ISSN: 0974-5823 Vol. 7 No. 5 May, 2022

International Journal of Mechanical Engineering

Comparison between software reliability models with novel reliability model using Cauchy distribution

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

[1]Department of computer Engineering Bharati Vidyapeeth (Deemed to be University) College of Engineering, India, Pune Research Scholar

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

[3]Department of information technology Bharati Vidyapeeth (Deemed to be University) College of Engineering, India, Pune Associate Professor

Abstract— In this paper the performance of software reliability is evaluated in this work by applying the Cauchy distributions to the finite-fault NHPP reliability model. To validate dependability performance, the Cauchy distribution models were compared to the Software basic model. This was accomplished using failures during software according to time, the maximum likelihood estimation method (MLE) is used to calculate parameters and using two congruent parts nonlinear equations was defined. The parameters were estimated using the Least Median of Squares (LMS) method, and nonlinear equations were solved using Newton's method. As a result, the Cauchy distributions model was effective in analyzing the intensity function since the failure occurrence rate decreases dramatically as the failure time increases, and the median square error (MSE) is also reduced.

Keywords— Finite-fault , Maximum likelihood estimation, least median of squares, Median square error, Cauchy distributions.

I. INTRODUCTION

When there is development of any software we need to consider low cost but have maintained reliability also. Management problems started to dominate as the scale and sophistication of software grew. One of the defining characteristics of the software development process is dependability. Software reliability is known as the probability of expected operation over a specified time interval. Software Reliability [1], along with functionality, accessibility, consistency, serviceability, capability, install ability, maintainability, and documentation, is an essential aspect of software quality. 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]. Furthermore, practitioners are attempting to quantify, monitor, and forecast software reliability. Reliability is the most notable feature of high-quality software, and predicting reliability is the most observable mechanism related to customer satisfaction. Furthermore, reliability prediction is capable of providing practitioners with perfect results every time [3]. It also assesses the functionality of software to ensure that it meets operational requirements. In other words, it will aid in the acceptance of the software product and ensures the software's ability to repair itself if it fails. Since software applications are complex information products, certain errors are inevitable during the software development phase [4]. The development phase should include several steps to detect errors and faults as soon as possible. Furthermore, the management process includes several measures such as finding errors, categorising errors based on seriousness, and resolving issues [5]. Along with the error criticality and severity management processes, developers may use reliability prediction approaches to measure and minimise faults in order to increase the software's reliability. The primary causes of software failure are the complexities of software design, inadequate quality control, the marketplace, capability profitable targets, and engineering design estimation. The number of lines in a software programme for a variety of features increases software

complexity. Lack of knowledge is also a major cause of programme failure. Producing error-free and 100% quality software is difficult due to the inherent complexity of the code and design of the 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. Measuring and predicting software reliability and its estimation necessitates the use of an appropriate reliability model that determines the difference of reliability including time with its proper application to the software industry [8]. The novel method has been developed to predict the failure function in software based on the reliability attribute factors. The main contribution of this work is to evaluate the performance of software reliability and to improve the software dependability models based on Non homogenous Poisson process (NHPP).

II. SOFTWARE RELIABILITY MODELS

The fundamental field of the fourth industrial revolution, software technology, is quickly converging and being implemented in a variety of industries. As a result, the demand for high-quality software that can process enormous amounts of big data information without failure is growing. To address this issue, software developers are devoting significant resources to research and development in order to improve program reliability. As a result, software dependability models based on the non-homogeneous Poisson process (NHPP) have been intensively researched to improve software quality

The dependability of technology is an essential aspect of software success. The possibility of a computer program's failure-free operation in a specified environment for a specified time span is referred to as software reliability prediction. 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. A Novel Evaluation Model for Software Reliability Prediction based on Cauchy distributions. As a consequence, the Cauchy distributions model was efficient in the analysis of the intensity function since the failure occurrence rate falls greatly as the failure time passed and the mean square error (MSE) is also minimal. When the software reliability is evaluated after setting the mission time in the future, the Cauchy distributions model exhibited a greater reliability trend than the other models, which indicates a drop in reliability with mission time. As a result, the Cauchy distributions model outperforms the other exiting models and thus, the improved model is used by software developers to improve program reliability.

BLOCK DIAGRAM**:**

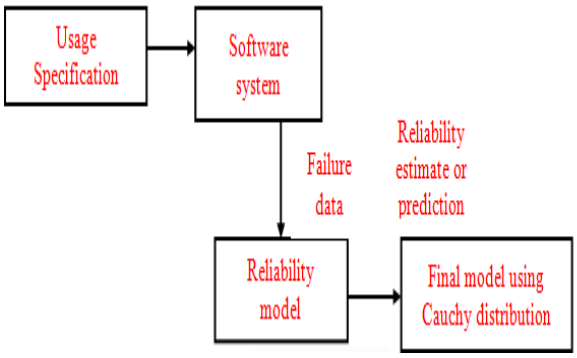

Figure 1: Block diagram

As a result, the Cauchy distributions model outperforms the other exiting models and thus, the improved model used to software developers to improve program reliability.

The pseudo code of software reliability model follows below,

A. Method Discussion using algorithm

Algorithm 1: Software Reliability Algorithm

1) Collect Failure information.
2) Choose an appropriate model.
3) Perform a fitted model.

a) By inserting the estimated values of the parameters in the chosen model, the fitted model is created. We now have a fitted model based on the available failure data and the model form we selected.

4) Parameters should be estimated.
5) Derive Performance measure estimation.
6) Software Reliability

Steps involved in the algorithm is given below,

STEP 1:For the most part, such data should be in the form of failure counts or times between failures. The first stage in creating a model is to thoroughly examine the data to get insight into the nature of the process being modeled.

STEP 2:Select a suitable model based on your knowledge of the testing procedure and assumptions.

STEP 3:By inserting the estimated values of the parameters in the chosen model, the fitted model is created.

STEP 4: Depending on the type of valuable data, different procedures are usually followed.

STEP 5: Finally, software reliability is obtained. This can be used for software development planning, scheduling, and assumptions.

The techniques of the software reliability architecture (Figure1) are explained in depth in the following section, which goes through each of the strategies used in the proposed system to overcome the constraints in a competent manner.

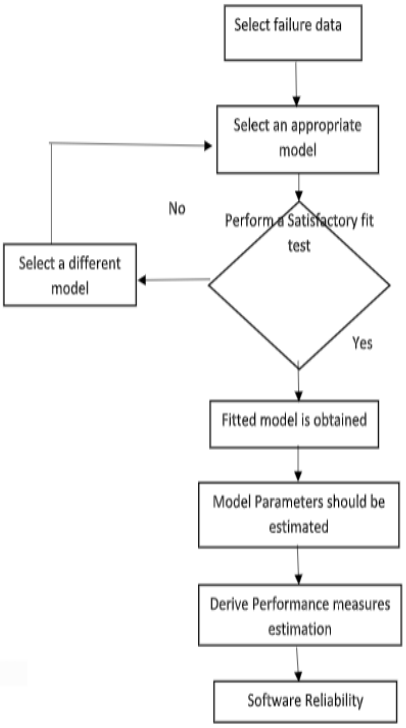

Figure 2: Flow chart for Software Reliability Modelling

This process is being carried out by the following Models which is been explained in the next section.

III. NHPP MODEL:

Models of software reliability that assume software failures behave like a Non-Homogeneous Poisson process (NHPP).The stochastic process's parameter failure intensity of software at time t is denoted by Y(t), which is time-dependent.

Let,

F(t) be the total number of faults discovered at time t.

n(t) be the expected number of faults.

Then, $n(t) = E(F(T)$ and $Y(t)$ be the failure intensity function is related as follows,

$$n(t) = \int_0^t \quad Y(S)ds \text{---------- (1)} \qquad \frac{dn(t)}{dt} = Y(t) \text{---------- (2)}$$

F(t) was previously known to have a Poisson probability density function (PDF) with parameter n(t), i.e.

$$p(F(t) = q) = \frac{[n(t)]^q}{q!} e^{-n(t)}$$

$q = 0,1, \ldots \ldots \infty$ --------- (3)

The probability of failure can be used to explain these time-domain models of the NHPP method. The failure intensity function (failure occurrence rates per fault) Y(t) will be written differently in this model, as will the mean value of the function n(t).

To add to this, the Record Value Statistics (RVS) model can be utilized with the NHPP model, and the mean value function was:

$$n(t) = -ln(1 - H(t))\text{--------- (4)}$$

$$H(t) \rightarrow \ cumulative\ distribution\ function$$
$$f(t) \rightarrow Probability\ distribution\ function$$

Equation (4) becomes,

$$Y(t) = n'(t) = \frac{f(t)}{(1-H(t))} = k(t) \quad \text{---------- (5)}$$
$$k(t) \rightarrow Hazard\ function$$

Given finite failure NHPP models, let θ be the predicted number of faults that would be discovered. The mean value function of the finite failure NHPP models is then calculated as follows,

$n(t) = \theta H(t)$ ---------- (6)

Equation (6) can be rewritten as follows,

$$Y(t) = [\theta - n(t)]\frac{H\prime(t)}{1-H(t)}[\theta - (n(t)]k(t) \quad \text{------- (7)}$$

$$k(t) \rightarrow Probability\ of\ failure$$
$$\theta - \big(n(t)\big) \rightarrow Expected\ number\ of\ failures$$

The sequence of times between successive software failures is denoted by

{ t_m,m = 1, 2,... }The time between (m– 1)st and nth failure is denoted by t_m. Assume that $x_m$represents failure time m, and thus

$$x_{m=\sum_{j=1}^{m}} \quad t_j \text{---------- (8)}$$

NHPP Model for Software Reliability with Monotonic Intensity Functions is explained below.

As a result, software dependability models based on the non-homogeneous Poisson process (NHPP) have been intensively researched to improve software quality.

A. POWER-LAW PROCESS

The Power Law Process (PLP) is a well-known large NHPP model that is used to depict the dependability of repairable frameworks by taking into account the analysis of observed failure data. This model was drawn from the equipment unshakable quality zone. A substantial amount of information about the PLP model is available when the traditional perception is taken into account.

ALGORITHM 2: POWER LAW MODEL

1) Power law modelling with general masked data.
2) Maximum Likelihood estimation.
3) Performance measure Evaluation(MSE).
4) To find Reliability:

i.e. Estimating and predicting number of failures.

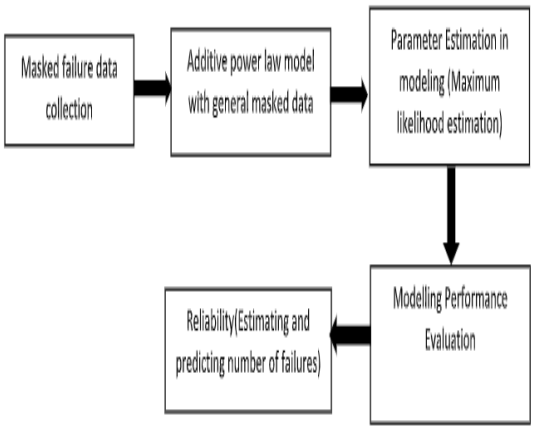

Figure 3: Flow chart for Power-law process

Steps involved in the algorithm is given below,

The Power law-based repair rate for a Non Homogenous Poisson process.

Step 1: Power law intensity function.

Power law intensity function is known as follows,

$$Y_{p-w(t)} = \gamma\beta t^{\beta-1} \text{-------- (9)}$$

Hence,

$\gamma(>0)$, Scale parameter and $\beta(>0), Shape\ parameter$

Step 2: Mean value function calculation.

Using the intensity function Equation (9), the mean value function is calculated as follows:

$$n_{p-w}(\theta) = \int_0^t \quad Y_{P-W}(s)ds = \gamma t^{\beta} \text{--------- (10)}$$

$\theta = (\gamma, \beta)$ is the parameter space

Step 3: Evaluating Maximum Likelihood function

The likelihood function is followed in equation (6)

$$W_{NHPP}(x) = \left(\pi_{j=1}^{m} \gamma\beta x_j^{\beta-1}\right) exp[-\gamma x_m^{\beta}] \text{ --- (11)}$$

Where, $x = x_1, x_2, \ldots \ldots x_m$

The parameter estimation method was used with the maximum likelihood estimation method. The expression for the log-likelihood function in Equation (9) is as follows.

$$lnW_{NHPP}(x) = mln\gamma + nln\beta - (\beta - 1)\sum_{j=1}^{m} \quad lnx_j - \gamma x_m^{\beta} \text{-------- (12)}$$

The following equation for the maximum likelihood estimate of each parameter is satisfied by γMLE and βMLE using the formulas (12).

$$\frac{\partial lnW_{NHPP}(\theta|x)}{\partial \gamma} = \frac{m}{\gamma} - x_m^{\beta} \text{-------- (13)}$$

$$\frac{\partial lnW_{NHPP}}{\partial \beta} = \frac{m}{\beta} - \sum_{j=1}^{m} \quad lnx_j - \gamma x_m^{\beta} lnx_m = 0 \text{--- (14)}$$

Step 4: Calculating Reliability

The following likelihood function is used to calculate the reliability.

$$fX_1, X_2, \ldots . X_m(x_1, x_2, \ldots x_m) = e^{-n(x_m)} \pi_{j=1}^{m} Y(x_j) \text{ (15)}$$

$$R(x_m) = exp[-\beta(\delta + x_m)^{\beta} + \gamma x_m^{\beta}] \quad \text{------ (16)}$$

Where δ *be the mission time*

In Power law process, By calculation MSE (Mean square error) in different iteration the number of failure rate decreases. The next section explains the results obtain from the Gompertz model and discusses it in detail.

B. Gompertz Intensity function:

The Gompertz distribution is a continuous probability distribution in probability and statistics. Demographers and economists frequently use the Gompertz distribution to describe the distribution of adult lifespans. The Gompertz distribution was also used to analyze survival in related domains such as biology and gerontology.

Steps involved in the Gompertz function is given below,

Step 1: The probability density function (pdf) and cumulative distribution function (cdf) for the Gompertz distribution employing several fields of industrial distribution.

$$f(\gamma.\beta) = \gamma\beta e^{\beta t} e^{\gamma} exp(-\gamma e^{\beta t}) \text{-------- (17)}$$

$$F(\gamma, \beta) = 1 - exp(-\gamma(e^{\beta t} - 1) \text{ ----- (18)}$$

Step 2:To calculate Hazard function using NHPP model.

In a state of perpetual failure. The NHPP has the same hazard function and intensity. Equations (17) and (18) are used to calculate the hazard function:

$$K(t) = \frac{F\prime(t)}{1-F(t)} = \gamma^{\beta} e^{\beta t} = Y_{I-P}(t)\text{--------- (19)}$$

Hence,

$\gamma(>0)$, Scale parameter and $\beta(>0), Shape\ parameter$

Using the intensity function Equation (18), the mean value function is calculated as follows:

$$n_{I-Q}(\theta) = \int_0^t \quad Y_{I-Q}(s)ds = \gamma(e^{\beta t} - 1)\text{-------- (20)}$$

Step3: Estimate MLE Parameters

The parameter estimation approach was employed with the maximum likelihood estimation method.

$$W_{NHPP}(x) = \left(\pi_{j=1}^{m} \gamma\beta e^{\beta x_j}\right) exp[-\gamma(e^{\beta x_m} - 1]\text{-------- (21)}$$

γMLE and βMLE meet the following equation for the maximum likelihood estimate of each parameter using the equations (21).

$$\frac{\partial lnW_{NHPP}}{\partial \beta} = \frac{m}{\beta} - \sum_{j=1}^{m} \quad x_j - \gamma x_m e^{\beta x_m} = 0\text{------- (22)}$$

Step4:To calculate Reliability

Hence, reliability is calculated as follows

$$R(x_m) = exp[-\{n(\delta + x_m) - n(x_m)\}]\text{-------- (23)}$$

where ,$\delta\ be\ the\ mission\ time$

By calculating the performance measures compared to the Power law, Gompertz NHPP model has small value. The next section explains the results obtain from the Musa okumoto model and discusses it in detail.

C. Musa-Okumoto Function:

The Musa-Okumoto model, commonly known as the logarithmic Poisson execution time model, implies that all errors are equally likely to occur and are independent of one another. In this model, the predicted number of faults is a logarithmic function of time, and the failure intensity reduces exponentially as the expected number of faults increases. Finally, the software will fail an infinite number of times over an indefinite period of time.

The Musa okumoto model assumes that the failure intensity function decreases exponentially with the number of failures observed.

$$X(\mu) = X_0\, e^{-\theta\mu(t)} \text{-------(1)}$$

Differentiate the above equation (1)

$$\frac{d\mu(t)}{dt} = X_0\, e^{-\theta\mu(t)}\text{------(2)}$$

$$X_0 = \frac{d\mu(t)}{dt} e^{\theta\mu(t)}\text{------(3)}$$

Hence, $$de\,\frac{\theta\mu(t)}{dt} = \theta\,\frac{d\mu(t)}{dt} e^{\theta\mu(t)}\text{-------(4)}$$

Substitute equation equation (3) in equation (4)

$$de\frac{\theta\mu(t)}{dt} = \theta X_0 \text{----------(5)}$$

By Integrating above equation (5)

$e^{\theta\mu(t)} = \theta t X_0 + A$ ----------(6)

Where $\mu(0) = 0, A = 1$ in equation(6)

$$'\mu(t)' be\ the\ mean\ value\ function$$

$$\mu(t) = \frac{ln(\theta t X_0 + 1)}{\theta} \text{ --------(7)}$$

The exponentially declining failure intensity indicates that the per-fault hazard rate takes the shape of a bathtub curve, as indicated by the Musa-Okumoto logarithmic model derivation by the fault exposure ratio.

D. Rayleigh model:

The Weibull lifespan distribution is a well-known model for life tests and reliability assessments. The cumulative distribution function and the probability density function with the shape parameter(γ) are the following:

$$f(t) = \frac{t^{\gamma-1}}{\beta^2} \text{ ---------(1)}$$

$$F(t) = (1 - e^{\frac{-t^{\gamma}}{2\beta^2}})\text{--------(2)}$$

In equation (1) and (2) sub, $\frac{1}{2\beta^2} = c$

$F(t) = (1 - e^{-ct^{\gamma}})$--------(3)

Estimate MLE function same as above models**.** Finally derive the parameters of $\theta\ and\ c$.

$\frac{\partial lnW_{NHPP}(\theta|x)}{\partial\theta} = \frac{m}{\theta} - 1 + e^{-cx_m{}^2} = 0$ ----------(4)

$\frac{\partial lnW_{NHPP}(\theta|x)}{\partial c} = \frac{m}{c} - \sum_{j=1}^{m} x_j{}^2 - \gamma\theta x_m e^{-cx_{m^2}} = 0$ ----------(5)

The equation (4) and (5) for the maximum likelihood estimate of each parameter is satisfied by θMLE and cMLE can be calculated numerically. Next section explains about the estimated model.

E. GO model:

A software reliability model's main goal is to predict software failure behavior when it's in use. This expected behavior changes quickly, and it may be observed throughout the program's testing time. Between failures, the execution times are exponentially spread. The expected value function of the cumulative number of failures follows a Non-Homogeneous Poisson process (NHPP). During the time that the software is being observed, the quantities of accessible resources remain constant. The number of problems discovered in each of the intervals is unrelated to the other. The expected number of error occurrences for each time x to $x + \Delta x$ is proportional to

the expected number of undetected mistakes at time t, according to the mean value function. It's also expected to be a non-decreasing, limited function of time, with

$$lim\, x \to \infty\, \mu(x) = M < \infty$$

If a fault causes a failure, it must be fixed immediately; otherwise, the failure will not be counted again.

$$\mu(x) = m(1 - e^{-nt}))$$

In above equation,

$m \rightarrow$ Expected number of defects

$n \rightarrow Shape\ factor$(i.e., Failure rate gets decreases)

$$\mu(x) \rightarrow Predicted\ number\ of\ defects$$

Steps involved in this model are given below,

Step 1: Number of defects that should be expected

Step2: Factor of roundness

Step3: Data obtained throughout the testing period

These parameters can be calculated using any statistical inference approach. once the failure data in terms of execution time is known. The parameters accuracy improves as the number of failures in the sample increases.

F. Finite fault NHPP reliability with Cauchy distribution:

Augustin Cauchy and Hendrik Lorentz are the names given to the Cauchy-Lorentz distribution.

The following diagram depicts the fault detection and rectification process:

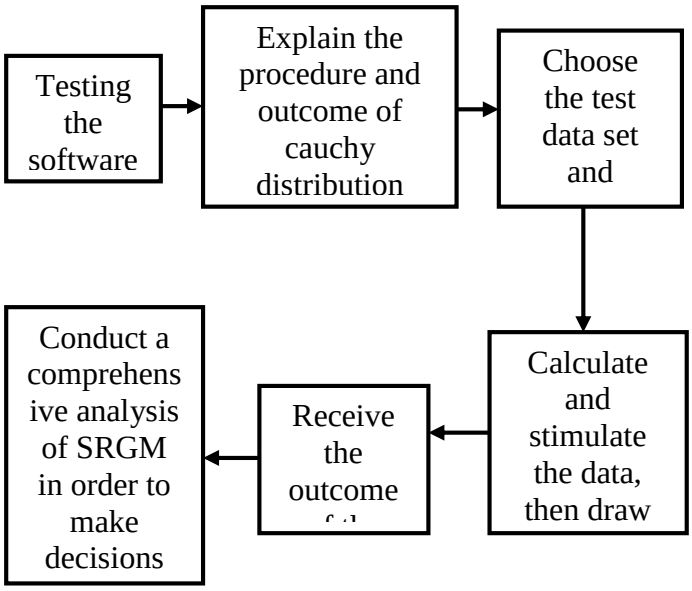

Figure 4:flowchart of software reliability growth model

Steps to be involved in Cauchy distribution:

Step1: To calculate CDF and PDF

It is a continuous probability distribution with the following probability distribution function PDF:

$f(z, z_0, \alpha) = \frac{1}{\pi}(\frac{\alpha}{(z-z_0)^2+\alpha^2})$--------(1)

$$z_0 \rightarrow Location\ parameter$$

$$\alpha \rightarrow Scale\ parameter$$

If $z_0 = 0, \alpha = 1$ (Standard cauchy distribution)

The Cauchy distribution is implemented using standard library tan and atan functions, which should result in very low error rates. Tangent lines are the foundation of Newton's method. The Newton-Raphson method (also known as Newton's method) is a method for quickly calculating the root of a real-valued function f y) =0. It is based on the premise that a straight line tangent to a continuous and differentiable function can approximate it.

The function of cumulative distribution is,

$$F(z; z_0, \alpha) = \frac{1}{\pi} arctan\ arctan\ \left(\frac{z-z_0}{\alpha}\right) + \frac{1}{2}$$---------(2)

Step2: Inverse CDF

The Cauchy distribution's inverse cumulative distribution function is.

$F^{-1}(q; z_0, \alpha) = z_0 + \alpha tan[\pi - \left(q - \frac{1}{2}\right)]$--------(3)

$z_1, z_2, \ldots\ldots z_n$ i.e.,Each random variable has a Cauchy distribution.

Where, z_0: $median$

The characteristic function is:

$\emptyset_z(u, z_0, \alpha) = E(e^{izu}) = exp(iz_0 - \alpha|u|)$ ---------(4)

The **mean** is not defined because,

$\int_{-\infty}^{\infty} zf(z)$ ---------(5)

$\int_{-\infty}^{\infty} zf(z) - \int_{-\infty}^{0} |z|f(z)$ ---------(6)

Both the positive and negative elements of the equation (6) are infinite in the case of a cauchy distribution. Hence, equation (6) is undefined.

Step 3: **Hazard function**

The hazard function of Cauchy is,

$$K(z) = \frac{1}{(1+Z^2)(0.5\pi - arctanz)}$$ --------(7)

Most general-purpose statistical software products provide skewness and kurtosis coefficients are shown in performance measures. The next section explains the estimated and predicted model and discusses it in detail.

The first raw moment about zero and the second raw moment about the mean represents the mean and variance, respectively. Skewness and kurtosis, the third and fourth moments about the mean, are also utilized as numerical representations of shape in risk analysis.

Step 4: Variance

The variance is a metric that indicates how far the probability distribution deviates from the mean: $V(Z) \geq 0$

The properties of variance are as follows: b is a constant, and Z, Zi are random variables.

$$V(bZ) = b^2V(Z) \text{ ---------(8)}$$

$$V(\sum_{j=1}^{z} Z_j) = \sum_{j=1}^{z} V(Z_j) \text{---------(9)}$$

Where 'Z' is uncorrelated. Next process skewness is expressed below.

Step 5: Skewness and Kurtosis:

The following formulae are used to calculate the skewness statistic:

$$S = \sum_{j=1}^{z} \frac{(z_j-\mu)^3 r_j}{\sigma^3} \quad \text{(Discrete variable)}$$

$$S = \int_{min}^{max} \frac{(z-\mu)^3 f(z)dz}{\sigma^3} \quad \text{(Continuous variable)}$$

When a distribution has a negative skewness (also known as left skewed), the tail to the left is longer than the tail to the right. Positively skewed (right skewed) distributions have a longer tail to the right, whereas zero skewed distributions are normally symmetric.

The following formulae are used to calculate the Kurtosis statistic:

$$K = \sum_{j=1}^{z} \frac{(z_j-\mu)^4 r_j}{\sigma^4} \quad \text{(Discrete variable)}$$

$$K = \int_{min}^{max} \frac{(z-\mu)^4 f(z)dz}{\sigma^4} \quad \text{(Continuous variable)}$$

Using above equations, kurtosis statistic measures the distribution's peakedness (see right panel above) the higher the kurtosis, the more peaked the distribution. Based on Cauchy distributions, this research provides a Novel Evaluation Model for Software Reliability Prediction. This paper investigates the performance of software reliability by using 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.

IV. ***RESULTS***

Here in this section we will discuss about result and comparison between proposed model and previous models using some of parameters

Conditional Reliability:

Conditional reliability is defined as the likelihood that a component or system will run without failure for a mission period. K(t) is the conditional failure probability:

$$K(t)=P(t \leq Z \leq t+\Delta t | z>t)$$

$$K(t)=(P(t \leq Z \leq t+\Delta t) \cap P(Z>t))/(P(Z>t))$$

The failure time is Z. The survival or reliability function at time t, i.e. P(Z>t) = E(t), is the denominator by definition (t)

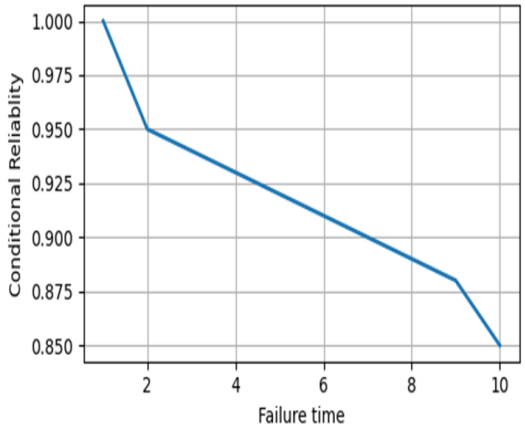

Figure 5:Plot for the model of Conditional Reliability

The above plot shows between the failure time and conditional reliability.where for different failure times, conditional probability gets decreases.

The mean square error result has been shown in the below plot, MSE (which evaluates the difference between the actual value and the predicted value) was used to compare software models.Figure11: Shows compared to Power-law, Musa-okumoto, and Gompertz model,the Proposed method failure times are reduced.

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.

Figure(12) shows compared to other Software reliability growth model, the proposed model, predictive relative error gets decrease with better testing progress (%).

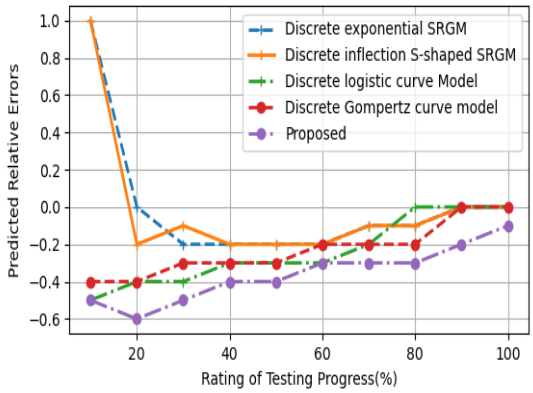

Figure 6:Rating of tesing progress(%) Vs Predictive relative errors

The MSF (mean of square fitting faults) is a metric that indicates how close a fitted line is to data points. Take the vertical distance between each data point and the corresponding value on the curve fit (the error) and square the value.

Testing Time	GO	WE-TEFM	GGO	Proposed
2	20	15	18	12
4	21	18	20	15
6	25	24	24	20
8	28	26	27	22
10	30	28	29	25
12	34	33	33	30
14	40	36	38	32
16	45	42	43	35
18	50	48	49	45
20	60	55	57	50

Table 1: Comparison metrics of Mean Square Fitting Faults with various Estimated and Predicted models.

The purpose of collecting fault and failure data is to be able to determine when the software is getting close to becoming fault-free. Table 2: Shows about the number of failure with different models.

Models	MSF
Logistic	118.29
Weibull	122.09
Rayleigh	268.42
Exponential	140.66
Proposed	101.52

Table 2: Number of failure

Using Table 2, the graph is plotted between Testing time with Number of failure.

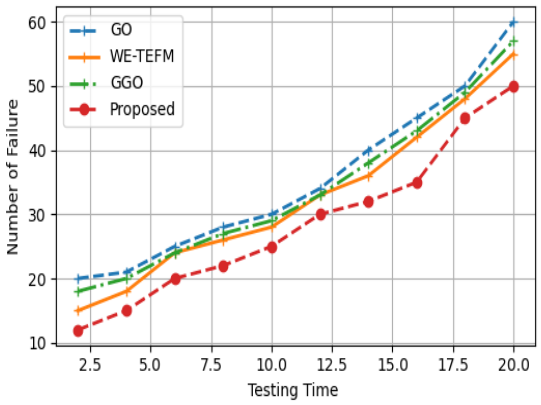

Figure 7:Testing time Vs Number of failure

The acquired results shows that the proposed model has less number of failures compared to other models.

V. CONCLUSION

The software reliability growth model can be used to calculate the best software release time and the cost of testing. To reduce the cost of testing and raise the profit of releasing software, a more accurate model is required. The usage of a software cost model can assist in precisely predicting the best software release schedule. Unlike previous models, the proposed model considers the total number of defects identified by users during the software operation period or after its release, rather than assuming that residual faults that are not detected will be discovered by the user. The cost of actual fault debugging is less than the cost of eradicating all residual faults during the operating phase, as can be observed. As we can gather information number of failure is less than previous models. Hence we can say here we achieve improvement to predict software reliability.

REFERENCES

[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] K.Sahu and R. K. Srivastava. Soft Computing Approach for Prediction of Software Reliability. ICIC Express Letters,12(12),1213-1222, 2018.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] H. Pham, "Distribution Function and its Application in Software Reliability," International Journal of Performability Engineering, Vol. 15, No. 5, pp. 1306-1313. 2019.

[10] Xuemei ZhangHoangPham,"An analysis of factors affecting software reliability". Journal of Systems and Software;

[11] Herbert Hecht,Myron Hecht," Software reliability in the system context". IEEE Transactions on Software Engineering ; (Volume: SE-12, Issue: 1, Jan. 1986)

[12] Daniel R jeske,Hoang pham,"On the Maximum Likelihood Estimates for the Goel– Okumoto Software Reliability Model", The American Statistician ;Volume 55,2001-Issue 3.

[13] Lianfen Qian,"Dynamic Two phase truncated Rayleigh model for release date prediction of software".Journal software Engineering &Applications;2010,3,603-609.

[14] XiangLi," Mean-variance-skewness model for portfolio selection with fuzzy returns". European Journal of Operational Research;Volume 202, Issue 1, 1 April 2010.

[15] A.Yadav&R.A.Khan," Critical Review on Software Reliability Models". International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009.

[16] An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes", Computer Standards & Interfaces, vol. 53, pp. 1-32, August2017.

[17] G. Bavota, M. Gethers, R. Oliveto, D. Poshyvanyk and A. D. Lucia, "Improving software modularization via automated analysis of latent topics and dependencies", ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 23, no. 1, pp. 4, 2014.

[18] T. H. Chen, S. W. Thomas, M. Nagappan and A. E. Hassan, "Explaining software defects using topic models", Mining Software Repositories (MSR) 2012 9th IEEE Working Conference on, pp. 189- 198, 2012, June.

[19] T. H. Chen, S. W. Thomas, M. Nagappan and A. E. Hassan, "Explaining software defects using topic models", Mining Software Repositories (MSR) 2012 9th IEEE Working Conference on, pp. 189- 198, 2012, June.

[20] Daniel Rodriguez, Israel Herraiz, Rachel Harrison, Javier Dolado and José C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction", Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, May 13-14, 2014.

[21] C. Y. Huang and T. Y. Kuo, "Queueing-Theory-Based Models for Software Reliability Analysis and Management", IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 4, pp. 540-550, 2017.

[22] K. Muthukumaran, N. L. Bhanu Murthy, G. Karthik Reddy and M. Aruna, "Comparative study on effectiveness of standard bug prediction approaches", Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop, pp. 1-4, October 17-19, 2013.

[23] Kim Herzig and Nachiappan Nagappan, "empirically detecting false test alarms using association rules", Proceedings of the 37th International Conference on Software Engineering, May 16-24, 2015.

[24] P. Treleaven and C. Yingsaeree, "Algorithmic trading", IEEE Computer Society, vol. 44, no. 11, pp. 61-69,2011.

[25] F. Machida, J. Xiang, K. Tadano and Y. Maeno, "Lifetime extension of software execution subject to aging", IEEE Transactions on Reliability, vol. 66, no. 1, pp. 123-134, 2017.

[26] T. Mende and R. Koschke, "Effort-aware defect prediction models", Software Maintenance and Reengineering (CSMR) 2010 14th European Conference on, pp. 107-116, 2010,March.

[27] Cowlessur, Sanjeev & Pattnaik, Saumendra & Pattanayak, Binod. (2020). A Review of Machine Learning Techniques for Software Quality Prediction. 10.1007/978-981-15-1483-8_45.

[28] Camelia S¸ erbana,∗ , Mohsin Shaikhb Software reliability prediction using package level modularization metrics, ScienceDirect Procedia Computer Science 176 (2020) 908–917 1877-0509

[29] Saini G.L., Deepak Panwar, Vijander Singh* "Software Reliability Prediction of Open Source Software Using Soft Computing Technique"Recent Advances in Computer Science and Communications ,Volume 14 , Issue 2 , 2021 DOI : 10.2174/2213275912666190307165332

[30] Meng Yan, Yicheng Fang, David Lo, Xin Xia and Xiaohong Zhang, "File-level defect prediction: unsupervised vs. supervised models", Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, November 09-10, 2017.

[31] G. Nuti, YuchenGuo Mirghaemi, Martin Shepperd and Ning Li, "Bridging effort-aware prediction and strong classification: a just-in- time software defect prediction study", Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, May 27-June 03, 2018.

[32] Jadhav, R. B. Software Defect Prediction Utilizing Deterministic and Probabilistic Approach for Optimizing Performance through Defect Association Learning. International Journal of Emerging Trends in Engineering Research, 8(6), 2600–2605. https://doi.org/10.30534/IJETER/2020/62862020

[33] 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.

[34] K. L. Peng and C. Y. Huang, "Reliability analysis of on- demand service-based software systems considering failure dependencies", IEEE Transactions on Services Computing, vol. 10, no. 3, pp. 423-435, 2017.

[35] J. Rumbaugh, I. Jacobson and G. Booch, "Unified modeling language reference manual the", Pearson Higher Education, 2004.

[36] DHARMENDRA LAL GUPTA1,2,* and KAVITA SAXENA2 Software bug prediction using object-oriented metrics Sa¯dhana¯ Vol. 42, No. 5, May 2017, pp. 655–669 , Indian Academy of Sciences

[37] Cowlessur, Sanjeev & Pattnaik, Saumendra & Pattanayak, Binod. (2020). A Review of Machine Learning Techniques for Software Quality Prediction. 10.1007/978-981-15-1483-8_45.

[38] Camelia S¸ erbana,Ø , Mohsin Shaikhb Software reliability prediction using package level modularization metrics, ScienceDirect Procedia Computer Science 176 (2020) 908–917 1877-0509

[39] Saini G.L., Deepak Panwar, Vijander Singh* "Software Reliability Prediction of Open Source Software Using Soft Computing Technique"Recent Advances in Computer Science and Communications ,Volume 14 , Issue 2 ,2021 DO**I** : 10.2174/2213275912666190307165332

[40] 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