ISSN 2278-3091
Volume 8, No.6, November – December 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse14862019.pdf
https://doi.org/10.30534/ijatcse/2019/14862019

Software reliability prediction using Knowledge Engineering approach

Mayuri H.Molawade[1], Dr. Shashank D. Joshi[2]
[1]Research Scholar, India, mhmolawade@bvducoep.edu.in
[2]Professor, India, sdj@live.in

ABSTRACT

Unwavering quality is one of significant quality traits of the product in which programming end client is more intrigued instead of the product engineer. In past some of good studies are carried out. But still some limitations are there as : Models have many shortcomings related to their unrealistic assumptions, environment-dependent applicability, and questionable predictability. In customary framework Predicting software abnormalities (like software maturing) brought about by asset depletion isn't a simple errand.. It is difficult to know from the earlier the parameters Involved with the software maturing. In this manner the abilities of Machine Learning (ML) calculations and the statistical methods can be analyzed to foresee the framework crash because of software maturing brought about by asset fatigue.A software system failure is unsurprising and expansive consideration is given to software system failure prediction not on progress. In this paper we are going to understand limitation of given techniques from various papers and identify that scope of improvement.

Key words: Machine Learning, parametric, non-parametric, reliability prediction

1. INTRODUCTION

Society is getting perpetually dependent on software and software controlled frameworks. A portion of this product is security basic, e.g., the product used to control vehicles, planes and other fast transport. Deformities in security basic software can prompt genuine damage or passing. An a lot bigger volume of software is business-basic, e.g., software that runs in cell phones, powers web servers and oversees server farms. Imperfections in this kind of software can prompt noteworthy money related misfortunes. Supporting these territories is frameworks software: the low-level working frameworks, compilers, gadget drivers and systems administration software on which complex frameworks are fabricated. This basic job implies that the dependability of frameworks software is of essential significance.

Software reliability quality models portray the failure lead of the product. The models are used to survey the product quantitatively. They assess the constancy of the product by predicting imperfections or frustrations for a product.

Reliability quality is one of noteworthy quality properties of the product in which Software end customer is more captivated rather than the product fashioner.

Thus, the execution of a software can be improved by joining significant quality characteristics like reliability, maintainability and availability of the software alongside execution properties like reaction time and throughput. The solid relationship that exists between quality traits and execution characteristics. With certain representations featuring the need of inside and out comprehension of the connection that exists among reliability quality and execution of the product. The determined information helps in improving the exhibition of the product economically over some undefined time frame and deal with the product all the more viably

Rigorous manual testing, code reviews and adherence to standards are essential to the success of large software projects, but they all suffer from two common problems:

1. **They depend fundamentally on human reasoning and judgment.** Humans are clever, but software can be devilishly complex. It is easy for subtle defects to creep into a project despite adherence to coding standards, and to evade manual testing and code review.

2. **They do not provide guarantees.** A test suite can demonstrate that certain executions of a software system do not exhibit defects, but provides no further guarantee. For safety- (and often business-) critical systems this may not be enough: it is highly desirable to have a *guarantee* of defect freedom; ideally an absolute guarantee, but at least a guarantee that system executions have been systematically checked up to some well-defined bound.

to accomplishing solid software systems, and they can likewise be viewed as four shortcoming lifecycle procedures:
1. Fault prevention: to maintain a strategic distance from, by development, flaw event
2. Fault removal: to recognize, by confirmation and approval, the presence of shortcomings and dispense with them. Fault tolerance: to provide, by redundancy, service complying with the specification in spite of faults having occurred or occurring.
3. Fault/failure forecasting: to estimate, by evaluation, the presence of faults and the occurrences and consequences of failures. This has been the main focus of software reliability modeling.

There are three main reliability modeling approaches:

1. The error seeding and tagging approach
2. The data domain approach
3. The time domain approach

1.2 Machine Learning Techniques and Big data

Techniques for computer performance analysis of machine l earning (ML) on big data. Use of ML techniques to forecast the desire for software reliability and survey them based on t he execution criteria selected. The most popular techniques of ML, including neuro fuzzy inference system (ANFIS), feed forward back propagation neural network, linear regression

2. A SURVEY OF WORK DONE IN THE RESEARCH (LITERATURE SURVEY) AREA AND THE NEED FOR MORE RESEARCH.

This section introduces the problems faced in software reliability. Different methodologies can be utilized to improve the dependability of software, be that as it may, it is difficult to adjust advancement time and spending plan with software reliability. Some of techniques as follow:

1. AnkurChoudhary et al. Dependable virtual products are the need of current advanced period. Disappointment nonlinearity makes software unwavering quality a confounded errand. Over past decades, numerous analysts have contributed numerous parametric/non parametric software dependability development models and talked about their suppositions, appropriateness and consistency. It reasoned that customary parametric software reliability models have numerous inadequacies identified with their unrealistic assumptions, condition subordinate pertinence, and flawed predictability. In difference to parametric software unwavering quality development models, the non-parametric software reliability development models which use AI procedures or time arrangement demonstrating have been proposed by analysts. This paper assesses and thinks about the precision of 2parametric and 2 non parametric software unwavering quality development models on 3 genuine informational collections for software disappointments.

2. Sultan H. Aljahdali et al. Software dependability models are important to evaluate the probability of the item flop along the time. A couple of particular models have been proposed to foresee the item software unwavering quality development (SRGM); regardless, none of them has exhibited to perform well contemplating different endeavor traits. The ability to anticipate the amount of inadequacies in the item in the midst of progression and testing structures. In this paper, we examine Genetic Algorithms (GA) as an elective method to manage derive these models. GA is a noteworthy AI framework and progression systems to assess the parameters of definitely saw reliably improvement models. What's more, AI calculations,, proposed the plan rout the vulnerabilities in the showing by combining various models using different objective ability to achieve the best theory execution where. The destinations are clashing and no structure exists which can be viewed as best as for all goals.
Tests have been organized in this paper to validate these hyp otheses. It was then overcome by determining the judicious a bility of the design outfit advanced using multi-target GA. E ventually, the tests and regular versions are distinguished.

3. ArunimaJaiswal et al Software Reliability is a key part of the reliability of the software and is one of the most necessary viewpoints for assessing the complexity of a software product. In rising substantially trustworthy software, the software industry bears various problems. Use AI (ML) methodologies for unfolding software quality desire has been shown and impressive results have been achieved. In this paper, they suggest the use of ML techniques to unfalter software quality conjecture and test them based on the execution parameters they have selected. They have associated ML methodology including versatile neuro-fluffy surmising framework (ANFIS), feed-back spread neural system, general neural recurrence system, bolstering vector machines, multi-layer observation, bagging, falling back neural proliferation system, case-based learning, straight recurrence, M5P, decreased blunder pruning tree, M5Rules to predict the determined quality of the item. Taking into account the tests conductedIt was seen that ANFIS yields better results in all cases and can therefore be used to anticipate unwavering quality software as it predicts reliability even more clearly and precisely when it is diverged from all other frameworks recently referenced. They also made a comparative evaluation of full frustration in this study.

4. RamakantaMohanty et al. In this paper, the creators utilized AI procedures, explicitly, propagation trained neural network (BPNN), Group method of data handling (GMDH), Counter propagation neural network (CPNN), Dynamic evolving neuro–fuzzy inference system (DENFIS), Genetic Software (GP), Tree Net, statistical multiple linear regression (MLR), and multivariate adaptive regression splines (MARS),, to precisely estimate software unwavering quality. Their viability is shown on three datasets taken from writing, where execution is looked at as far normalized root mean square error (NRMSE) acquired in the test set. From thorough analyses directed, it was seen that GP outflanked all strategies in all datasets, with GMDH coming a nearby second.

5. G.Krishna Mohan et al. Software unwavering quality models get to the dependability by shortcoming forecast. Unwavering quality is a certifiable marvel with many related ongoing issues and to get answers for issues rapidly, precisely and acceptably a huge no. of delicate processing systems has been created. They endeavor to address the product disappointment issues by demonstrating software disappointment information utilizing the AI methods, for example, bolster vector machine (SVM) relapse and summed up added substance models. The investigation of software dependability can be ordered into three sections: displaying, estimation, improvement. Software enduring quality

exhibiting has created to a point that significant results can be obtained by applying proper models to the issue; there is no single model comprehensive to all of the conditions. They propose distinctive AI strategies for the assessment of software steadfast quality, for instance, fake neural systems, bolster vector machine count drew nearer. They by then separate the results from machine getting the hang of illustrating, and balance them with that of some summarized direct showing systems that are corresponding to software trustworthiness models.

6. D. HemaLatha et al. Software dependability expectation is exceptionally testing in support stage just as in the beginning periods of software advancement. In the previous scarcely any years numerous product dependability models have been proposed for surveying unwavering quality of software yet creating precise dependability forecast models is troublesome because of the intermittent or successive changes in information in the space of software designing. Subsequently, the product unwavering quality forecast models based on one dataset show a noteworthy diminishing in their precision when they are utilized with new information. The target of this paper is to present another methodology that advances the precision of software unwavering quality prescient models when utilized with crude information. Subterranean insect Colony Optimization Technique (ACOT) is proposed to anticipate software dependability dependent on information gathered from writing. An insect state framework by joining Traveling Sales Problem (TSP) calculation has been utilized, which has been changed by actualizing various calculations and additional usefulness, trying to accomplish better software unwavering quality outcomes with new information for change situated frameworks. The scholarly conduct of the subterranean insect settlement system by methods for a state of coordinating counterfeit ants are bringing about promising outcomes. The strategy is approved with genuine dataset utilizing Mean Time to Failure (MTTF) and Mean Time Between Failure (MTBF).

1. Simulation Results

Reliability is calculated as:

Reliability = MTBF / (1+MTBF)

MTBF : Mean Time Between Failure

$$MTTF = \int_0^\infty t f(t)dt = \int_0^\infty R(t)dt$$

MTTF : Mean Time to Failure

To calculate MTTF in MS-Excel the following formula is used:

MTTF = f(x+dx)-f(x)/dx

Or

If the reliability is checked on hourly basis the following formula can be used:

MTTF = 1/ failure rate

Mean Time to Failure

Measures time between observable system failures

For example, assume you tested 3 identical systems starting from time 0 until all of them failed. The first system failed at 10 hours, the second failed at 12 hours and the third failed at 13 hours. The MTTF is the average of the three failure times, which is 11.667 hours. If these three failures are random samples from a population and the failure times of this population follow a distribution with a probability density function (*pdf*) of $f(t)$, then the population MTTF can be mathematically calculated by:

Mean Time between Failures

The points on the plot are the observed cumulative MTBFs. These values are calculated by the following equation:

$$MTBF(t) = \frac{t}{N(t)}$$

where:

- *t* is the cumulative operating time.
- *N(t)* is the observed number of failures by time *t*.

Or

MTBF = 1/MTTF

7. V. Sangeetha et al Software unwavering quality is a likelihood of software framework to work under working conditions over timeframe. Software unwavering quality focuses on forecast of leftover issues, estimation of disappointment force, dependability and cost. Software issues causes because of the product disappointments created through the formative procedure of software. Software framework disappointments have huge downsides on schedule and assets for recuperation. A product framework disappointment is unsurprising and enormous consideration is given to software framework disappointment expectation. Be that as it may, the likelihood of disappointment estimation stayed unsolved. The exploration work is done to play out the quick expectation of software disappointments through Genetic-based Bayesian model for improving the genuine positive rate.

8. Yoshinobu Tamura et al. Previously, numerous product dependability models have been proposed by a few analysts. Likewise, a few model choice criteria, for example, Akaike's data foundation, mean square mistakes, anticipated relative blunder, etc, have been utilized for the choice of ideal software unwavering quality models. These appraisal criteria can be helpful for the product supervisors to evaluate the past pattern of shortcoming information. Be that as it may, it is critical to survey the expectation precision of model after the finish of issue information perception in the real software task. In this paper, they propose a strategy for ideal software dependability model determination dependent on the profound learning. Besides, they show a few numerical instances of software unwavering quality appraisal in the genuine software ventures. Specifically, they discuss the ideal discharge time and absolute expected software cost as far as the model choice dependent on the profound learning.

9. ManjubalaBisi et al. This paper proposes an unwavering quality model based neural system software to anticipate the total number of disappointments dependent on Feed Forward design. Contingent on the tally information of accessible software disappointment, the execution time is encoded using Exponential and Logarithmic capacity to give the encoded an

incentive as a contribution to neural system. The effect of encoding and the impact on expectation accuracy of different encoding parameters are considered. The effect of the neural system design to the degree of concealed hubs was also investigated. Eighteen software disappointment information indexes were tried to present the proposed methodology. Numerical results show that through various software ventures, the proposed methodology provides worthy results cross-sectionally. The approach demonstration was contrasted with some observable models and empirical models taking into account three datasets of change point. The results of the comparison indicate that the model proposed has a good predictive potential.

10. Fu Yangzhen et al. With the advancement of software dependability research and AI, many AI models have been utilized in software unwavering quality expectation. A long transient memory arrange (LSTM) displaying approach for software dependability forecast is proposed. Benefit from its specific information stream control structure, the model beats the disappearing and detonating affectability of basic recursive neural system for software dependability forecast. Proposed approach additionally joins with layer standardization and truncate back proliferation. Somewhat, these two strategies advance the impact of the proposed model. Contrasted and the straightforward recursive neural system, numerical outcomes show that our proposed methodology has a superior presentation and power as for software unwavering quality expectation.

3. LIMITATION AND SCOPE IS IDENTIFIED

There are several problems and limitations in the literature a nalysis as discussed below are described in the prediction of computer reliability.

It is well known in the traditional system that software agein g happens due to software error rather than hardware faults. Several studies [1], [2], [3] have reported that one of the causes of unplanned software outages is the software aging phenomenon. From this idea, [2] produce a model for software changes based on historical databases. Recently, this type of predictors, which is known as just-in-time model, has also appeared in several studies [2], [4].

In traditional system It concluded that traditional parametric software reliability.

Models have many short comings related to their unrealistic assumptions, environment-dependent applicability, and questionable predictability.

Limitation of this paper is Software reliability prediction is very challenging in maintenance phase as well as in the starting phases of software development.

In the previous scarcely any years numerous product dependability models have been proposed for surveying unwavering quality of software yet creating exact unwavering quality expectation models is troublesome because of the intermittent or successive changes in information in the space of software engineering[13]

3.1 Scope

Reliability of technology is an important part of the performance of software. Software reliability prediction is the prospect of a computer program's failure-free operation in a specified environment for a specified time period. So for software development and testing industry, software reliability research is considered useful.

4. CONCLUSION

Software Reliability is a significant piece of software quality. Software unwavering quality forecast is the possibility of the disappointment free activity of a PC program for a predetermined timeframe in a predefined domain . Along these lines, software unwavering quality research is viewed as helpful for software advancement and testing industry.

REFERENCES

[1] Hossein A khavan "**Power systems big data analytics: An assessment of paradigm shift barriers and prospects**" Energy Reports 4 (2018) 91–100 30 November 2017 https://doi.org/10.1016/j.egyr.2017.11.002

[2] Javier Alonso "**Predicting Software Anomalies using Machine Learning Techniques**"

[3] Ankur Choudhary "**Software Reliability Prediction Modeling: A Comparison of Parametric and Non-Parametric Modeling**"

[4] Ashima Gupta "**Prediction Of Software Anomalies Using Time Series Analysis – A Recent Study**" International Journal of Advances In Computer Science and Cloud Computing, ISSN: 2321-4058 Volume- 1, Issue- 1, May-2013

[5] Sultan H. Aljahdali "**Software Reliability Prediction Using Multi-Objective Genetic Algorithm**"

[6] Soumya Joseph "**Software Defect Prediction Using Enhanced Machine Learning Technique**" International Journal of Innovative Research in Computer and Communication Engineering (*An* ISO 3297: 2007 Certified Organization)Vol. 4, Issue 6, June 2016

[7] Philip Gross "**Predicting Electricity Distribution Feeder Failures Using Machine Learning Susceptibility Analysis**"

[8] ArunimaJaiswal "**Software reliability prediction using machine learning techniques**" Int J SystAssurEngManag (February 2018) 9(1):230–244 https://doi.org/10.1007/s13198-016-0543y

[9] RamakantaMohanty "**Application of Machine learning techniques to Predict software reliability**" 70 International Journal of Applied Evolutionary Computation, 1(3), 70-86, July-September 2010 https://doi.org/10.4018/jaec.2010070104

[10] Rita G. Al gargoor "Software Reliability Prediction Using Artificial Techniques**"** IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013

ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784 www.IJCSI.org.

[11]G.Krishna Mohan "Assessment and Analysis of Software Reliability Using Machine Learning Techniques" International Journal of Engineering & Technology, 7 (2.32) (2018) 201-205 International Journal of Engineering & Technology Website: www.sciencepubco.com/index.php/IJET
https://doi.org/10.14419/ijet.v7i2.32.15567

[12] G. Martínez-Arellano "In-process Tool Wear Prediction System Based on Machine Learning Techniques and Force Analysis" Peer-review under responsibility of the International Scientific Committee of the 8th CIRP Conference on High Performance Cutting (HPC 2018)..

[13] D. HemaLatha"A Development Model for Predicting Software Reliability Using Ant Colony Optimization Technique for Change Oriented Software Process" IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 2, Ver. I (Mar.-Apr. 2017), PP 57-64 www.iosrjournals.org
https://doi.org/10.9790/0661-1902015764

[14]VipulVashisht "A Framework for Software Defect Prediction Using Neural Networks" Journal of Software Engineering and Applications, 2015, 8, 384-394Published Online August 2015 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2015.88038

[15] V**. Sangeetha"**A Survival Study on Software Failure Prediction Using Adaptive Dimensional Gene Model**"** International Journal of Computational Intelligence and Informatics, Vol. 6: No. 4, March 2017

[16]Guru Prasad PANDIAN "A critique of reliability prediction techniques for avionics applications**"** Chinese Journal of Aeronautics Received Date: 20 February 2017
https://doi.org/10.1016/j.cja.2017.11.004

[17]Awni Hammouri"Software Bug Prediction using Machine Learning Approach" *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 2, 2018*
https://doi.org/10.14569/IJACSA.2018.090212

[18]TeeratPitakrata "Hora: Architecture-aware Online Failure Prediction" *The Journal of Systems & Software* 24 February 2017

[19] Ajay Chandramouly "Reducing Client Incidents through Big Data Predictive Analytics"IT@Intel White Paper Intel IT IT Best Practices Big Data Predictive Analytics December 2013

[20]Yoshinobu Tamura "Software Reliability Model Selection Based on Deep Learning with Application to the Optimal Release Problem"Journal of Industrial Engineering and Management Science, Vol. 1, 43–58.doi: 10.13052/jiems2446-1822.2016.003_c 2016 River Publishers. All rights reserved.
https://doi.org/10.1109/ICIMSA.2016.7504034

[21]LanGuo "Robust Prediction of Fault-Proneness by Random Forests"

[22]Manjubala Bisi "Software Reliability Prediction using Neural Network with Encoded Input"International Journal of Computer Applications (0975 – 8887) Volume 47– No.22, June 2012
https://doi.org/10.5120/7492-0586

[23] Christian Ruiz "Improving Data Validation Using Machine Learning" United Nations Economic Commission For Europe Conference Of European Statisticians18-20 September 2018)

[24]Jyoti Devi "A Review of Improving Software Quality using Machine Learning Algorithms" International Journal of Computer Science and Mobile Computing A Monthly Journal of Computer Science and Information Technology ISSN 2320–088X
IMPACT FACTOR: 6.017 IJCSMC, Vol. 6, Issue. 3, March 2017, pg.148 – 153

[25]Jinyong Wang "Software Reliability Prediction Using a Deep Learning Model based on the RNN Encoder–Decoder"Reliability Engineering and System Safety 21 October 2017
https://doi.org/10.1016/j.ress.2017.10.019

[26] Fu Yangzhen "A Software Reliability Prediction Model Using Improved Long Short Term Memory Network" 2017 IEEE International Conference on Software Quality, Reliability and Security (Companion Volume) 2017.
https://doi.org/10.1109/QRS-C.2017.115

[27] Rohini B. Jadhav , Shashank D. Joshi , Umesh G. Thorat Aditi S. Joshi "A Software Defect Learning and Analysis, Utilizing Regression Method for Quality Software Development" ,Volume 8, No.4, July – August 2019.
https://doi.org/10.30534/ijatcse/2019/38842019

[28] Marcel Bonar Kristanda , Seng Hansun "MOODLE LMS Resources Prediction: Exponential Moving Average Approach", Volume 8, No.4, July – August 2019.
https://doi.org/10.30534/ijatcse/2019/43842019