Software reliability prediction using NHPP and Least median of squares (LMS) for parameters estimation

Mayuri H. Molawade [1*], Dr. Shashank D. Joshi[2], Dr. Rohini Jadhav[3]

[1] Department of computer Engineering BharatiVidyapeeth(Deemed to beUniversity) College of Engineering,PuneResearch Scholar, India

[2] Department of computer Engineering BharatiVidyapeeth (Deemed to be University) College of Engineering, Pune Professor, India

[3]Department of Information TechnologyBharatiVidyapeeth(Deemed to beUniversity) College of Engineering, Pune Associate Professor, India

Article Info
Page Number: 1194-1206
Publication Issue:
Vol. 71 No. 3 (2022)

Article History
Article Received: 12 January 2022
Revised: 25 Febuary 2022
Accepted: 20 April 2022
Publication: 09 June 2022

Abstract— Based on Cauchy distributions, this research provides an Innovative Assessment Model for Software Reliability Prediction. In this study, the finite-fault NHPP reliability model is used to analyse the performance of software reliability using Cauchy distributions. The Least Median of Squares (LMS) approach was used to estimate the parameters, and Newton's method was used to solve the nonlinear equations.The Cauchy distributions model therefore performed well. Because the mean square error (MSE) and failure incidence rate both decrease over time, it is important to look at the intensity function. When the software reliability was examined after setting the mission time down the road, the Cauchy distributions model used to have a larger durability trend than another models, which indicates a decline in reliability with mission time. As a result, the Cauchy distributions model outperforms the other exiting models and thus, the improved model used to software developers in order to improve programmereliability.

Keywords— Least Median of Squares,Mean square error, Cauchy distributions, Mission time.

1. INTRODUCTION

Software engineering's main goal is to produce slightly elevated software at a minimal cost. As the scope and complexity of software developed, management issues became more prevalent. Dependability is one of the distinguishing stages of the software production process. Software reliability is the likelihood that a certain operation will occur over a specific period of time. Software Reliability [1] is a crucial component of software quality, together with performance, accessibility, consistency, serviceability, capability, install ability, ease of maintenance, and documentation.The acceptance or failure of a software product is determined by its reliability. Developers have raised the demand to improve software reliability for business continuity due to high development costs and increasing economic competition [2]. Practitioners are also working to measure, track, and predict software reliability. High-quality software is most noticeable for its reliability, which is also the least observable mechanism in relation to customer happiness. Additionally, dependability prediction can consistently give practitioners excellent outcomes [3]. It also assesses the functionality of software to ensure that it meets operational requirements. In the other words, it ensures that the programme can fix itself if it malfunctions and will help with user acceptance of the product. Software development failures are unavoidable because software applications are complicated information products [4]. Several actions should be taken during the development phase to find flaws and problems as soon as possible.Furthermore, the management process includes several measures such as finding errors, categorizing errors based on seriousness, and resolving issues [5]. Developers may utilise reliability prediction methodologies to measure and reduce faults in addition to the error agents and severity organizational activities to boost the dependability of the product. The complexity of software design, poor quality control, the market, capabilities profitable targets, and design engineering estimation are the main causes of software failure. A software programme becomes more complex as more lines are added for various functions. Another key factor in programme failure is ignorance. Due to the intrinsic complexity of the software's code and architecture, it is challenging to produce error-free, perfect software. [6]. A software reliability growth model is intended to build suggestions by forecasting failure in advance and learning from previous failures in order to create quality software that is also reliable [7]. If the evaluated growth is reduced as a result of planned progress, the designers would have enough time to refine new projects, including new proposals or resources to address the identified difficulties. The use of an adequate reliability model that establishes the variance of reliability involving time with correct application to the tech sector is required for measuring, predicting, and estimating software dependability [8].To provide a measurement for software reliability, software reliability engineering creates a model of a software system based on failure data. Over the last three decades, several software reliability models have been established. Non-Homogeneous Poisson Process (NHPP) models have been successfully used in studying hardware reliability problems as a general class of well-defined stochastic process models in reliability engineering. They are particularly useful for describing failure processes that exhibit specific patterns, such as reliability growth and deterioration. As a result, applying NHPP models to software reliability analysis is easy.A software reliability model based on the non-homogeneous Poisson process (NHPP) that predicts the failure severity function and the mean value function based on reliability attribute factors such as the number of remaining failures and failure characteristics [9]. As a result, a novel approach for predicting the failed functions in technical grounds on dependability attribute elements must be developed. Software technology, the core of the 4th industrial revolution, is rapidly convergent and being used in a wide range of businesses. As a result, there is an increasing requirement of high software that can analyze massive amounts of big data without failing. Software developers are dedicating extensive amounts to future research in order to quality of the selected reliability in order to address this issue. As a result, in order to increase software quality, system reliability models based on the non-homogeneous Poisson process (NHPP) have been extensively investigated.

The dependability of technology is an essential aspect of software success. Software reliability prediction is the potential of a computer programme operating faultlessly in a given environment for a given period of time.As a result, the software production and testing industry considers software dependability research to be beneficial. One of the most challenging tasks in software maintenance is predicting programme quality.To detect high-risk modules for future projects, a machine learning prediction model is constructed utilizing methodologies and defective data from existing projects, permitting testing efforts to be directed toward those specific 'risky' modules.

2. LITERATURE SURVEY:

Anjum et al [15] explains how software problems and cyber-security flaws offer significant dangers, making software testing a crucial duty for enterprises. In order to take into consideration the subjectivity and ambiguity of decision-makers, it prioritised software vulnerabilities using a hybrid Fuzzy best-worst approach (FBWM).

The Type-2 Gumbel and Burr-XII lifespan distributions, which were followed by a diminishing contour over time when the risk function first increased in software product testing, were employed in Kim et aldependability .'s presentation of the software reliability model [16]. Both the maximum likelihood estimation technique and the parametric estimation strategy were used to study the characteristics of the software reliability model. This parametric estimating technique, even though it's unsuccessful.

Kim et al. [17] found that the Goel-Okumoto model is smaller than the modified Lindley distribution model. Therefore, the current Goel-Okumoto model could be viewed as a successful model. The modified Lindley distribution model outperforms the Goel-Okumoto model, however the dependability function shows a non-incremental trend as mission time grows. The mean square error, average value, and hazard function trends can be used by software operators to learn more about the many forms of software dependability failures and how they relate to different lifetimes.

Pham et al [18] presented a new distribution function that uses a Vtub-shaped failure rate function to characterize.

The reliability model's design challenges and the various factors influencing software reliability were both highlighted by Kishan et al in their study [19].

The performance of a finite failure's reliability Yang et al. [20] have investigated the Weibull lifespan distribution-based NHPP software reliability model.

Tae-Hyunyoo[21]

The Laplace trend test was used to determine whether the data were accurate. This study's suggested monotonic intensity function outperforms its competitors in terms of consistency. Therefore, a replacement model can be created using the monotonic intensity function.

Huang, C-Y [22]

In Non-Homogeneous Poisson Process, a method for developing a software reliability growth model is presented. The major goal is to present a framework for modelling software reliability that takes testing effort and change point into account.

XuemeiZhangHoangPham [23]All parameters were analyzed and ranked in terms of their impact on software dependability using two techniques: the relative weight method and analysis of variance methodology (ANOVA). The findings of the study have significant implications for future research and software development practise.

Herbert hecht,Myron Hecht [24] The fundamental failure model that provides a unified approach to software and hardware reliability is described, as well as the consequences of that model for traditional software reliability approaches.

Daniel R jeske,Hoangpham[25]the observation time for reported software failures stretches to infinity, the maximum likelihood (ML) estimates of the parameters of a well-known software reliability model are inconsistent.

LianfenQian[26]The (single-phase) Rayleigh model has been found to fit software defect arrival patterns among dynamic models.

Tae-Jin, Yang[27] The Weibull life distribution is applied to the finite-fault NHPP reliability model to examine the performance of reliability.

XiangLi[28] A mean-variance-skewness model is offered as an extension of the fuzzy mean-variance model, and the relevant variations are also explored. A genetic algorithm using fuzzy simulation is created to solve the presented models.

A.Yadav&R.A.Khan[29] Models of reliability based on various dimensions. The models under consideration can have an infinite or finite number of failures. There are finite failures in all exponential distribution models.

K. VenkataSubba Reddy [30] Product engineers may forecast when to finish testing, release the software, and avoid overtesting in order to reduce development costs according to the software reliability growth model, which offers accurate parameter estimation early in the testing and debugging phase.

All phases of the NHPP software dependability model based on the Cauchy distribution will benefit from the suggested methods. The method begins by discussing a non-homogenous Poisson process technique that completely lowers the failure rate. Finally, it created a system that, by combining all of the earlier technical steps, can quickly determine the software reliability. The methods previously described are those that are now employed in NHPP. The next paragraphs go over the methods and advantages of the algorithms in the proposed method.

3. Motivation

Software technology, the cornerstone of the fourth industrial revolution, is swiftly integrating and being used in a wide range of sectors. As a result, there is an increasing need for highcaliber software that can successfully process massive amounts of big data information. Software developers are investing large resources in research and development to overcome this problem and increase the dependability of their programmes. Software reliability models given the non-Poisson process (NHPP) have so been thoroughly studied in an effort to improve software quality.

4.WHAT WILL BE THE INPUT PARAMETERS OF SOFTWARE FOR RELIABILITY PREDICTION

The values of the parameters in this reliability model are used to forecast the software's reliability. The model cannot be used to forecast software dependability since the beginning failure intensity, the total number of mistakes, and their values are unknown.

If the values of these variables were broadly common to all types of software and could be easily identified based on any software attribute, that would be very helpful.

However, there isn't yet a reliable, simple way accessible.

Currently, the methods used by both software reliability models involve estimating the values of these characteristics for each piece of software that is being modelled using reliable information about that programme. To put it another way, the value of these parameters is determined using the precise outcomes of the programmed being model.

Because of this, throughout system testing, the software system's performance is carefully monitored, and reliability data is analysed throughout testing and recorded up to timeT.

Statistical methods are then used on the gathered data to determine the value of these parameters. The software's reliability (measured in terms of reliability intensity) can be evaluated if the values of the parameters are known. A substantial amount of data is required for these mathematical procedures; otherwise, the values of the variables remain unknown.

- Software input parameters
- Find reliability
- Find reliabilityparameters

4.1 What will be output measures?

Accuracy

The accuracy in a binary class issue is the proportion of accurately classified instances to all instances. It evaluates the hit-or-miss effectiveness of the classifier.

Accuracy = TP+TN÷TP+FP+TN+FN

4.2 Precision

It gauges the accuracy of the anticipated outcomes and measures the miss categorization rate. It is determined by dividing the total number of identified instances for a particular class by the number of detected instances.

Precision = TP ÷TP+FP

4.3 Recall

For a given class, the recall is the proportion of correctly identified examples to all instances that truly belong to the class. The number of times the classifier has hit the class is recorded. Additionally known as "detection chance," it is quantified as follows:

Recall= TP÷TP+FN

On the other hand, it's critical to take into account the compromise between recall and precision. For instance, the recall will be minimal if the desired class comprises numerous modules but the precision will be 1 if the model correctly predicts just one module from the intended class. In contrast, a high recall but a low accuracy is the outcome if the model correctly predicts all defective parts while still making some incorrect predictions about some non-defective modules. The F-measure, which effectively combines precision and recall into a single useful statistic, is a required method for combined computation.

4.4 F-measure

The F-measure for a classifier combines the accuracy and recall values for a specific class (faulty or not-faulty). It determines the measures' harmonic means, recall, and precision, all of which are equally important.

F1-Measure = 2×Precision×Recall ÷Precision+ Recall.

5. Proposed Method

Technology dependability is a critical component of software success. Software reliability prediction refers to the potential of a software project's fault-free functioning in a specific environment over a given time period.As a result, software reliability research is considered useful in the software development and testing industry. One of the most difficult tasks in software creation and maintenance is predicting software quality. A machine learning prediction model is built using software metrics and defective data from previous projects to detect high-risk modules for future projects, allowing testing efforts to be directed toward those particular 'risky' modules. This paper proposes a NovelEvaluation Model for Software Reliability Prediction based on Cauchy distributions.The performance of software reliability is investigated in this work by implementing the Cauchy distributions to the finite-fault NHPP reliability model. The parameters were estimated using the Least Median of Squares (LMS) method, and nonlinear equations were solved using Newton's method. As a conclusion, the Cauchy distributions theory were helpful in evaluating the strength functions because the failure incidence rates reduces drastically as the failure duration grows, as does the mean square error (MSE).When evaluating software reliability after establishing a future mission time, the Cauchy distributions model showed a stronger durability tendency than the other models, indicating a reduction in reliability with mission duration. As a consequence, the Cauchy distributions model beats other models, making it a better model for software developers to utilize in order to increase programme reliability.

5.1 Block Diagram:

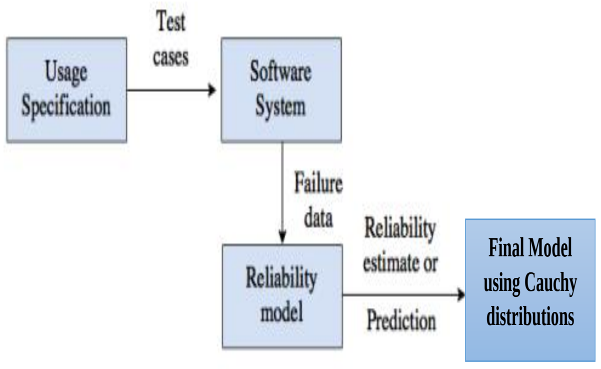

Figure 1.proposed system

1. Finite Failure NHPP: Software Reliability Model

The non-homogeneous Poisson process (NHPP) software reliability model is a model that measures reliability that use the mean rate of failure function generated per unit time.

$$P\{N(t) = n\} = \frac{[m(t)]^n . e^{-m(t)}}{n!}$$

Mean value function $m(t) = \int_0^t \lambda(s)ds$

$\frac{dm(t)}{d(t)} = \lambda(t)$- intensity function of the NHPP model.

We will examine the system durability using finite failurecases in this study. That instance, if indeed the estimation of the failure that can be found up to time [0, t] in the finite-failure NHPP model is, therefore the mean valuation and the intensity function are
function are asfollows.

m $(t|\theta$, b) = $\theta F(t)$

$\lambda(t|\theta$, b) = $\theta F(t)' = \theta f(t)$

the likelihood function of the finite-failure NHPP model isderived as follows.

$$L_{NHPP}(\ominus |x) = (\prod_{i=1}^{n} \lambda(x_i)) \exp[-m(x_n)]$$

5.2 Proposed software reliability model algorithm

1. Verifying the data on software failures gathered from trend test analysis

2. Using the LMS approach to calculate the coefficients for the suggested model

3. Examining the suggested model's performance factors and reliability

Expression of Reliability model is:

a = initial fault-content of the software

k = A constant parameter

b1 = bug prediction rate/detection rate per unit time.

M (t) = expected number of reliability predicted,

$$\mathrm{m1(t) = a/(1 + k.\,e^{-b1.t})}$$

Inheritance Tree Depth, the maximum distance in the superclass from the base to a particular class. DIT is referred to as the root class's longest possible inheritance path.

$$\mathrm{m2(t) = a/(1 + k.\,e^{-b1.dit})}$$

The weighted methods per class, or WMC, is the total complexity of the class's methods. When all methods have a complexity of one, the number of methods is equal. the total normalised complexity of all of a class's methods.

$$\mathrm{m3(t) = a/(1 + k.\,e^{-b1.wmc})}$$

Connection between Objects The number of classes connected to a particular class is represented by the CBO metric. Method calls, field accesses, inheritance, method parameters, return types, and exceptions can all lead to these couplings.

$$\mathrm{m4(t) = a/(1 + k.\,e^{-b1.cbo})}$$

The Line of Code (LOC) measure, which is based on Java binary code, sums the amount of fields, methods, and instructions in each method of the class under investigation.

$$\mathrm{m5(t) = a/(1 + k.\,e^{-b1.loc})}$$

At the conclusion, the average reliability will be used to verify for product reliability after taking into account each and every attribute.

$$\mathrm{R(\delta|Xm) = \int_{t=o}^{t} x(m(dit) + m(wmc) + m(cbo) + m(loc))/y}$$

5.3SOFTWARE FAILURE TIME ANALYSIS OF RELIABILITY ATTRIBUTES

The software reliability model's reliability structures were examined using software failure time data in this part. Furthermore, a trend test should be performed to ensure data reliability. The Laplace trend test was used to analyze trends in this investigation. Figure 5 illustrates the Laplace trend test findings, which show that the Laplace factor approximations were scattered around 0 and 1, suggesting that high values are uncommon. As a result, recommending a dependability system based

n the this data is appropriate.

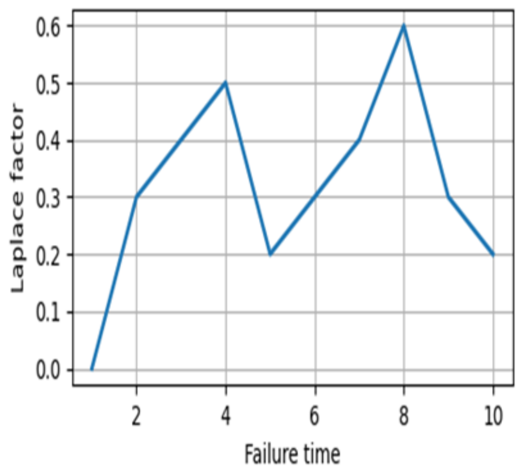

Figure2:Trend test of Laplace factor

5.3 Analysis of data using Cauchy distribution:

Before beginning any type of statistical analysis, it is highly recommended that you carefully examine the data. By plotting the data as a function of time. The failure, times are plotted against the cumulative number of observed failures are shown in Fig3.

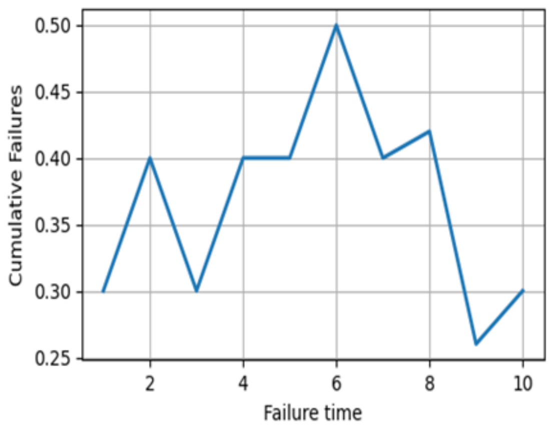

Figure3:Failure time Vs Cumulative failures

5.4 Model of reliability:

Since it is anticipated that the software flaw will be fixed with each failure, the total number of failures in infinite amount of time is bounded. The total number of errors in a piece of software, whose reliability is being modelled, is finite, hence the number of failures is finite. The failure intensity lowers as a result of the nature of the software development life cycle, particularly system testing, the activity wherein reliability modelling is implemented.

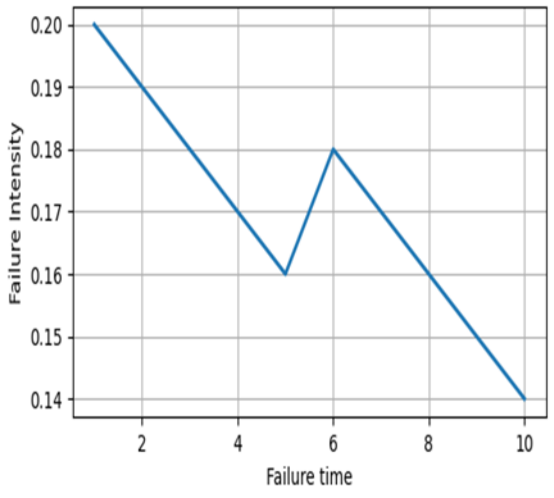

Figure 4: Plot for the model of Reliability

We propose a flexible SRGM based on the NHPP model that includes a discrete inflection S-shaped model. The proposed SRGM's performance is compared to that of other discrete logistic and Gompertz curve models utilising various criteria.

TABLE 1.

Predicted Relative Errors

Rating of Testing Progress(%)	**Discrete exponential SRGM**	**Discrete inflection S-shaped SRGM**	**Discrete logistic curve Model**	**Discrete Gompertz curve model**	**Proposed**
10	1	1	0.5	0.4	0.5
20	0	0.2	0.4	0.4	0.6
30	0.2	0.1	0.4	0.3	0.5
40	0.2	0.2	0.3	0.3	0.4
50	0.2	0.2	0.3	0.3	0.4
60	0.2	0.2	0.3	0.2	0.3
70	0.1	0.1	0.2	0.2	0.3
80	0.1	0.1	0	0.2	0.3
90	0	0	0	0	0.2
100	0	0	0	0	0.1

The acquired results show that the proposed model has a better fit and is more applicable to a variety of specific applications. We conclude that the suggested SRGM technique outperforms other SRGMs and provides a good predictive capability for failure data.

6. CONCLUSION

Software reliability models are design to measure best quality of software as well as it will be helpful for measuring reliability. In this model we calculating parametric reliability using depth of inheritance tree, weighted methods per classCoupling between objects,line of code after calculating each reliability. We are going to calculate average of all parameter and predict reliability. Trend test of Laplace factorFailure time vs cumulative failures, plot for the model of reliability, predicted relative errors by using these graphs and table we are stating that proposed model 0.1% of errors.

REFERENCE

1. Sahu, Kavita, Fahad A. Alzahrani, R. K. Srivastava, and Rajeev Kumar. "Evaluating the Impact of Prediction Techniques: Software Reliability Perspective." CMC-COMPUTERS MATERIALS & CONTINUA 67, no. 2 (2021): 1471-1488.
2. Kaswan KS, Choudhary S, Sharma K., Software Reliability Modeling using Soft Computing Techniques: Critical Review. J Inform Tech SoftwEng 5:144. 2015.
3. C.Y. Huang, M.R. Lyu, Estimation and Analysis of Some Generalized Multiple Change-Point Software Reliability Models, IEEE Transaction on Reliability, Vol. 60, no. 2, pp.498-514, 2011.
4. J. E. Gaffney and J. Pietrolewiez. An Automated Model for Software Early Error Prediction. In Proc. of 13th Minnow Brook Workshop on Software Reliability, Blue Mountain Lake, NY,45-57, 1990.
5. K. S. Trivedi, M. Malhotra and R. M. Fricks. Markov Reward Approach toPerformabilityand Reliability Analysis.In Proc. of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE,7-11, 1994.
6. S. S. Gokhale. Architecture-Based Software Reliability Analysis: Overview and Limitations. IEEE Transactions on Dependable and Secure Computing, 4(1), 32-40, 2007.
7. K.Sahu and R. K. Srivastava. Revisiting Software Reliability, Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing, Springer, 221-235, 2019.
8. K.Sahu and R. K. Srivastava. Soft Computing Approach for Prediction of Software Reliability. ICIC Express Letters,12(12),1213-1222, 2018.
9. I. H. Chang,K. Y. Song, H. Pham, A Software Reliability Model with a Weibull Fault Detection Rate Function Subject to Operating Environments. Applied Science.2017; 7(983): 1-16.

10. P. Nistala, K. V. Nori and R. Reddy, Software Quality Models: A Systematic Mapping Study, IEEE/ACM International Conference on Software and System Processes (ICSSP), pp. 125-134, 2019.
11. D. K. Singh and A. K. Bharti, A Comparative Studies Of Software Quality Model For The Software Product Evaluation, International Journal of Research in Engineering & Technology, Vol. 6, Issue 8, pp. 1-18,2018.
12. M. P. Cristescu, J. A. Vasilev, M. V. Stoyanova, and A.M. R. Stancu, Capability And Maturity.Characteristics Used In Software Reliability Engineering Modeling, Land Forces Academy Review,Vol. XXIV, no 4(96), pp. 332-341, 2019.
13. M. Gheisari et al., An Optimization Model for Software Quality Prediction With Case Study Analysis Using MATLAB, IEEE Access, Vol. 7, pp.85123-85138, 2019.
14. D. Tomar, and P. Tomar, New Component-Based Reliability Model (CBRM) to Predict the Reliability of Component-Based Software, International Journal of Reliability and Safety Publication, Inder Science, Vol. 13, no.1, pp. 83-95, 2019, Print ISSN: 1479-389X Online ISSN: 1479-3903.
15. Anjum, Misbah&Kapur, P.K. &Agarwal, Vernika&Khatri, Sunil Kumar. (2020). A Framework for Prioritizing Software Vulnerabilities Using Fuzzy Best-Worst Method. 311-316. 10.1109/ICRITO48877.2020.9197854.
16. Hee-Cheul. Kim, "The Comparison Analysis about Reliability Features of Software Reliability Model Using Burr-XII and Type-2 Gumbel Lifetime Distribution." International Journal of Engineering Research and Technology 12, no. 1 (2019): 73-78.
17. Molawade, Mayuri. (2019). Software reliability prediction using Knowledge Engineering approach. International Journal of Advanced Trends in Computer Science and Engineering. 8. 2768-2772.10.30534/ijatcse/2019/14862019.
18. Kim, Hee-Cheul. "A Comparative Study on the Finite Failure Software Reliability Model with Modified Lindley Type Lifetime Distribution." International Journal of Engineering Research and Technology 12, no. 6 (2019): 760-764.
19. H. Pham, “Distribution Function and its Application in Software Reliability,” International Journal of Performability Engineering, Vol. 15, No. 5, pp. 1306-1313. 2019.
20. S. Yadav and B. Kishan, “Assessments of Computational Intelligence Techniques for Predicting Reliability of Component-Based Software Parameter and Design Issues,” International Journal of Advanced Research in Engineering and Technology, Vol. 11, No. 6. pp. 565-584, 2020.
21. Tae-Jin Yang, “A Study on the Reliability Performance Analysis of Finite Failure NHPP Software Reliability Model Based on Weibull Life Distribution,” International Journal of Engineering Research and Technology. Vol. 12, No. 11, pp.1890-1896. 2019.
22. Tae-Hyunyoo,”The Infinite NHPP Software Reliability model based on monotonic function”,Indian Journal of Science and Technology,Vol 8(14), DOI: 10.17485/ijst/2015/v8i14/68342, July 2015.
23. Huang C-Y,” Performance analysis of software reliability growth models with testing-effort and change-point”. J Syst Software. 2005; 76:181–94.
24. XuemeiZhangHoangPham,”An analysis of factors affecting software reliability”. Journal of Systems and Software

25. .Herbert Hecht,Myron Hecht," Software reliability in the system context". IEEE Transactions on Software Engineering ; (Volume: SE-12, Issue: 1, Jan. 1986)
26. Daniel R jeske,Hoangpham,"On the Maximum Likelihood Estimates for the Goel–Okumoto Software Reliability Model", The American Statistician ;Volume 55,2001-Issue 3.
27. LianfenQian,"Dynamic Two phase truncated Rayleigh model for release date prediction of software".Journal software Engineering &Applications;2010,3,603-609.
28. Tae-Jin, Yang," A Study on the Reliability Performance Analysis of Finite Failure NHPP Software Reliability Model Based on Weibull Life Distribution", International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 11 (2019), pp. 1890-1896.
29. .XiangLi," Mean-variance-skewness model for portfolio selection with fuzzy returns". European Journal of Operational Research;Volume 202, Issue 1, 1 April 2010.
30. .A.Yadav&R.A.Khan," Critical Review on Software Reliability Models". International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009.
31. K.VenkataSubba Reddy 1 and Dr.B.Raveendra Babu2," logistic regression approach to software reliability engineering with failure prediction". International Journal of Software Engineering & Applications (IJSEA), Vol.4, No.1, January 2013.

32. M. H. Molawade, S. D. Joshi and R. Jadhav, "Software reliability prediction using data abstraction and Random forest Algorithm," 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2021, pp. 1-5, doi: 10.1109/UPCON52273.2021.9667