Educational Administration: Theory and Practice
2024,30(4), 3563-3568
ISSN:2148-2403
https://kuey.net/

Research Article

A Stochastic Analysis Of Computing Models In Computer System Reliability Based On Software

Chaudhary Ashish[1*], Milind[2]

[1*], [2]Department of Computer Science and Engineering, SCRIET, Chaudhary Charan Singh University, Meerut, India.
Email: ashishchaudhary3015@gmail.com, milindccsu@yahoo.com

Citation: Chaudhary Ashish, et al. (2024), A Stochastic Analysis Of Computing Models In Computer System Reliability Based On Software, *Educational Administration: Theory and Practice*, *30*(4), 3563-3568
DOI:10.53555/kuey.v30i4.2079

ARTICLE INFO

ABSTRACT

Background

The backdrop concentrates on the research and creation of models for software dependability, with an emphasis on the utilization of Non-Homogeneous Poisson Process (NHPP) prototypes to forecast that rate of software malfunction. Software reliability analysis and simulation methodologies are getting better thanks to the efforts of researchers and academics in computer science, software engineering, and related fields. The significance of precise modelling for software reliability in terms of boosting software quality, making the most of testing, and boosting software systems' overall reliability. Software reliability can be anticipated and evaluated using a variety of models and methods, such as failure detection, fault repair techniques, and dependable growth assumptions.

Objective

In order to predict and identify software failure rates, the study paper will examine and assess a variety of software reliability models, including stochastic models such as Non-Homogeneous Poisson Process (NHPP) models. The paper's objectives are to examine multiple models in respect to the Theil statistic and performance indicators such as variance, bias, root median square prediction mistake and the mean square mistake. The analysis's findings will show how precise and well-suited the recommended methods are to more normative growth models of software reliability. In order to anticipate software dependability, the study also discusses the construction of multiple change-point extended queueing models that take time delays in fault detection and rectification processes and defect recovery rates into account.

Developed throughout the previous three decades, software reliability growth models (SRGMs) are a valuable tool for estimating and forecasting software dependability. The majority of frequent SRGMs seem to think that errors will be corrected immediately. These expectations might not always be realistic or correct in real life. To debug software, programmers need to be able to replicate mistakes, identify their root causes, make the required adjustments, and then run the programme again. It takes time to complete this process. It's possible that the issue's clearance rate fluctuates as well may vary over time and between sites, according to some research or observations. Consequently, we shall look into the subject of how queueing models for mistake detection and correction define software development processes. We introduce an enhanced model for infinite server queuing that features numerous change-points to accurately estimate and assess software stability. The suggested model may be more accurate than traditional SRGMs in capturing the variance in fault rectification rates and influencing software development behaviour, according to empirical results and real failure statistics.

Keywords: software reliability; stochastic models; software reliability growth model; non homogeneous poisson process..

Introduction-

In relation to physical structures, software (and its design process) is a recent product. The vast majority of software engineers have dedicated a great deal of time and energy to learning how to create trustworthy, flawless software. Software reliability, as specified by ANSI, is the possibility that a programme is going to run error-free within an appropriate setting and time frame.

Prioritizing software source code produced by several successive stages is necessary for it to be regarded as trustworthy. Early intervention can save time as well as cash. trustworthiness predictions are observable and achievable. A few software reliability growth models (SRGMs) in light of the non-homogeneous Poisson process (NHPP) have evolved within the last three decades in an effort to assess and support the evolution of product dependability [1,4].

In theory, SRGMs allow programmers the power to decide as evaluation should end and the ability can observe how reliable software evolves over time. SRGMs are frequently employed in a variety of real-world settings. SRGMs are necessary for vital safety systems including the shuttle's onboard computer software and weaponry. In addition to ANSI/AIAA, prominent companies and research institutions like Bell Canada, AT&T, JPL, AFOTEC (the Air Force Operational Evaluation Centre), North Telecom, and Bell have also embraced or sponsored SRGMs. Baumer et al. employed NHPP models for several large software projects for Ericsson AB to forecast fault ingress. Their findings illustrated how well The instantaneous efficacy associated with the curved NHPP predictions forecasting. Within the industry, The prototypes from NHPP still remain in use [5,14].

Three important elements frequently affect software durability estimates: failure diagnosis, troubleshooting, and accessible profile. If an error occurs during deployment and a programme crashes, the fault can be identified and rectified. The idea behind the majority of rudimentary SRGMs is that mistakes are corrected as soon as they are discovered. Actually, it's possible that the aforementioned assumption isn't always correct or true.

It's common for problems to take some time to resolve once corrective action has begun. In the previous decade, several scholars have studied queueing-based techniques in an attempt to better understand debugging behaviour and allocate restricted testing resources. When using queueing-based software dependability modelling techniques, debuggers and found defects are frequently viewed as consumers and servers. The gap in the interval of time between a client's arrival and their departure following their experience is completed is used to calculate how long it takes to remedy a defect.

Related work

A study on the accuracy of software models has been conducted. great deal of related work, research continues to grow as new research is contributed. Software reliability growth models were examined in practice by Huang and Hung (2010), who highlighted the models' value in forecasting and enhancing software reliability. Putting software reliability growth models into practice in software development was covered by Almering et al. (2007), who highlighted the models' importance in improving software quality. Using IEEE standards, Schneidewind (2007) presented a quantitative method to software development that focuses on dependability of the software as a crucial indicator for evaluating quality.

Kanoun et al. (1991) introduced a technique for predicting and analyzing reliability of applications that is notably used to the TROPICO-R moving architecture, showcasing the practical implications of reliability models in real-world systems. Schneidewind and Keller (1992) demonstrated program, underscoring the importance of reliability assessment in critical systems. Carnes (1997) discussed software reliability in weapon systems, presenting case studies and insights into the challenges and strategies for ensuring reliability in defense applications.

These recent studies highlight the ongoing relevance and applicability of software reliability models in various domains, emphasizing their role in predicting, assessing, and enhancing software quality and reliability in different contexts. The research continues to advance, exploring new methodologies, applications, significant difficulties in the field of simulation software dependability.

3 Description of different SRGMs

Zhang et al. and Jeske et al. state that SRGMs have been extensively employed in a number of industries, including telecommunications, to study and estimate the measures connected to programs services' dependability. A few parameters must be understood before delving explore every detail of various prototype. Assume {X(t),t≥0} correspond to a counting process that shows the total number of flaws found up until time t. One may express an SRGM that originated on the mean value function of the following formula: NHPP (MVF), m(t) can be implemented as: P{X(t)=n} = $e^{-[m(t)]} * \frac{[m(t)]\hat{}n}{n!}$ where n = 0, 1, 2, 3, etc., and m(t) is that anticipated total quantity of errors found by timestamp t. Concerning the test period t, given MVF m(t) is not reducing in the confined situation m(∞) = a, where an is the anticipated The entirety of errors to be subsequently found. Knowing its worth can support ourselves decide how much further testing is needed and

whether the program is ready to be distributed to the public. Additionally, it can give an estimate of how many failures the clients will eventually experience. Generally, by utilizing various non-decreasing mean value functions, we can obtain distinctive NHPP models. At testing time t, the process that measures failure intensity is (t)=$\frac{dm(t)}{dt}$ = m'(t).

The likelihood that there will be no software problems, or the software's dependability (s,s+t) Considering that the most recent failure happened during testing(s≥0,t>0), is R(t|s) =exp,[−(m(t+s)−m(t))]

The equation since throughout testing t, the rate of fault detection per fault is d(t) = $\frac{m'(t)}{a-m(t)} = \frac{\lambda(t)}{a-m(t)}$

There are numerous Popular NHPP models that are currently being used with different MVFs are listed below.

3.1 Jelinski-Moranda de-eutrophication model:

The de-eutrophication prototype, developed in 1972 by Jelinski and Moranda, was Among the earliest hypotheses on the evolution of software reliability. The testing procedure is based on the assumption that there are N different, independent initial software defects. Debugging ensures that faulty errors are corrected and that no new faults are introduced. Consequently, there is a direct correlation between The Risk Measure and the rate of program collapse, which is the total amount of mistakes in the programme at any one time. As a result, the Hazard Function for time tn, or the interval between the nth Failure and (n-1) can be defined as follows:

$Z(t_n) = \phi[N-(n-1)]$,

Where, ϕ= Constant Proportional treatment and,

N= Software bugs when testing first begins.

The following illustrates a typical Jelinski-Moranda de-eutrophication model plot:

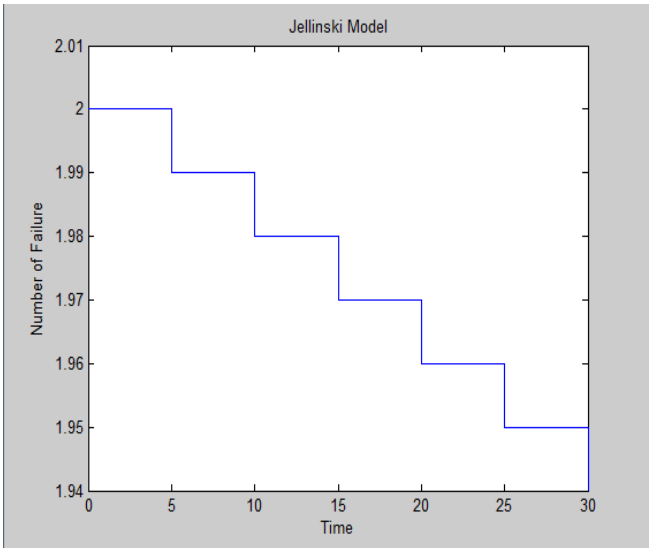

Fig3.1.1: Model of de-eutrophication Jelinski-Moranda

3.2 Goel-Okumoto model:

Goel and Okumoto introduced the model in 1979.Goel and Okumoto the Jelinski-Moranda Framework was expanded to produce an incomplete debugging framework. Every software program's entirety of errors at any given moment is described in this case by the Markov Process n, or X(n). That amount of time between a defect X(n) change is dependent on the faults that are now present in the system. Okumoto model throughout that time t_n which is the time during (n-1) and nth A Failure is represented by:

$$Z(t_n)=[N-p(n-1)]\lambda$$

Here, N= Early Software System Flaws discovered, P=Probability of imperfect Debugging,

λ= rate of failure for each defect.

A standard Goel-Okumoto Model plot looks like this:

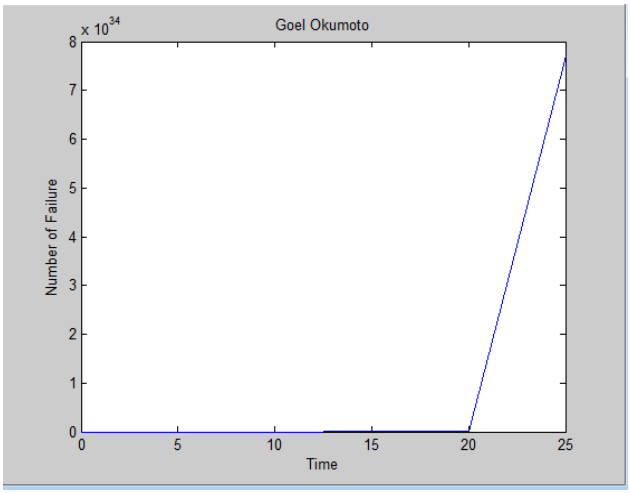

Fig 3.2.2Goel–Okumoto model

3.3 Littlewood software reliability model:

Littlewood and Verrall thought of an alternative method for creating the model that calculates the average amount of time between failures. They argued that program flaws in the software are not a direct indicator of software dependability. According to this approach, the Hazard Function based on the difficulty of the assignment and the level of programming expertise of the coder.
The Littlewood Software Reliability model plot looks like this:

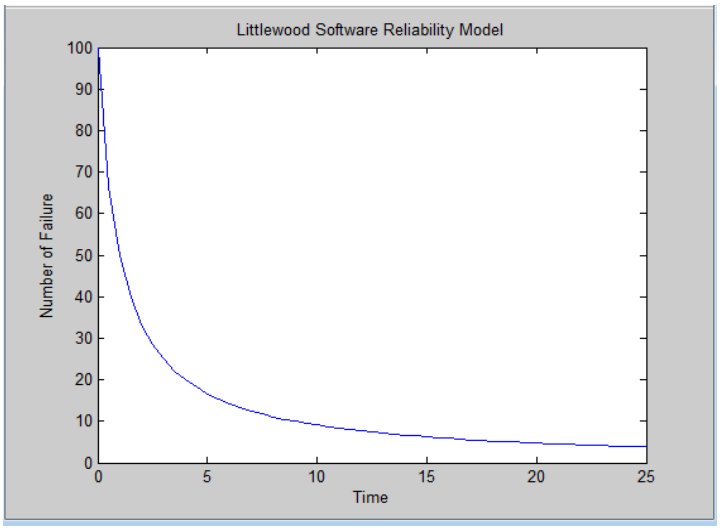

Fig3.3.3: Littlewood Software Reliability model

4 The Significance of Software Reliability

4.1 Maintaining regular system functioning requires software reliability.

The impact of software is growing in significance as more and more digital devices are being used. Millions of lines of source code make up the airborne software in the aerospace area. But the fast rise in software's scope and complexity also contributes to a rise in the number of failures. According to one study, professional software developers would write codes with six errors for every ten thousand lines.

4.2 The improvement of system dependability is hindered by software reliability.

Systems are using more and more software. For instance, the capabilities realized by software increased with each subsequent generation of fighter aircraft. System and software reliability are directly correlated. Software, in contrast to hardware, cannot restore a system by replacing or repairing its parts; instead, it must redesign it. Redundancy alone cannot ensure software reliability, and approaches to test software reliability differ from those used for hardware, which has a fully functional theoretical system. Software dependability is generally harder to ensure than hardware reliability. The dependability of NASA's software system is much inferior than that of hardware. As a result, the dependability of the system as a whole is greatly impacted by the software. Consideration must be given careful consideration to software reliability in order to increase system reliability.

5 Conclusion

A successful method for managing, forecasting, and evaluating software dependability is the use of reliability models. Software and hardware reliability issues have been successfully analyzed using a broad spectrum of complex stochastic process modelling in reliability engineering, also referred to as NHPP models. Due to their special capacity to recognize failure processes that display distinguishing patterns, such as reliability rise and degradation, NHPP models are easily used to software reliability evaluation.

To give a measurement for software dependability, Initially, we investigated the original Markov process-based framework (J-M).in this study. NHPP and Bayesian models were then created by integrating these models. We described the NHPP-based modelling procedure for both discrete and continuous temporal models. As an alternative, these models can be categorized as concave or S-shaped dependent on how their asymptotic representation looks. A few popular expansions of the NHPP model were looked at. Next, we looked at the flexible modelling approach, which allows for customization of the model based on requirements. subsequently, we covered time-dependent transition probability models and up-to-date NHPP models that reflect calendar time as well as implementation. The current NHHP SRGMs have benefits. Increased testing can help to remove errors. and effectively assigning and managing the testing resources.

Developers often have to balance the benefits of early availability against the disadvantages of postponing release in order to enhance usability or quality when determining when to release software. Real-world questions like how much testing is necessary, how to effectively and efficiently handle testing resources, when to release a product, what the market window is, what end users' and customers' expectations are, etc. are challenges that real-world software engineers have to answer. Before a product is released, a variety of stakeholders are usually involved in the decision-making process, and these stakeholders might not have the same preferences for the decision's outcome. The only requirement that determines the decision's implementation criteria is the congruence between the expected reliability level and the actual outcome is deemed successful. Software practitioners can determine when a software system is ready for release and whether the reliability of a product has reached a certain threshold with the use of NHHP SRGMs.

References

1. Li, S., Dohi, T. and Okamura, H. (2022) 'Are infinite-failure NHPP-based software reliability models useful?', *Software*, 2(1), pp. 1–18. doi:10.3390/software2010001.
2. Lin, J.-S., Huang, C.-Y. and Fang, C.-C. (2022) 'Analysis and assessment of software reliability modeling with preemptive priority queueing policy', *Journal of Systems and Software*, 187, p. 111249. doi:10.1016/j.jss.2022.111249.
3. Hanagal, D.D. and Bhalerao, N.N. (2021) 'NHPP Software Reliability Growth Models', *Infosys Science Foundation Series*, pp. 27–37. doi:10.1007/978-981-16-0025-8_3.
4. Risks in software testing techniques a literature review' (2018) *International Journal of Recent Trends in Engineering and Research*, 4(4), pp. 379–384. doi:10.23883/ijrter.2018.4250.0q4ai.
5. Selen, J. and Fralix, B. (2017) 'Time-dependent analysis of an M / m / C preemptive priority system with two priority classes', *Queueing Systems*, 87(3–4), pp. 379–415. doi:10.1007/s11134-017-9541-2.
6. '54 robotic software development using standard parts' (2017) *Software Methodologies A Quantitative Guide*, pp. 419–438. doi:10.1201/9781315314488-55.
7. 'Software complex for analysis of Bayesian models in queueing and reliability theories' (2015) *Systems and Means of Informatics* [Preprint]. doi:10.14357/08696527150316
8. Lai, R. and Garg, M. (2012a) 'A detailed study of NHPP Software Reliability Models (invited paper)', *Journal of Software*, 7(6). doi:10.4304/jsw.7.6.1296-1306.
9. Lai, R. and Garg, M. (2012b) 'A detailed study of NHPP Software Reliability Models (invited paper)', *Journal of Software*, 7(6). doi:10.4304/jsw.7.6.1296-1306.
10. Huang, C.-Y. and Hung, T.-Y. (2010) 'Software reliability analysis and assessment using queueing models with multiple change-points', *Computers & Mathematics with Applications*, 60(7), pp. 2015–2030. doi:10.1016/j.camwa.2010.07.039.
11. Huang, C.-Y., Hung, T.-Y. and Hsu, C.-J. (2009) 'Software reliability prediction and analysis using queueing models with multiple change-points', *2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement* [Preprint]. doi:10.1109/ssiri.2009.11.
12. Richard Lai*, Mohit Garg Department of Computer Science and Computer Engineering, La Trobe University, Victoria, Australia 2012
13. Lung, C.H.; Zhang, X.; Rajeswaran, P. Improving software performance and reliability in a distributed and concurrent environment with an architecture-based self-adaptive framework. J. Syst. Softw. 2016, 121,311
14. Wang, S; Wu, Y.; Lu M.; Li, H. Discrete nonhomogeneous Poisson process software reliability growth models based on test coverage. Qual. Reliab. Eng. Int. 2013, 29, 103–112
15. Yang, Q.; Zhang, N.; Hong, Y. Reliability analysis of repairable systems with dependent component failures under partially perfect repair. IEEE Trans. Reliab. 2013, 62, 490–498.

16. Amiripour, F.; Khaledi, B.E.; Shaked, M. Stochastic orderings of convolution residuals. Metrika 2013, 76, 559–576.
17. Arnold, B.C.; Villasenor, J.A. Exponential characterizations motivated by the structure of order statistics in samples of size two. Stat. Probab. Lett. 2013, 83, 596–601
18. Kavi, K.M. and Bhat, U.N. 'Reliability Analysis of computer systems using Dataflow Graph Models', *IEEE Transactions on Reliability*, 35(5), pp. 529–531. doi:10.1109/tr.1986.4335538.
19. Bhat, U.N. and Kavi, K.M. 'Reliability models for computer systems: An overview including dataflow graphs', *Sadhana*, 11(1–2), pp. 167–186. doi:10.1007/bf02811317.