JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30th April 2018, Volume 3, Issue 1, JSEIS, CADMEI Copyright © 2016-2018
www.jseis.org

A review of software fault detection and correction process, models and techniques

[1]Sabia Sulman, [2]Bushra Nisar
[1,2]International Islamic University, Islamabad, Pakistan
Email: [1]sabiasulman64@yahoo.com, [2]bushra.rajput2011@gmail.com

ABSTRACT

In software development life cycle, the most important activity is software maintenance, in order to get a reliable and quality product. Huge amount of time, cost and effort is involved in it. Maintenance of software encompasses various activities like prediction, detection, prevention and correction of fault. Due to refined and multifaceted applications, clustered architecture, artificial intelligence and commercial hardware are in use. Hence, in this research work a review is conducted in the field of software fault detection and correction. There are a lot of software reliability growth models and techniques which help in software fault detection and correction, nevertheless, the room for more models and processes is vacant to detect and correct faults.

Keywords: software maintenance; reliability; software fault detection; software fault prevention; reliability models;

1. INTRODUCTION

An error, bug or fault in computer program is a software defect. Quality of the software is decreased if a system produces an incorrect, unwanted and unexpected result, which occurs due to the software faults. Software maintenance is of paramount importance as it involves a great quantity of efforts and cost. For the development of high assurance systems, reliability and performance requirements are essential. Software testing is the way toward practicing a program with the specific expectation of finding faults preceding delivery to the clients. At times, fault correction is not performed promptly once a failure is recognized [1]. Schneidewind [2] announced that the software developers may delay fault correction in situations where the failures were classified as non-crucial on the current release, and were not viewed as crucial for one to three releases later on.

The software is a solo entity which has recognized a strong influence on all the domain software. For their accurate and reliable service need, the domain activities always demand for quality software [3-5]. Software quality means a faultless product, which will be capable of producing anticipated results within the limitations of cost and time. Furthermore, fault detection and testing the system has become quite important processes in software life cycle. In order to detect faults and correct them, numerous fault prediction techniques, fault detection and correction processes, and reliability growth models are proposed and analyzed [6]. The contribution of this paper is to discuss and present the work done to detect and correct software faults.

2. RELATED WORK

To gauge and to equate testing scope approach, [7] authors suggested transformation analysis. In order to check the viability of test suits, a statistical investigation was made to check the observational outcomes. This was hard to discover the genuine faults in software and approve the basis that is the reason countless faults were infused in the software to think about the test suits on the product (software). The mutants were created to demonstrate the fault discovery successfully. The outcomes were extremely predictable over the examined criteria as they have demonstrated that the utilization of transformation administrators was reliable outcome. The advantage of the mutant era was that the mutant administrators portrayed in precise way and the fault infusion process executed effectively to recognize flaws. An extensive number of mutants diminished the effect of arbitrary variety in the analysis and permitted the utilization of various examination approaches. This approach concentrated on the decision of low cost and also viability of test suits.

In [8] authors proposed the mechanized static analysis to recognize faults in software. The quantity of procedures as software audit, software review and testing were utilized to identify faults, however, every strategy was not compelling to distinguish mistakes. To dispose of faults and errors before the delivery of software, the static analysis tools were utilized. The static examination was a modest approach and an inexpensive approach. There was no need of execution because the faults were noticed physically and consequently in Nortel programming items. Automated static analysis utilized Orthogonal Defect Classification Proposition to recognize task and checking flaws. The programming errors which brought on security susceptibilities were additionally identified via automated static investigation strategies. The outcomes show that to improve the quality of software

in financial way, the computerized static examination was an effective method. Be that as it may, there were a few restrictions additionally as the outcomes just gave by utilizing three bigger Nortel systems for C/C++.

In [9] researchers proposed a joined way to identify faults in in embedded control software framework. At two levels that is programming level and controlled-framework level, the viewed data was checked with the suitable basics. The inserted control framework contains both programming and equipment parts. A model-based procedure was utilized to identify faults amid run time operation. The proposed approach was utilized to look at the conduct of programming that the usefulness was performed in coveted way or not. When the examined conduct was dismissed by a software level screen, the software fault was distinguished. To segregate the faults in programming, the I/OEFA (input-output extended finite automata) were utilized. To discover and identify faults various existing procedures including N-version programming and Formal verification were utilized, however there were a few constraints like as intricacy and absence of finish determination and so forth. Along these lines, a two-tier method was utilized to discover all faults in software. In any case, the approach guarantied no false-alert.

In [10] researchers used automated search and suggested statistical testing to identify faults in software. Statistical and the basic testing were dignified to identify faults. The statistical testing was ideal to identify faults than the basic and irregular testing since it considered the approach of both procedures to minimize the issues. To identify faults in real world applications, this research depended on the robotized search techniques. The uniform irregular testing was more productive in vast test sizes than statistical testing and the dispersions have demonstrated the reduced fault discovery capability. To create test data in samples, the probability distribution was utilized. Be that as it may, it was extremely hard to derive probability distribution for huge and multifaceted software. Search Based Software Engineering (SBSE) was utilized to derive probability distribution in statistical testing which was actualized physically. It has demonstrated that the method was powerful and applicable for bigger input area/domain.

In [11] authors projected an immaculate model-based way to deal with challenges in the distributed frameworks. Automatic system monitoring and recuperation enhanced the trustworthiness viably in distributed programming frameworks yet it was exceptionally troublesome to discover the faults. The strategies considered are Bayesian estimation and Markov decision theory to give recuperation in a framework. This approach investigated more focal points by relationship of checking and recuperation techniques in examination of utilizing them independently. The faults were infused in practical web-based business frameworks to approve this approach. By utilizing automatic system monitoring and recuperation methods, the constancy of disseminated programming framework was enhanced. The recuperation was exceptionally troublesome on the grounds that it was hard to pick devices and strategies to discover the faults in distributed software system. The outcomes have demonstrated that the approach was of minimal cost and gave high accessibility solution for distributed framework.

2.1 Software fault prediction techniques

The expenses of finding and adjusting software deserts have been the costliest movement in software improvement. The exact forecast of defect-prone software modules can help the product testing exertion, decrease costs, and enhance the software testing process by concentrating on fault inclined module. The new methodology is proposed for software fault prediction, which is the combination of two techniques “bagging and swarm optimization”. From the result it is found that the proposed method is best for the improvement in software quality as compared to other fault prediction techniques [12].The software administration under various datasets with the criticality disclosure is played out and in addition keeps up the product framework execution under the fault prediction models and characterized the fault examination based model utilizing the metric examination and the fault based assessment. Author performed the dependability and additionally the execution estimation.

An anomaly identification approach is characterized, so that contrast metric investigation should be possible effectively [13]**.** In order to identify and recognize the system reliability and accessibility under different software features, an Artificial Immune system-based fault prediction program is characterized. The ongoing programming framework investigation, under the constancy appraisal by dealing with the software prerequisite, is defined. Author additionally play out the characterization of these modules under the fault criticality level and also in view of the fault recurrence. The individual module investigation is played out and also the entire framework examination under the information examination and in addition class level investigation is performed. The model introduced by the author was motivated from the Immune framework adapted in various applications. The fault prediction demonstrating at the class level, is characterized [14].

To find out the value of object-oriented matrices in foreseeing fault-prone classes, various design attributes are examined and concluded that the coupling and complexity are more responsible for the occurrence of fault in the classes [15]. Different approaches to reduce difficulties in different procedures related to software development are analyzed such as, the defects in software modules by using object-oriented approach to minimize testing efforts and time for software development, development time analysis for better reliability, extensive case analysis, attribute selection for better software classification and the prediction model to identify software reliability [16].

An examination on the design metrics alongside fault investigation for the object-oriented programming modules is considered. The product framework investigation under the machine learning calculation is described, so that the better outcome examination will be performed. In a research work, authors performed work, datasets and experiments demonstrated the powerful validation and seclusion of faults in these software systems [17]. A suit of configuration measures to foresee the software fault in object-oriented programs is exhibited. For playing out the fault discovery in an object-oriented software system, a probabilistic approach is defined. The principle target was to concentrate on the product item and the quality. The product examination under the recurrence investigation of the faults particular to the classes and the fault frequency is considered. The fault inclination in the product system is additionally concentrated. [18] A calculation examination based forecast model for object-oriented software is exhibited, so that the product quality will be recognized under the software fault investigation and programming cost investigation. For the quality characteristic forecast with different, a constructing model with clustering approaches is characterized. Author played out the investigation under the FCM and neural system based forecast display so that the successful characterization of the product measurements will be done. Likewise, the experimentation on the product framework under the quality investigation and the fault inclination is measured. Experimentation for the viability assessment is considered so that the more precise framework will be constructed. [19]

There are innumerable techniques proposed to predict software faults but none has proven to be perfect and complete. While the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is tested on the basis of java based object-oriented software system and results showed that this technique is useful for prediction of fault proneness. The proposed technique was tested at different numbers of matrices and was concluded that with the reduced number of matrices accuracy increases significantly which proves it to be the best technique for software fault prediction [20]. The quality vector analysis under the software fault investigation is characterized. In order to do more exact module investigation, the prediction model under the performance examination is considered. Author characterized the work for java based programming framework [21]. A fault prediction analysis-based review for the open source software framework is demonstrated. Author characterized the part for the software applications and also characterized a level program so that the software adequacy will be drawn. The code-based examination under the attributes investigation and the nature investigation is performed. The framework under the fault prediction abilities is characterized [22]. A fault forecast based review under the UML framework displayed, so that the fault analysis in the product framework will be made. The forecast demonstrate for the logistic regression is considered and also the UML measurements examination for the software codes are described. The code estimation under the direct scaling is additionally played out, so that the unit change will be evaluated completely and fault prediction will be done adequately [23].

2.2 Software fault detection and correction process

Many automatic tools and techniques analyzed that, these automatic tools are not enough to detect and correct architectural software design defects fully automatically so, still some manual effort is required in these tools. There is a need in future work to propose the fully automatic ASD defect detection tool and semi-automatic fault correction tools [24]. A novel modeling system for software FDP and FCP is produced and its parameter estimation with WLS technique is contemplated. The exploration depends on the speculation on GOS that the identification time and redress span of every fault around and autonomously takes after certain stochastic distribution, and the created FDP and FCP models have a general structure. Utilizing two datasets of which one is from distributed work and the other one is gathered amid the development of a practical software system. The use of the models represented that the proposed model with the WLS evaluation has brought down expectation mistakes. This is on account of the proposed demonstrate contains the fault correction data and additionally the detection data, and the WLS estimation weights more on the errors of recent information, which have more bits of knowledge with the future test process. There are a few constraints in ebb and flow research, for example, it sets the exponential order statistic as illustration, and this model neglects to fit the S-formed fault discovery dataset [25].

Exploration of fault detection mechanism, and additionally fault counteractive action mechanism in connection to the late pattern of the most recent advances have been talked about. Their fault discovery and software frameworks used to analyze the immeasurable number of strategies and methods, however not each tech suits each framework. Selection of, technology system arrangement, technology platform, size and intricacy of adaptableness and reliability targets, is driven by critical factors. Automated approach to distinguish an inclination to more elevated amounts in hybrid mining methods and statistical models are in inclining toward more traditional systems-oriented solutions for diagnostics and aversion. Fault handling in advanced applications are in the early phases of research and the solution architecture attempt to manufacture resistance level however much as could reasonably be expected [26].The software reliability modeling is analyzed and a Bayesian way to deal with gauge the obscure parameters for FDP with failure checks is displayed. A far reaching and efficient review on the Bayesian estimation for consolidated FDP and FCP is carried out. The techniques are created under the Bayesian system to produce a progression of arbitrary samples from the posterior distributions of the parameters. To show the proposed technique, a reproduction study is directed to look at the performance of the proposed Bayesian approach with that of MLE strategy. Simulation studies expose that the Bayesian technique performs exceptionally well in numerous settings without torment from the convergence issue. A numerical illustration is applied and demonstrated the proposed Bayesian approach can be connected to more broad circumstances contrasted with MLE technique. Additionally, the proposed Bayesian technique is more effective in computational speed than MLE strategy [27].

Precompiled Fault Detection (PFD) technique is proposed to overcome the problem of fault occurrence in runtime. This technique is used to detect and repair the fault before the compilation of source code. PFD technique is tested by using the simulators and it is concluded that by using this technique errors are removed appropriately before the execution of source code and the reliability of software increased [28]. A structure of Software Reliability Estimation in view of Fault Detection and Correction Processes is produced and parameter estimation issue in this circumstance is considered. The correct expulsion time interim of every fault is considered. To depict this kind of dataset, the fault expelling matrix is characterized. The proposed demonstrating structure is connected to a real three-edition test dataset. The outcomes demonstrated that the proposed process shows structure with Maximum Likelihood estimation delivered enhanced parameter measures and a dependable stochastic model. One test dataset with three discharges from a viable software project is connected with the proposed structure, which indicated suitable performance. Likewise, the proposed process shows system can give a decent estimation and expectation on the dataset with numerous discharges [29].

2.3 Software reliability growth Models

Software reliability is defined as the probability of errors occurred in software, considered as the measurement scale of software quality. Usually software growth models are used only for fault detection. This is not enough to detect the faults in the software; rather there is always a need to correct them for the reliability of software. Therefore, some beneficial techniques for the detection and correction of fault are discussed and experiment shows as a result that Maximum Likelihood Estimation (MLE) has better capability of fault prediction as compared to the Least Square Estimation (LSE). The testing time and testing coverage is important for reliability [30]. Previously one dimensional approach was used in software reliability growth model which focused either on testing time or testing coverage therefore multi released two dimensional approach based on growth model is introduced. In the present case testing effort and testing time are considered as inputs which have impact on output testing resources [31].

Research proposed process level redundancy (PLR) approach for the soft errors which have serious impact on software reliability. PLR is the software-based technique which focuses on the detection of errors which are in the boundary of sphere of replication (SOR). PLR is used to emerge multi-core process as well as influences the OS to uninhibitedly plan the procedures to all accessible resources. For the deployment of this technique, there is no need to modify the operating system as well as hardware [32]. Several software reliability growth models were introduced in the last 3 decades for the fault detection and correction by using statistical assumptions. Many of them assumed that fault detection and correction is done parallel but it is not true there is always a time break between fault detection and correction process. There is always a problem in resource allocation process of fault detection and correction. Mathematical optimization model is proposed for the resource allocation of fault detection and correction [33].

From the study of software reliability growth models, it is found that these models present either infinite or finite number of failure. One dimensional model is used to work either on testing time or testing effort which is not enough for accurate software reliability. To resolve this problem two dimensional model is preferred [34]. Queuing based models are best for achieving the reliability accuracy however these models are not taking in to

account the amount of resources that were consumed during the fault detection and fault correction process. Therefore, the new model integrates Exponentiated Weibull TEF and Logistic TEF techniques that are very useful for considering the amount of resources consumed during the testing process. This model is best for improvement of estimation and evaluation of software [35]. In past SRGM are based on the assumption of fault detection and removal process at the same time, however, in real it is not possible. From the observation it is concluded that removal of mutually dependent faults can only be possible if the leading faults are removed first. Along these lines, the thoughts of fault dependency and time-dependent delay work into software reliability growth modeling are incorporated [36].

Testing is very essential phase in every software development life cycle (SDLC). In testing software fault detection and correction is one of the important steps. There are various SDLCs developed from last three decades but majority of them were developed by using static approach. The main reason for which testing is important, is the consumption of resources in a very large quantity. To consume the resources efficiently a mathematical SRGM is introduced and to achieve this model Pontryagain's Maximum principle is used. Moreover, for the purpose of resource allocation Differential Evolution (DE) is used [37]. SRGM have their own approaches some are flexible according to the situation while other are not flexible as such. Therefore, the selection of appropriate process model is not an easy task. To overcome this problem many authors donated their efforts by using different approaches. It is observed that "Random lag function, Infinite server queuing theory, Hazard rate function" approaches are different in their assumption but mathematically these approaches are proved to be equal and also it is concluded that Hazard rate function is best for generalized purpose and provides a common platform for perfect and imperfect SRGM [38].

SRGMs were developed for the detection and correction of faults in the software however many of them not consider the resource expenses. Subsequently, in request to address this issue, a new model is introduced by joining the resource uses and change-point, which spends on software troubleshooting process. A genuine Software failure project exhibited the adequacy of proposed models, and numerical outcomes demonstrate new queuing model can give better fit and estimate [39]. To achieve the great demand of software reliability there is a need of SRGM which estimates the testing effort and resource utilization. Therefore, to overcome these problems, improved GO model is introduced by considering the two factors for the effectiveness of fault detection and these factors are human learning ability and fault detection rate by observing the remaining faults after the fault removal. This model is tested under the faulty data and proves that this model is accurate to fit as compared to other models [40]. This analysis is based on introduction and renewal of different approaches to reach a certain level of reliability and perfection in software development, for instance evolutionary algorithm is defined to design an effective application which would be fault free. Similarly resource utilization approach was defined and lot more [41].

Object-oriented metrics are introduced to separate faulty and non-faulty modules and offering the design by using programmed tools so that fault analysis in modules can be done effectively. There are number of strategies and methods used to recognize faults in programming framework however every strategy has a few constraints too. SRGM includes the stochastic procedure for statistical examination of software. How SRGM is utilized to ascertain the time delay and minimize the cost of programming items is examined? Utilizations of SRGM model in some basic applications, for example safety-critical flight software, online banking service and huge scale business software, is considered. The proposed work is to apply any SRGM model on the web applications to test and compute its consistency. Firstly, the web application is tested to check the presence or absence of faults, then the SRGM model is utilized to discover the rate of identification of the faults and to figure out the reliability of web applications [42].

A new model is introduced to compare the fault detection and correction processes in analysis of the number of faults. The maximum SRGM used for fault detection but the fault correction was ignored. Because, it was assumed that the faults were eliminated immediately and efficiently but it was not realistic. The fault correction was difficult part because it was difficult to correct all faults. When the faults were detected, the faults were located and removed by doing some changes in the codes by fault correction process. A logistic function was used to model both fault detection and fault correction processes based on NHHP. The dependency of the two processes was described by the ratio of corrected fault number to detected fault number in software [43]. A software reliability growth model incorporating the Burr type Xii testing technique is introduced, for detection and correction of faults in software which helps increasing reliability, an important factor for maintenance of quality of software. This model is statistically evaluated and compared with other models involving different techniques. These experimental results proved that present model is much better to detect and correct faults of software, which decreases failure rate of software. [44]

3. CONCLUSION

A lot of work is being done in the field of software maintenance. To minimize the testing efforts, various fault prediction, fault detection and fault correction techniques are explored. In the field of software maintenance, fault detection and fault correction, enough research work has been done and going on. In relation to the recent trend of advanced technologies, research on fault detection and fault correction mechanism is discussed in this survey paper. Fault handling in modern day applications are in the early stages of research and the solution architecture try to build tolerance level as much as possible. More models and techniques are needed for the improvement of the fault detection and correction activities and processes.

ACKNOWLEDGEMENT

Special thanks to International Islamic University, Islamabad, Pakistan in supporting this research.

REFERENCES

1. Dai, Y.-S., M. Xie, and K.-L. Poh, *Modeling and analysis of correlated software failures of multiple types.* IEEE Transactions on Reliability, 2005. **54**(1): p. 100-106.
2. Schneidewind, N.F. *An integrated failure detection and fault correction model*. in *Software Maintenance, 2002. Proceedings. International Conference on*. 2002: IEEE.
3. Monden, A., et al., *Assessing the cost effectiveness of fault prediction in acceptance testing.* IEEE Transactions on Software Engineering, 2013. **39**(10): p. 1345-1357.
4. Wedyan, F., D. Alrmuny, and J.M. Bieman. *The effectiveness of automated static analysis tools for fault detection and refactoring prediction*. in *Software Testing Verification and Validation, 2009. ICST'09. International Conference on*. 2009: IEEE.
5. Liu, S., et al., *Formal specification-based inspection for verification of programs.* IEEE Transactions on software engineering, 2012. **38**(5): p. 1100-1122.
6. Neeraj Mohan, P.S.S., and Hardeep Singh, *Impact of Faults in Different Software Systems: A Survey* 2010.
7. Andrews, J.H., et al., *Using Mutation Analysis for Assessing and Comparing Testing Coverage Criteria.* IEEE Transactions on Software Engineering, 2006. **32**(8): p. 608-624.
8. Zheng, J., et al., *On the value of static analysis for fault detection in software.* IEEE Transactions on Software Engineering, 2006. **32**(4): p. 240-253.
9. Zhou, C., R. Kumar, and S. Jiang. *Keynote: Hierarchical Fault Detection in Embedded Control Software*. in *2008 32nd Annual IEEE International Computer Software and Applications Conference*. 2008.
10. Poulding, S. and J.A. Clark, *Efficient Software Verification: Statistical Testing Using Automated Search.* IEEE Trans. Softw. Eng., 2010. **36**(6): p. 763-777.
11. Joshi, K.R., et al., *Probabilistic Model-Driven Recovery in Distributed Systems.* IEEE Transactions on Dependable and Secure Computing, 2011. **8**(6): p. 913-928.
12. Romi Satria Wahono1, a.N.S., *Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction.* International Journal of Software Engineering and Its Applications, 2013.
13. Oral Alan, *An Outlier Detection Algorithm Based on Object-Oriented Metrics Thresholds.* IEEE, 2009.
14. Catal, C., B. Diri, and B. Ozumut. *An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software*. in *Dependability of Computer Systems, 2007. DepCoS-RELCOMEX '07. 2nd International Conference on*. 2007.
15. Jeenam Chawla, A.A., *Object-Oriented Design Metrics to Predict Fault Proneness of Software Applications.* Jeenam Chawla et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, 2015.
16. Bharavi Mishra, *Impact of Attribute Selection on Defect Proneness Prediction in OO Software.* International Conference on Computer & Communication Technology, 2011.
17. Rathore, S.S. and A. Gupta. *Investigating object-oriented design metrics to predict fault-proneness of software modules*. in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. 2012.
18. J. Daly, V.P.L.B., et al., *Predicting Fault-Prone Classes with Design Measures in Object-Oriented Systems*, in *Proceedings of the The Ninth International Symposium on Software Reliability Engineering*. 1998, IEEE Computer Society. p. 334.

19. Cong, J., et al. *Quality prediction model of object-oriented software system using computational intelligence*. in *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*. 2009.
20. Supreet Kaur, D.K., *Quality Prediction of Object Oriented Software Using Density Based Clustering Approach.* IACSIT International Journal of Engineering and Technology, 2011.
21. Shatnawi, R. *Improving software fault-prediction for imbalanced data*. in *2012 International Conference on Innovations in Information Technology (IIT)*. 2012.
22. Singh, P. and S. Verma. *Empirical investigation of fault prediction capability of object oriented metrics of open source software*. in *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*. 2012.
23. Cruz, A.E.C. *Exploratory study of a UML metric for fault prediction*. in *2010 ACM/IEEE 32nd International Conference on Software Engineering*. 2010.
24. Gu´eh´eneuc, N.M.a.Y.-G.e., *On the Automatic Detection and Correction of Software Architectural Defects in Object-Oriented Designs.* 2005.
25. Liu, Y., et al., *A general modeling and analysis framework for software fault detection and correction process.* Software Testing, Verification and Reliability, 2016. **26**(5): p. 351-365.
26. Holley, R., *How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs.* D-Lib Magazine, 2009. **15**(3/4).
27. Wang, L., Q. Hu, and M. Xie. *Bayesian analysis for NHPP-based software fault detection and correction processes*. in *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*. 2015: IEEE.
28. Prattana Deeprasertkul, P.B., *Automatic detection and correction of programming faults for software applications.* The Journal of Systems and Software, 2005.
29. Liu, Y., et al. *A New Framework and Application of Software Reliability Estimation Based on Fault Detection and Correction Processes*. in *2015 IEEE International Conference on Software Quality, Reliability and Security*. 2015.
30. Y.P. Wu, Q.P.H., M. Xie and S.H. Ng, *Detection and Correction Process Modeling Considering the Time Dependency.* 12th Pacific Rim International Symposium on Dependable Computing IEEE, 2006.
31. Shrivastava, A.K., *Generalized Modeling for Multiple Release of Two Dimensional Software Reliability Growth Model.* INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH AND DEVELOPMENT, 2016.
32. Connors, A.S.T.M.V.J.R.J.B.D.A., *Using Process-Level Redundancy to Exploit Multiple Cores for Transient Fault Tolerance.* 2007.
33. Nasar, M., *Resource Allocation Policies for Fault Detection and Removal Process.* Modern Education and Computer Science, 2014.
34. Rashmi Upadhyay, P.J., *Review on Software Reliability Growth Models and Software Release Planning.* International Journal of Computer Applications, 2013.
35. Zhang, N., *Software Reliability Analysis using Queuing based Model with Testing Effort.* JOURNAL OF SOFTWARE, 2013.
36. Chin-Yu Huang1, C.-T.L., Sy-Yen Kuo2, Michael R. Lyu3, and Chuan-Ching Sue4, *Software Reliability Growth Models Incorporating Fault Dependency with Various Debugging Time Lags.* 2005.
37. Johri, M.N.a.P., *Testing and Debugging Resource Allocation for Fault Detection and Removal Process.* International Journal of New Computer Architectures and their Applications, 2014.
38. P.K. KAPUR1, A.G.A., and SAMEER ANAND2, *A New Insight into Software Reliability Growth Modeling.* International Journal of Performability Engineering, 2009.
39. Nan, Z., *Infinite Server Queuing Models with Resource Expenditures and Change-point for Software Reliability.* International Journal of Hybrid Information Technology, 2016.
40. Chandra Mouli Venkata Srinivas Akana, D.C.D., 3Dr. Ch. Satyanarayana, *Residual Fault Detection and Performance Analysis of G-O Software Growth Model.* Journal of Current Computer Science and Technology, 2015.
41. Marshima Mohd Rosli, *The Design of a Software Fault Prone Application Using Evolutionary Algorithm.* IEEE Conference on Open Systems, 2011.
42. Tamak, J., *A Review of Fault Detection Techniques to Detect Faults and Improve the Reliability in Web Applications.* International Journal of Advanced Research in Computer Science and Software Engineering, 2013. **3**(6).
43. Shu, Y., et al. *Considering the Dependency of Fault Detection and Correction in Software Reliability Modeling*. in *2008 International Conference on Computer Science and Software Engineering*. 2008.

44. Md.Zaffar amam , s.s., ahmed, *ANALYSIS OF SOFTWARE FAULT DETECTION AND CORRECTION PROCESS MODELS WITH BURR TYPE X11 TESTING EFFORT.* IEEE, 2016.

AUTHORS PROFILE