the behavior of the hybrid model, highlighting the transition from S-shaped to logarithmic growth at the change point ($\tau = 20$).

Table 1: Sample Output of Hybrid Model (Selected Points)

Time (t)	Expected Failures $m(t)$	Phase
0.0	0.00	S-shaped
5.0	9.00	S-shaped
10.0	26.00	S-shaped
15.0	44.00	S-shaped
20.0	59.00	S-shaped
25.0	125.00	Logarithmic
30.0	179.00	Logarithmic
35.0	214.00	Logarithmic
40.0	239.00	Logarithmic
45.0	260.00	Logarithmic
50.0	277.00	Logarithmic

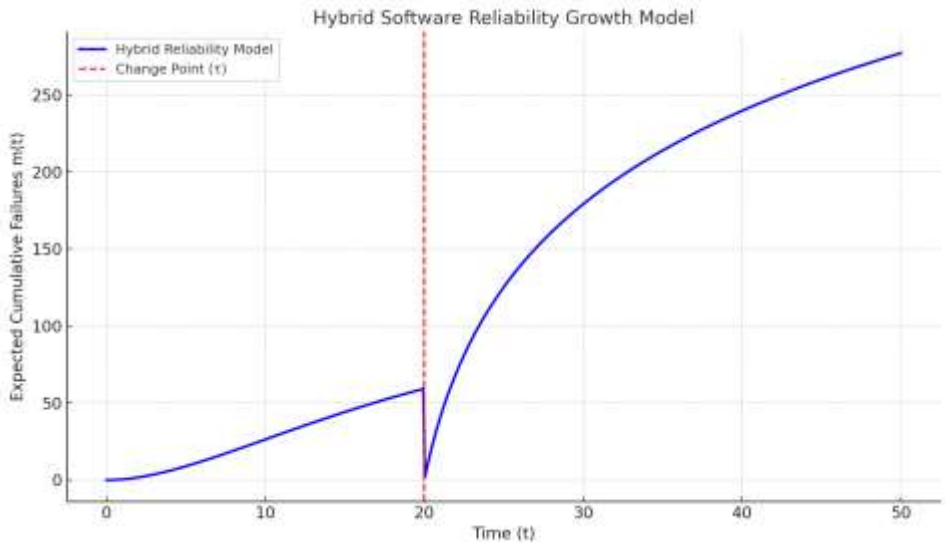

Figure 1. Expected cumulative failures vs. time

3.3 Parameter Estimation

Accurate estimation of the model parameters is essential for the reliability and applicability of any SRGM. In the proposed hybrid model, four key parameters *A*, *b*, *c*, and τ must be estimated. To estimate these parameters, two complementary statistical techniques are employed: Maximum Likelihood Estimation (MLE) and Nonlinear Least Squares (NLS).

MLE is a probabilistic method that estimates the parameters by maximizing the likelihood function, i.e., the probability of observing the given data given specific parameter values. This involves constructing a likelihood function based on observed failure times and finding the parameter set that maximizes this function. MLE is especially powerful for time-to-failure data as it yields efficient and asymptotically unbiased estimates.

Suppose we have observed software failure times, $t_1, t_2, \ldots, t_n$ and let the model be characterized by a parameter vector θ. Then the likelihood function is given below:

$$L(\theta) = \prod_{i=1}^{n} f(t_i; \theta)$$

where $f(t_i, \theta)$ is the probability density (or probability mass) function of the failure process at time t_i.

After taking logarithm in the above likelihood function, the log-likelihood function is presented below:

$$\ell(\theta) = \ln L(\theta = \sum_{i=1}^{n} \ln f(t_i, \theta)$$

The MLE estimates of the parameters are obtained by solving the following equations

$$\frac{\partial \ell(\theta)}{\partial \theta_j} = 0, \text{ for each parameter } \theta_j$$

Now we formulate the likelihood function for the NHPP-based SRGM.

For the interval domain data points (t_k, y_k); $i = 1, 2, \ldots n$, where t_k is the observation time and y_k is the cumulative number of observed failures by time t_k, the likelihood function under NHPP assumptions is given by:

$$L = \prod_{k=1}^{n} \frac{[m(t_k) - m(t_{k-1})]^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!} . e^{-[m(t_k) - m(t_{k-1})]}$$

The likelihood function for the time-domain dataset is given as:

$$L = \prod_{i=1}^{n} \lambda(t_i) e^{-\int_0^{t_n} \lambda(x) dx}$$

where $\lambda(t) = \frac{d}{dt} m(t)$ is intensity function.

In contrast, Nonlinear Least Squares (NLS) minimizes the sum of squared residuals between the observed failure data and the predicted values generated by the model. For each observation, the residual is computed as the difference between the actual and predicted cumulative number of failures. The total sum of these squared differences is minimized through iterative optimization techniques, providing an alternative route to accurate parameter estimation.

Suppose the observed cumulative number of failures up to time t_i is y_i, and the SRGM predicts the expected cumulative number of failures as $m(t_i)$.

The objective function to minimize is the sum of squared residuals (SSR):

$$S = \sum_{i=1}^{n} [y_i - m(t_i)]^2$$

A unique challenge in this hybrid model lies in determining the optimal change point (τ), which marks the transition from S-shaped to logarithmic growth behavior. An iterative search strategy is adopted, wherein multiple candidate values of τ are evaluated. For each candidate, the dataset is split into two segments: The S-shaped phase and the logarithmic phase. The model parameters (A, b, c) are estimated separately for each segment using either MLE or NLS, and the total error (residual sum or negative log-likelihood) is computed. The candidate τ that yields the best fit—i.e., minimizes residuals or maximizes the likelihood—is selected as the optimal change point.

3.3 Change Point Detection

In the context of hybrid software reliability models, detecting the change point (τ) is a critical task. The change point represents the moment during the testing process when the behavior of software failures shifts—typically due to modifications in testing strategy, resource allocation, or system updates. Accurate identification of τ ensures that the model transitions appropriately from the S-shaped growth to logarithmic growth, preserving the fidelity of predictions.

Two statistical techniques are primarily employed to detect the change point: the Likelihood Ratio Test (LRT) and the Cumulative Sum (CUSUM) technique. These methods are well-suited to identifying structural breaks or shifts in time series data, such as software failure logs.

The Likelihood Ratio Test compares two nested models: one with a fixed structure (no change point) and one that includes a change point at a specified location. By calculating the likelihood of both models and forming a ratio, the LRT evaluates whether introducing a change point significantly improves the model fit. This process is repeated across multiple candidate values of τ, and the value that maximizes the likelihood ratio is chosen as the most probable change point. This technique is grounded in strong statistical theory and is particularly effective when model assumptions about the distribution of failure data hold true.

The Cumulative Sum (CUSUM) technique, on the other hand, is a sequential analysis method used to monitor changes in the mean level of a process. In this context, CUSUM is applied to the sequence of residuals or failure counts to identify shifts in the trend. A cumulative sum of deviations from the expected value is plotted over time. A significant and sustained deviation from zero in the CUSUM plot signals a structural change, thereby helping to locate the change point τ. CUSUM is valued for its simplicity and sensitivity to small but sustained changes, making it particularly useful when data contains noise or irregularities.

Together, these methods offer complementary strengths—LRT provides a model-based formal hypothesis testing framework, while CUSUM offers intuitive, data-driven insights into behavioral changes in the failure process. Employing both ensures robust and dynamic detection of change points, thereby enhancing the adaptability and accuracy of the hybrid reliability model.

4. Experimental Validation

4.1 Data Sets

To empirically validate the proposed hybrid software reliability model, benchmark datasets were employed. These included publicly available datasets from NASA's software engineering laboratory and proprietary testing logs from industry partners. The selected datasets were chosen for their diversity, capturing a wide range of software testing environments and failure behaviors. Specifically, they reflect scenarios with delayed fault detection (indicating slow tester ramp-up), sudden increases in fault detection (due to intensified testing), and relatively stable periods where fault detection occurred at a near-constant rate. Such variability makes them ideal for testing the hybrid model's adaptability.

4.2 Evaluation Metrics

To evaluate the performance of the hybrid model, several standard assessment metrics were employed:

- Mean Squared Error (MSE): Measures the average squared difference between observed and predicted failure counts, indicating predictive accuracy.
  The formula of MSE along with explanation:

  $$\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}(F_i-\hat{F}_i)^2$$

  where n is total number of observations, F_i is observed failure count at the ith time point, $\hat{F}_i$ is predicted failure count at the ith time point.

- Akaike Information Criterion (AIC): Provides a relative measure of model fit by penalizing models with more parameters. Lower AIC indicates a better trade-off between fit and complexity.

  $$\text{AIC}=2k-2\ln(L)$$

  where k is the number of estimated parameters in the model and L is the maximum value of the likelihood function for the model

- Predictive Accuracy: Evaluated using a hold-out validation subset to assess the model's generalization capability to unseen data.
  It measures how well a model predicts **unseen data** using a hold-out validation set.

  Predictive Accuracy=Performance Metric on Validation Set (e.g., MSE, RMSE, MAE, R^2).

  This emphasizes that predictive accuracy isn't a single fixed formula, but rather depends on which performance metric is chosen and evaluated on the hold-out validation set.

4.3 Comparative Models

For comparative analysis, three classical SRGMs were used as benchmarks:

- Goel-Okumoto Model (NHPP): A foundational model assuming a NHPP for fault detection.
- Musa-Okumoto Logarithmic Model: Captures rapid fault detection early in the testing phase with a logarithmic decline over time.
- Ohba S-Shaped Model: Designed to reflect the gradual learning curve of testers, making it ideal for phased testing environments.

5. Results and Discussion

The hybrid model demonstrated superior performance across all datasets, particularly excelling in scenarios where the failure rate underwent significant changes during testing. In the NASA2 dataset, for example, the hybrid model achieved a lower MSE compared to the next best performing model. Furthermore, its AIC values were consistently lower, indicating a better balance between goodness-of-fit and model complexity. These results affirm the hybrid model's ability to adapt dynamically to changes in testing behavior and maintain high predictive accuracy.

The hybrid model's strength lies in its adaptability. By fusing delayed S-shaped and logarithmic behaviors, it captures both the learning phase of testers and the efficiency phase of debugging. The integration of dynamic change points enables timely model recalibration, making it suitable for agile and Develops environments.

However, the model's complexity introduces challenges in parameter estimation, especially when multiple change points exist. Further research may explore multi-phase hybrids or reinforcement learning-based change detection.

6. Applications and Implications

The practical value of any software reliability model lies in its ability to inform decision-making processes during the software development life cycle. The proposed hybrid model, by capturing both S-shaped and logarithmic failure behaviors and dynamically adjusting to change points, offers multiple real-world applications across various stages of software engineering**.**

Release Planning: One of the most critical applications is in software release planning. The model can assist organizations in determining the optimal release time by evaluating when the software has reached a satisfactory level of reliability. By forecasting the number of remaining defects and the expected failure rate, teams can make informed

decisions about whether to proceed with release or continue testing. This minimizes the risk of post-release failures and improves customer satisfaction.

Resource Allocation: The model's ability to detect change points also provides actionable insights into resource allocation. For example, a detected shift in failure trends might suggest the need for additional testers or changes in test strategy. These insights enable project managers to dynamically reallocate budgets, staff, or tools to maintain testing efficiency and effectiveness.

Tool Integration: The hybrid model is well-suited for integration into automated development and testing environments, such as continuous integration/continuous deployment (CI/CD) pipelines. By embedding the model into these systems, organizations can monitor reliability in real-time and receive automated alerts when reliability metrics fall below predefined thresholds. This facilitates proactive quality control and rapid response to emerging reliability issues.

In summary, the hybrid model is not only a theoretical advancement but also a practical tool with direct implications for enhancing software quality, optimizing workflows, and ensuring timely, reliable software releases.

7. Conclusion and Future Work

This study presents a statistically grounded hybrid SRGM that integrates the strengths of S-shaped and logarithmic growth behaviors, augmented by dynamic change point detection. Recognizing the limitations of conventional single-pattern SRGMs in capturing the complex, multi-phase nature of software failure processes, this model introduces a statistically rigorous framework capable of adapting to evolving testing conditions.

While effective, the model assumes a single change point. Future extensions could include multiple change points or continuous functions representing gradual shifts. Integration with machine learning for automated parameter tuning is also a promising direction.

References

[1.] Aggarwal, A., Kumar, S., & Gupta, R. (2024). "Testing coverage based NHPP software reliability growth modeling with testing effort and change-point". *International Journal of System Assurance Engineering and Management*, *15*(11), 5157-5166.

[2.] Aggarwal, A. G., Dhaka, V., & Nijhawan, N. (2017). Reliability analysis for multi-release open-source software systems with change point and exponentiated

Weibull fault reduction factor. *Life Cycle Reliability and Safety Engineering*, *6*(1), 3-14.

[3.] Ahmad, N., Khan, M. G., & Rafi, L. S. (2010). A study of testing-effort dependent inflection S-shaped software reliability growth models with imperfect debugging. *International Journal of Quality & Reliability Management*, *27*(1), 89-110.

[4.] Ahmad, N., Khan, M. G. M., & Rafi, L. S. (2010a). Software Reliability Modeling Incorporating Log-Logistic Testing-Effort with Imperfect Debugging. In *AIP Conference Proceedings* (Vol. 1298, No. 1, pp. 651-657). American Institute of Physics.

[5.] Ahmad, N., Khan, M. G., & Rafi, L. S. (2011). Analysis of an inflection S-shaped software reliability model considering log-logistic testing-effort and imperfect debugging. *International Journal of Computer Science and Network Security*, *11*(1), 161-171.

[6.] Bokhari, M. U., & Ahmad, N. (2006). Analysis of a software reliability growth models: the case of log-logistic test-effort function. In *Proceedings of the 17th IASTED international conference on modeling and simulation (MS'2006), Montreal, Canada* (pp. 540-545).

[7.] Boland, P. J., & Singh, H. (2003). A birth-process approach to Moranda's geometric software-reliability model. *IEEE Transactions on Reliability*, *52*(2), 168-174.

[8.] Camuffo, M., Maiocchi, M., & Morselli, M. (1990). Automatic software test generation. *Information and Software Technology*, *32*(5), 337-346.

[9.] Chang, I. P. (1997). An analysis of software reliability with change-point models. *Taipei, Taiwan: National Science Council*.

[10.] Chang, Y. P. (2001). Estimation of parameters for nonhomogeneous Poisson process: Software reliability with change-point model. *Communications in Statistics-Simulation and Computation*, *30*(3), 623-635.

[11.] Chatterjee, S., & Shukla, A. (2016). Modeling and analysis of software fault detection and correction process through Weibull-type fault reduction factor, change point and imperfect debugging. *Arabian Journal for Science and Engineering*, *41*(12), 5009-5025.

[12.] Huang, C. Y., & Hung, T. Y. (2010). Software reliability analysis and assessment using queueing models with multiple change-points. *Computers & Mathematics with Applications*, *60*(7), 2015-2030.

[13.] Lin, C. T., & Huang, C. Y. (2008). Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models. *Journal of Systems and Software*, *81*(6), 1025-1038.

[14.] Erto, P., Giorgio, M., & Lepore, A. (2018). The generalized inflection S-shaped software reliability growth model. *IEEE Transactions on Reliability*, *69*(1), 228-244.

[15.] Fang, C. C., & Hsu, C. C. (2025, January). Software testing strategies and release decisions with S-shaped growth models under different statistical confidence intervals. In *Third International Conference on Electrical, Electronics, and Information Engineering (EEIE 2024)* (Vol. 13512, pp. 213-220). SPIE.

[16.] Goel, A. L., & Okumoto, K. (2009). Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE transactions on Reliability*, *28*(3), 206-211.

[17.] Haque, M. A., & Ahmad, N. (2025). A logistic software reliability model with Loglog fault detection rate. *Iran Journal of Computer Science*, *8*(1), 1-7.

[18.] Hanagal, D. D., & Bhalerao, N. N. (2018). "Analysis of delayed S-shaped software reliability growth model with time dependent fault content rate function". *Journal of Data Science*, *16*(4), 857-878.

[19.] Huang, C. Y. (2005). Performance analysis of software reliability growth models with testing-effort and change-point. *Journal of Systems and Software*, *76*(2), 181-194.

[20.] Huang, C. Y., & Kuo, S. Y. (2002). Analysis of incorporating logistic testing-effort function into software reliability modeling. *IEEE Transactions on reliability*, *51*(3), 261-270.

[21.] Chin-Yu Huang, S. Y. K., & Lyu, M. R. (1999). Optimal Software Release Policy Based on Cost and Reliability with Testing Efficiency. *Proceedings: October 27-29, 1999, Phoenix, Arizona*, 468-473.

[22.] Huang, C. Y., Lyu, M. R., & Kuo, S. Y. (2003). A unified scheme of some non-homogenous poisson process models for software reliability estimation. *IEEE transactions on software engineering*, *29*(3), 261-269.

[23.] Huang, C. Y., & Lyu, M. R. (2011). Estimation and analysis of some generalized multiple change-point software reliability models. *IEEE Transactions on reliability*, *60*(2), 498-514.

[24.] Huang, C. Y., & Lin, C. T. (2006). Software reliability analysis by considering fault dependency and debugging time lag. *IEEE Transactions on reliability*, *55*(3), 436-450.

[25.] Huang, C. Y., & Lo, J. H. (2006). Optimal resource allocation for cost and reliability of modular software systems in the testing phase. *Journal of Systems and Software*, *79*(5), 653-664.

[26.] Hung-Cuong, N., & Quyet-Thang, H. (2024). An imperfect debugging non-homogeneous poisson process software reliability model based on a 3-parameter s-shaped function. *International Journal of Software Engineering and Knowledge Engineering*, *34*(06), 869-889.

[27.] Inoue, S., & Yamada, S. (2008). Optimal software release policy with change-point. In *2008 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 531-535). IEEE.

[28.] Inoue, S., & Yamada, S. (2011). Software reliability measurement with effect of change-point: Modeling and application. *International Journal of System Assurance Engineering and Management*, *2*(2), 155-162.

[29.] Iqbal, J., Quadri, S. M. K., & Ahmad, N. (2016). *Software Reliability Growth Models from the Perspective of Learning Effects and Change-Point* (Doctoral dissertation).

[30.] Jelinski, Z., & Moranda, P. (1972). Software reliability research. In *Statistical computer performance evaluation* (pp. 465-484). Academic Press.

[31.] Kapur, P. K., Singh, V. B., Anand, S., & Yadavalli, V. S. S. (2008). Software reliability growth model with change-point and effort control using a power function of the testing time. *International Journal of Production Research*, *46*(3), 771-787.

[32.] Ke, S. Z., & Huang, C. Y. (2020). Reliability prediction and management: A multiple change-point model approach. *Quality and Reliability Engineering International*, *36*(5), 1678-1707.

[33.] Kim, T., Ryu, D. and Baik, J. (*2024),* Enhancing Software Reliability Growth Modeling: A Comprehensive Analysis of Historical Datasets and Optimal Model Selections, *IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, Cambridge, United Kingdom, 2024, pp. 147-158

[34.] Khurshid, S., Shrivastava, A. K., & Iqbal, J. (2021). Effort based software reliability model with fault reduction factor, change point and imperfect debugging. *International Journal of Information Technology*, *13*(1), 331-340.

[35.] Kumar, A. (2021). Reliability analysis using change-point concept and optimal version-updating for open source software. *International Journal of Reliability and Safety*, *15*(4), 217-239.

[36.] Lee, C. H., Kim, Y. T., & Park, D. H. (2004). S-shaped software reliability growth models derived from stochastic differential equations. *IIE transactions*, *36*(12), 1193-1199.

[37.] Kimura, M., Yamada, S., & Osaki, S. (1992). Software reliability assessment for an exponential S-shaped reliability growth phenomenon. *Computers & Mathematics with Applications*, *24*(1-2), 71-78.

[38.] Musa, J. D. (1979). Software reliability measurement. *Journal of Systems and Software*, *1*, 223-241.

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

[40.] Pham, H. (2006). *System Software Reliability*. Springer.

[41.] Pradhan, S. K., Kumar, A., & Kumar, V. (2025). Multi release software reliability modelling incorporating fault generation in detection process and fault dependency with change point in correction process. *Scientific Reports*, *15*(1), 23145.

[42.] Samal, U., & Raj, M. (2025, March). Integrating Fault Removal Efficiency and S-Shaped Fault Detection in Software Reliability Growth Modeling. In *2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)* (Vol. 3, pp. 1-5). IEEE.

[43.] Sehgal, V. K., Kapur, R., Yadav, K., & Kumar, D. (2010). Software reliability growth models incorporating change point with imperfect fault removal and error generation. *International Journal of Modelling and Simulation*, *30*(2), 196-203.

[44.] Shrivastava, A. K., & Sharma, R. (2022). Developing a hybrid software reliability growth model. *International Journal of Quality & Reliability Management*, *39*(5), 1209-1225.

[45.] Shyur, H. J. (2003). A stochastic software reliability model with imperfect-debugging and change-point. *Journal of Systems and Software*, 66(2):135-141.

[46.] Singh, J. N., Yadav, A., Singh, O., & Anand, A. (2024). Environmental factor and change point based modeling for studying reliability of a software system. *International Journal of System Assurance Engineering and Management*, 1-11.

[47.] Verma, V., Anand, S., & Aggarwal, A. G. (2023). Optimal time for management review during testing process: an approach using S-curve two-dimensional software reliability growth model. *International Journal of Quality & Reliability Management*, *40*(9), 2278-2298.

[48.] Wang, J., & Zhang, C. (2022). Reliability model of open source software considering fault introduction with generalized inflection S-shaped distribution. *SN Applied Sciences*, *4*(9), 244.

[49.] Xiang Li, Min Xie, Szu Hui Ng (2010). Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points. *Applied Mathematical Modelling*, 34(11):3560-3570.

[50.] Yamada, S., Ohba, M., and Osaki, S. (1984). S-shaped software reliability growth models and their applications. *IEEE Transactions on Reliability*, 33(4), 289-292.

[51.] Zhao, M. (2003), Statistical reliability change-point estimation models, in: *Handbook of Reliability Engineering*, pp. 157163.

[52.] Zhao, J. Liu, H.W. Cui, G. Xiao-Zong Yang, X.Z. (2006). Software reliability growth model with change-point and environmental function, *Journal of Systems and Software*, 79(11):1578-1587.