Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7716449

RELIABILITY OF SOFTWARE RELIABILITY GROWTH MODELS: A REVIEW

Nofiya Yousuf Mir[1], Anwaar Ahmad Wani[2]
[1] Senior Assistant Professor, Islamia College of Science & Commerce, Srinagar, UT-J&K, India, email:- Ayifon15@gmail.com
[2]Assistant Professor (C), Higher Education Department, Government of J&K, Srinagar, UT- J&K, India, email:- anwaar1007@gmail.com

Abstract- Today we live in the world which is revolutionised by technology. Technology has become part of our everyday life: from accomplishing a simpletask like turning on a light bulb to launching satellites, technology has become an inseparable part of everyone's life. In order to be able to use this technology reliably, the software which with we interact with technology needs to be reliable. Software reliability refers toprobability that a software execution will be as expected under specified terms and conditions in a given amount of time without any fail. Software reliability is measured using software reliability growth models. Overtime hundreds of models have been developed so far. This paper aims toreview and examine several different non-homogenousPoisson process software reliability growth models.
Keywords— *Software reliability, non-homogenous Poisson process, Software Reliability Growth Models.*

I. Introduction

Software in today's technological world is an integral part of our day to day life. We interact with technology at out homes, in school, in our offices even in our cars; from searching on Alexa to teaching students on smart boards and finding routes on GPS or doing complex financial transactions weare rely on technology. We interact with technology by means of software's. In order to be able to use thissoftware for carrying out our chores efficiently & reliably, the software needs to possess certain characteristics or qualities [1] which include

- Functionality
- Usability
- Efficiency
- Maintainability
- Probability
- Reliability

Out of all the characteristics mentioned above Reliability is widely considered as key quality parameter. Software Reliability is defined as the "Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub- characteristics: Maturity, Availability, Fault tolerance and Recoverability"[2] as per ISO standards. Software Reliability being an intangible in nature is a bit tedious to measure. Since 1960s several scientists have established Models called thesoftware reliability growth models (SRGM) that calculate reliability of the software in terms of time for which software is expected to run reliably without any flaw. SRGMs also help software practitioner's in determining the right time to stop testing as to when the reliability is achieved and hence releasing the software[3].

The main objective of this paper is to go through a systematic approach for readers to gain knowledge about the software reliability, NHPP SRGMs and also put forward a critical review of what a particular model lacks and how consequent modeling technique overcame that/those drawback(s).This review paper also aims to present to readers , young scholars , researchers an insight into the growth, development of SRGMs in general.Thus it will also aim to persuade researcher to studyexisting model from a critical point of view motivating him/her to develop better models for Software Reliability.

II. Classification of SRGMs

Based on the parameters taken into consideration for measurement of reliability , this paper discusses SRGMs belonging to 3 categories aslisted below:

1. Perfect debugging models

Models under this category assume that a failure when encountered is removed in entirety in no time in addition to assumptions listed below:

- The failures are governed by non-homogenous Poisson process (NHPP).
- A failure once detected isremoved immediately.
- Removal of failure is completelyperfect.
- All faults in software are mutuallyindependent.

This category is further divided into two sub categories based on the fact whether time is calculated in terms of execution time of software orcalendar time.

Perfect debugging models are described by a general formula

$$y(t) = a(1-X)$$

Where

y(t) is no. of expected failures at time t

a is the estimated total no. of failures at an infinitetime i.e

$y(\infty) = a$

X is parameter defined differently by each models discussed below and summarized in Table 1

Execution time model

J D Musa [4] devised the basic execution model in which time *t* is expressed in execution units. The mean value function for failure occurrence for software under this model isgiven by formula

$$y(t) = a(1-X)$$

Where

$$X = e^{-\frac{\lambda_o t}{a}}$$

λ_o is initial failure intensity .

Calendar time model

Goel Okumoto (GO Model) [5]also known as exponential model , is defined by Non-homogenous Poisson Process (NHPP)in which failures are given by

$$y(t)=a(1-X)$$

$$X=e^{-bt}$$

$y(t)=a(1-e^{-bt})$. (1.1)where b is the fault removal rate, t is time expressed in calendar units.

Delayed S-Shaped Model

Then came the model given by Yamada & Osaki called the delayed S-shaped Model[6] which described testing as a 2- step process

Step I: fault detectionStep II: fault removal

For step I *y(t)* is same as that for GO i.e

$$y(t)=a(1-e^{-bt})$$

While for step II X is given as

$$X=(1+bt)e^{-bt}$$

Table1

	Value for X	**Formula**
Basic time execution model[4]	$e^{-(\frac{\lambda o}{a}t)}$	$y(t)=a(1-e^{-(\frac{\lambda o}{a}t)})$
GO model[5]	e^{-bt}	$y(t)=a(1-e^{-bt})$
Delayed S-shaped model Step I[6]	e^{-bt}	$y(t)=a(1-e^{-bt})$
Delayed S-shaped model Step II[6]	$(1+bt)e^{-bt}$	$y(t)=a(1-(1+bt)e^{-bt})$

They [6] further enhanced their model with better estimation techniques [7]Following GO, Ohba[8] proposed hyper exponential model that assumed software to contain independent clusters of independent units/modules. Each unit having a different initial number of errors and a different failure rates, such as new and existing/reused units, simple and composite units, and interactive units and units which do not interact.

Further Yamada & Osaki [9] assumed software to contain two types of faults; ones easy to detect and correct and another difficult to detect which resulted in a model called Fault Categorization Model. On thesimilar lines Erlang Model[10] assumed software tocontain 3 types of faults ; simple, hard& complex. Kapur & Garg [11]Another remarkable Model was added by Bittanti et al.[12] that replaced constant error rate b by initial and final Fault detection rates b_i & b_f. With mean value for occurrence of failures given by

$$y(t) = a\left(\frac{b_i\left(e^{b_f t} - 1\right)}{b_f + b_i\left(e^{b_f t} - 1\right)}\right)$$

Recently Hanagal and Bhalerao[13],[14],[15]obtained SRGMs based on general inverse exponential distribution & extended inverse exponential distributions.

All of the models discussed above lack onething i.e the presumption that the errors as and when found are debugged immediately which is unrealistic.

This is the critique point for most of the conventional SRGMs including the ones discussed above.The models discussed in the next segment try to overcome this by taking into account the imperfection debugging as a parameter for SRGMs.

2. IMPERFECT DEBUGGING MODELS

As the name implies this section discusses models where debugging process is not perfect as assumed in models we discussed above.Models that work onthe principle

a) that a debugging process can induce new faults into the software and /or

b) detected faults are not removed completely

are known as Imperfect debugging models. The models described here under this category are basedon the general assumptions of NHPP SRGMs listedin the first section except that the debugging processis not at all perfect which is quite realistic. In addition to listed assumption, each model discussedbelow have some additional conditions.

Pure error generation Model

Ohba & Chou [16] formulated a model under imperfect debugging that induced new faults during the debugging process. The inductions of new faults were assumed to occur at a constant rate 'α'. The overall mean value fault function for model is given by

$$y(t) = a(1-\alpha)(1-e^{-b(1-\alpha)t})$$

Pure Imperfect Debugging Model

Kapur & Garg[17] incorporated imperfect debugging by assuming that there exists a probability p that an error is perfectly detected & removed and hence the mean value function formodel is given separately for fault detection and fault removal and are defined by

$$y_d(t) = a(1-e^{-bpt})$$

$$y_r(t) = (a/p)(1-e^{-bpt})$$

Where $y_d(t)$ is the function for error detection

& $y_r(t)$ is the function for error removal at time t. This model was further improvised by Yamada etal.[18] by assuming fault introduction as exponentialin pure error generation model.

Testing Efficiency Model

Zhang et al.[19] developed a model that incorporated features from both the models discussed above i.e a)introduction of new faults from Ohba & Chou[16] & b)probabilistic removal of

$$y(t) = \frac{a}{p-\alpha}\left(1-\left(\frac{(1+\beta)e^{-bt}}{1+\beta e^{-bt}}\right)\right)^{(c/b)(p-\alpha)}$$

faults from Kapur & Garg[11]. The mean value fault function for model is given by

Another remarkable Model called PNZ[20]Model incorporated fault introduction probability into imperfect debugging.Kumar et al.[21] incorporated imperfect debugging into Yamada[6] Model called the delayed S-shaped model with pureerror generation. This way the researchers introduced imperfect debugging into SRGMs. In reality error detection and correction is a tedious andtime consuming process. It takes expertise and considerable amount of time to fix an error which tosome extent is accounted in the next section.

3. TESTING EFFORT BASED SRGMS

Testing is an integral part of Software DevelopmentLife Cycle and heart & soul of SRGMs. The idea behind development of testing effort based SRGMs was to incorporate testing resources(manpower, hardware and software)into regular SRGMs. Putnam[22] proposed use of Rayleigh model to describe the test-effort expenses time-dependent behavioral which is given by cumulative distribution of test efforts as

$$z(t)=(1-e^{\alpha t^2}) \qquad (2.1)$$

Yamada et al. [23] presumed the fault detection rateis dependent to the amount of test effort used duringthe testing phase and is proportionate to the present failure content .Thus the amount of test effort used throughout the software testing phase was incorporated into a software-reliability growth model. The model is devised using a non- homogeneous Poisson process. Yamada[23]incorporated z(t) from "(2.1)" in GO model yielding

$$y(t)=a(1-e^{-bz(t)})$$

Where *b* is the fault detection/removal rate per remaining error.

The technique for data analysis for measuring software dependability is built using the model. This model is used to anticipate the cost of further test efforts needed to meet the objective number of software testing faults found, as well as the best time to end software testing before a release.Yamada [24] soon developed model calculating test effort using Weibull distribution andincorporating it in SRGM as above. On the similar lines, Bokhari & Ahmad[25] used formulated Log Logistic Test Effort model followed by Kapur et al.[26] and Khatri[27], [28]. They didn't stopped here Khatri[29] began studying Ohba & Chou Imperfect Debugging Model.They came up with [30]Model beautifully quantifying Software Reliability under imperfect debugging and incorporating testing effort '*z(t)*' in Ohba & Chou[16]as

$$y(t)= \quad a/(1-\alpha) \quad (1-e^{-pb(1-\alpha)z(t)})$$

The next step in SRGM development was to incorporate Testing effort in imperfect debugging Models and initiated by Kapur et al.[31] This was followed by incorporating test effort and imperfect debugging in Yamada's Delayed S-shaped SRGM [6], [32]Later Khatri[33] Kapur et al.[34] proposed a generalized framework utilizing Test effort for modeling multi release of a software introducing the effect of fault reduction factor.
The question still remains that even with better test efforts; testing process is never going to be simple, it is still going to be complex & time consuming. In SRGMs discussed above certain factors are assumed to remain constant over time. However it may not be the case , which is whythe next category of Models came into existence.

4. CHANGE POINT MODELS

In the applications of SRGM it is assumed that the debugging /testing environment stays same. In reality this environment changes over time e.g test teams/tools/resources can change affecting the errordetection rate.Change Point Models capture such transitions and incorporate them into SRGMs.

Exponential Change Point Model

This model was given by Chang[35] suggesting single change point 'ρ' in basic GO [5] such that the rate of failure detection *b* before and after single change point is b1 and b2 respectively.

Using in 1.1, we have,

$$y(t)=\begin{cases} a(1-e^{-b_1 t}), & 0 \le t \le \rho, \\ a(1-e^{-(b1\rho+b2(t-\rho))}) & \rho < t \end{cases}$$

S-shaped Change Point model

Inoue & Yamada [36] devised a model by incorporating single change point in delayed S-shaped Model[6]'ρ' .The mean value fault functionfor model is given by

$$y(t)=\begin{cases} a(1-(1+b_1 t)e^{-b_1 t}), & 0 \le t \le \rho, \\ a\left(1-\frac{(1+b_1\rho)}{(1+b_2\rho)}(1+b_2 t)e^{-b_1\rho-b_2(t-\rho)}\right) & \rho < t \end{cases}$$

This was followed by Kapur[37] who devised modelwith single change points where failures followed different probability distributions before and after change point.In the similar manner Exponential Change point Model [38]under imperfect debuggingwith different fault induction rates before and after Change point followed by[39]. Recent advances[40] uses test effort with fault reduction factor, underimperfect debugging in change point.A pioneering breakthrough was led by Huang[41] introducing Multiple Change Points. Later Kapur & Khatri[42],[43],[44],[45][46]introduced multiple change point by means of categorisation of faults as easy and difficult faults.

III. **Conclusion:**

Some of major NHPP models that have appeared in literature are discussed in this paper. Reliability models are a powerful tool for estimating, administering and examining software reliability. They are especially useful to describe reliability growth and fault deterioration, making itsimple to analyse software reliability & predict software release . In this paper, we first studied the basic execution model followed by GO model to provide a quantification for software reliability. In order to capture both execution and calendar time, we looked at a few widely used variations of NHPP models based on time-dependent transition probabilities. Then, we studied the modelling approach utilising test efforts. We further discussedmodel incorporating much more realistic imperfectdebugging in which the earlier models can be tailored using probabilities of fault correction. Finally, we discussed change point models where in , the change(s) in software environment is included in estimation of failures. By increasing the testing effort intensity and properly allocating and managing the testing resources and by adding morerealistic parameters to these SRGMs can aid in the removal of flaws, assisting software practitioners indetermining when a software system is prepared for release and if its reliability has reached a predetermined threshold

REFERENCES

[1] P. K. Kapur, G. Singh, N. Sachdeva, and A. Tickoo, "Measuring software testing efficiency using two-way assessment technique," in *Proceedings of 3rdInternational Conference on Reliability ,Infocom Technologies and Optimization*, IEEE, Oct. 2014, pp. 1–6. doi: 10.1109/ICRITO.2014.7014679.

[2] "Reliability ISO."https://iso25000.com/index.php/en/is o-25000-standards/iso-25010/62-
reliability (accessed Jan. 12, 2023).

[3] R. Majumdar, P. K. Kapur, and S. K.Khatri, "Measuring testing efficiency & effectiveness for software upgradation and its impact on CBP," in *2016 International Conference on Innovationand Challenges in Cyber Security (ICICCS-INBUSH)*, IEEE, Feb. 2016, pp. 123–128. doi: 10.1109/ICICCS.2016.7542347.

[4] J. D. Musa, "A theory of software reliability and its application," *IEEETransactions on Software Engineering*, vol. SE-1, no. 3, pp. 312–327, Sep. 1975, doi: 10.1109/TSE.1975.6312856.

[5] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and OtherPerformance Measures," *IEEE Trans Reliab*, vol. R-28, no. 3, pp. 206–211,Aug. 1979, doi:10.1109/TR.1979.5220566.

[6] S. Yamada, M. Ohba, and S. Osaki, "s- Shaped Software Reliability Growth Models and Their Applications," *IEEE Trans Reliab*, vol. R-33, no. 4, pp. 289–292, Oct. 1984, doi: 10.1109/TR.1984.5221826.

[7] S. Yamada and S. Osaki, "Software Reliability Growth Modeling: Models and Applications," *IEEE Transactions onSoftware Engineering*, vol. SE-11, no. 12, pp. 1431–1437, Dec. 1985, doi: 10.1109/TSE.1985.232179.

[8] M. Ohba, "Software reliability analysismodels," *IBM J Res Dev*, vol. 28, no. 4,pp. 428–443, Jul. 1984, doi: 10.1147/rd.284.0428.

[9] S. Yamada and S. Osaki, "Optimalsoftware release policies with simultaneous cost and reliability requirements," *Eur J Oper Res*, vol. 31, no. 1, pp. 46–51, Jul. 1987, doi: 10.1016/0377-2217(87)90135-4.

[10] P. K. Kapur, S. Younes, and S. Agarwala,"A General Discrete Software Reliability

Growth Model," *International Journal of Modelling and Simulation*, vol. 18, no. 1, pp. 60–65, Jan. 1998, doi: 10.1080/02286203.1998.11760358.

[11] P. K. Kapur and R. B. Garg, "A software reliability growth model for an error- removal phenomenon," *Software Engineering Journal*, vol. 7, no. 4, p. 291, 1992, doi: 10.1049/sej.1992.0030.

[12] S. Bittanti, P. Bolzern, E. Pedrotti, M. Pozzi, and R. Scattolini, "A flexible modeling approach for software reliability growth," in *Software Reliability Modelling and Identification*, Berlin/Heidelberg: Springer-Verlag, pp. 101–140. doi: 10.1007/BFb0034288.

[13] D. D. Hanagal and N. N. Bhalerao, "Modeling and Statistical Inference on Generalized Inverse Weibull Software Reliability Growth Model," *Journal of the Indian Society for Probability and Statistics*, vol. 17, no. 2, pp. 145–160, Dec. 2016, doi: 10.1007/s41096-016-0010-8.

[14] D. D. Hanagal and N. N. Bhalerao, "Analysis of delayed s shaped software reliability growth model with time dependent fault content rate function," *Journal of Data Science*, vol. 16, no. 4, 2018.

[15] D. D. Hanagal and N. Bhalerao, "MODELING ON GENERALIZED EXTENDED INVERSE WEIBULL SOFTWARE RELIABILITY GROWTH MODEL.," *Journal of Data Science . Jul2019, Vol. 17 Issue 3, p573-589*, vol. 17, no. 3, pp. 573–589, 2019.

[16] M. Ohba and Xiao-Mei Chou, "Does Imperfect Debugging Affect Software Reliability Growth?," in *11th International Conference on Software Engineering*, IEEE, 1989, pp. 237–244. doi: 10.1109/ICSE.1989.714425.

[17] Kapur P.K. and Garg R. B., " Optimal sofware release policies for software reliability growth models under imperfect debugging. (1990) no. 3, pp. 295-305. ," *RAIRO – Operations Research - Recherche Opérationnelle, Volume 24* , pp. 295–305, 1990.

[18] S. YAMADA, K. TOKUNO, and S. OSAKI, "Imperfect debugging models with fault introduction rate for software reliability assessment," *Int J Syst Sci*, vol. 23, no. 12, pp. 2241–2252, Dec. 1992, doi: 10.1080/00207729208949452.

[19] Xuemei Zhang, Xiaolin Teng, and Hoang Pham, "Considering fault removal efficiency in software reliability assessment," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 1, pp. 114–120, Jan. 2003, doi: 10.1109/TSMCA.2003.812597.

[20] Hoang Pham, L. Nordmann, and Zuemei Zhang, "A general imperfect- software-debugging model with S- shaped fault-detection rate," *IEEE Trans Reliab*, vol. 48, no. 2, pp. 169–175, Jun. 1999, doi: 10.1109/24.784276.

[21] D. Kumar, Kapur R, V. Sehgal, and P. Jha, "On the development of software reliability growth models with two types of imperfect debugging," *Commun Dependability Qual Manag Int J*, vol. 10, no. 3, pp. 105–122, 2007.

[22] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering*, vol. SE-4, no. 4, pp. 345–361, Jul. 1978, doi: 10.1109/TSE.1978.231521.

[23] S. YAMADA, J. HISHITANI, and S. OSAKI, "Test-effort dependent software reliability measurement," *Int J Syst Sci*, vol. 22, no. 1, pp. 73–83, Jan. 1991, doi: 10.1080/00207729108910590.

[24] S. Yamada, J. Hishitani, and S. Osaki, "Software-reliability growth with a Weibull test-

effort: a model and application," *IEEE Trans Reliab*, vol. 42,no. 1, pp. 100–106, Mar. 1993, doi: 10.1109/24.210278.

[25] MU Bokhari and N. Ahmad, "Proceedings of the 17th IASTED international conference on modeling and simulation (MS'2006), Montreal, Canada," in *ANALYSIS OF A SOFTWARE RELIABILITY GROWTH MODELS: THE CASE OF LOG-LOGISTIC TEST-EFFORT FUNCTION*, Montreal: Proceedings ofthe 17th IASTED international conference on modeling and simulation (MS'2006), Montreal,Canada, 2006, pp. 540–545.

[26] V. B. Singh, P. K. Kapur, and A. Tandon,"Measuring reliability growth of software by considering fault dependency, debugging time Lag functions and irregular fluctuation,"*ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 1–11, May 2010, doi: 10.1145/1764810.1764831.

[27] S. K. Khatri, D. Kumar, A. Dwivedi, and N. Mrinal, "Software Reliability Growth Model with testing effort using learning function," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, IEEE, Sep. 2012, pp. 1–5. doi: 10.1109/CONSEG.2012.6349470.

[28] A. Tickoo, P. K. Kapur, A. K. Shrivastava,and S. K. Khatri, "Testing effort based modeling to determine optimal releaseand patching time of software," *International Journal of SystemAssurance Engineering and Management*, vol. 7, no. 4, pp. 427– 434, Dec. 2016, doi: 10.1007/s13198-016-0470-y.

[29] KHATRI S K, Jagvinder Singh, Avneesh Kumar, and Adarsh Anand, "A Discrete Formulation of Successive Software Releases Based on Imperfect Debugging," *MIS Review :An International Journal*, Sep. 2014.

[30] S. K. Khatri, S. A. John, and R. Majumdar, "Quantifying software reliability using testing effort," in *2016International Conference on Information Technology (InCITe) -The Next Generation IT Summit on the Theme - Internet of Things: Connectyour Worlds*, 2016, pp. 23–26. doi: 10.1109/INCITE.2016.7857582.

[31] P. K. Kapur, P. S. Grover, and S. Younes, "Modelling an imperfect debugging phenomenon with testing effort," in *Proceedings of 1994 IEEE InternationalSymposium on Software Reliability Engineering*, IEEE Comput. Soc. Press, pp. 178–183. doi: 10.1109/ISSRE.1994.341371.

[32] R. Peng, Y. Li, W. Zhang, and Q. Hu,"Testing effort dependent softwarereliability model for imperfect debugging process considering both detection and correction.," *Reliab Eng Syst Saf*, vol. 126, pp. 37–43, 2014.

[33] V. KUMAR, S. K. KHATRI, H. DUA, M. SHARMA, and P. MATHUR, "AN ASSESSMENT OF TESTING COST WITH EFFORT-DEPENDENT FDP AND FCP UNDER LEARNING EFFECT: A GENETIC ALGORITHM APPROACH," *International Journal of Reliability, Quality and Safety Engineering*, vol. 21, no. 06, p.1450027, Dec. 2014, doi:10.1142/S0218539314500272.

[34] P. K. Kapur, P. Mishra, A. K. Shrivastava,and S. K. Khatri, "Multi release modeling of a software with testingeffort and fault reduction factor," in*2016 International Conference on Innovation and Challenges in CyberSecurity (ICICCS-INBUSH)*, IEEE, Feb.2016, pp. 54–59. doi:10.1109/ICICCS.2016.7542352.

[35] Y.-P. Chang, "ESTIMATION OF PARAMETERS FOR NONHOMOGENEOUS POISSON PROCESS: SOFTWARE RELIABILITY WITH CHANGE-POINT MODEL,"

Commun Stat Simul Comput, vol. 30, no. 3, pp. 623–635, Aug. 2001, doi: 10.1081/SAC-100105083.

[36] S. Inoue and S. Yamada, "Optimal software release policy with change-point," in *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE, Dec. 2008, pp. 531–535. doi: 10.1109/IEEM.2008.4737925.

[37] P. K. Kapur, J. Kumar, and R. Kumar, "A Unified Modeling Framework Incorporating Change-Point for Measuring Reliability Growth daring Software Testing," *OPSEARCH*, vol. 45, no. 4, pp. 317–334, Dec. 2008, doi: 10.1007/BF03398823.

[38] H.-J. Shyur, "A stochastic software reliability model with imperfect-debugging and change-point," *Journal of Systems and Software*, vol. 66, no. 2, pp. 135–141, May 2003, doi: 10.1016/S0164-1212(02)00071-7.

[39] C.-Y. Huang, "Performance analysis of software reliability growth models withtesting-effort and change-point," *Journal of Systems and Software*, vol. 76, no. 2, pp. 181–194, May 2005, doi: 10.1016/j.jss.2004.04.024.

[40] S. Khurshid, A. K. Shrivastava, and J. Iqbal, "Effort based software reliability model with fault reduction factor, change point and imperfectdebugging," *International Journal of Information Technology*, vol. 13, no. 1, pp. 331–340, Feb. 2021, doi: 10.1007/s41870-019-00286-x.

[41] Chin-Yu Huang and Chu-Ti Lin, "Reliability Prediction and Assessment of Fielded Software Based on Multiple Change-Point Models," in *11th Pacific Rim International Symposium on Dependable Computing (PRDC'05)*, IEEE, pp. 379–386. doi: 10.1109/PRDC.2005.52.

[42] P. K. KAPUR, A. KUMAR, K. YADAV, and S. K. KHATRI, "SOFTWARE RELIABILITY GROWTH MODELLING FOR ERRORS OF DIFFERENT SEVERITY USING CHANGE POINT," *International Journal of Reliability, Quality and Safety Engineering*, vol. 14, no. 04, pp. 311–326, Aug. 2007, doi: 10.1142/S0218539307002672.

[43] A. A. Wani, L. Faisal, M. Zahoor, and J. Iqbal, "Design & Development of Novel Hybrid Set of Rules for Detection and type of Malignant or Non-Malignant Tumor in Human Brain based on SVM Using Artificial Intelligence Classifier," Mathematical Statistician and Engineering Applications, vol. 71, no. 4, pp. 10253–10276, Dec. 2022, Accessed: Apr. 02, 2023. [Online]. Available: https://www.philstat.org/index.php/MSEA/article/view/1853.

[44] R. Mohd, A. M. Azad, A. A. Wani, and I. ahmad Bhat, "A Prognostic Approach For Precipitation Forecast Using Naive Bayes Algorithm," Solid State Technology, vol. 63, no. 6, pp. 7435–7444, Nov. 2020, Accessed: Apr. 02, 2023. [Online]. Available: http://solidstatetechnology.us/index.php/JSST/article/view/4603.

[45] D. A. A. Wani, D. J. Iqbal, and D. M. Makhdoomi, "Modelling an Intrusion Detection system using ensemble approach based on voting to improve accuracy of base classifiers," JOURNAL OF ALGEBRAIC STATISTICS, vol. 13, no. 2, pp. 1844–1865, Jun. 2022, Accessed: Apr. 02, 2023. [Online]. Available: https://publishoa.com/index.php/journal/article/view/360.

[46] A. Wani and S. Dixit, "Enhanced Frame Aggregation Scheduler (EFAS) for data Transmission over IEEE802.11ac," https://old.rrjournals.com/, May 25, 2019. https://old.rrjournals.com/past-issue/enhanced-frame-aggre