A Review on Software Fault Detection Mechanisms and Fault Prevention Mechanisms in Networks

Preeti Baderiya [1], Chetan Gupta [2], Shivendra Dubey [3]

[1] M. Tech. Scholar, Department of CSE Sagar Institute of Research Technology & Science, Madhya Pradesh, India

[2] Department of CSE Sagar Institute of Research Technology & Science, Madhya Pradesh, India

[3] Department of CSE, School of Engineering & Technology, Jagran Lakecity University, Bhopal, Madhya Pradesh, India

Emails: preeti.baderiya84@gmail.com; chetangupta.gupta1@gmail.com; shivendrashivay@gmail.com

Abstract

It is possible to improve software quality by anticipating fault location through the utilization of software metrics within fault prediction models in network. This article provides a comprehensive literature review on the topic of software fault forecasting. The paper also seeks to identify software metrics and evaluate how applicable those metrics are to the process of software fault prediction. It is recommended that additional research be conducted on large industrial software systems to identify metrics that are more pertinent for the industry and to find an answer to the question of which metrics should be employed in a particular setting.

Keywords: Software development; Fault Detection; Network; Fault Prevention; Software faults; Dynamic selection

1. Introduction

The quality of software can be improved with the help of fault prediction models, which can also assist with software inspection by finding potential bugs. The modelling approach and measurements used both have an impact on the performance of the model. There does not appear to be a significant performance gap between the various modelling techniques [1, 2], and the selection of a modelling methodology seems to have a less influence on the classification accuracy of a model than the selection of a metrics set [3]. In light of this, we came to the conclusion that it would be more fruitful to research the metrics that are utilised in software failure prediction rather than the modelling methodologies. As the number of software applications used in day-to-day activities continues to rise, assuring the dependability and quality of those apps has emerged as an essential component of software development. The existence of software errors has an impact on both the dependability and the quality of the software [4]. A rise in the number of software defects results in a decrease in the software's reliability, which in turn leads to an increase in the amount of effort required by the software to maintain its quality. According to the report that was published in 2018 by the Consortium for Information Technology Software Quality (CISQ), in the United States, 37.46% of the total IT industries have incurred costs as a result of the losses that have been caused by software failures. Furthermore, an additional 16.87% of the budget has been spent on finding and fixing faults. It demonstrates how time-consuming and costly it is to locate and correct software errors. According to yet another estimate, errors in software cost the world $2.84 trillion annually and affected more than 4 billion individuals. Therefore, the identification and removal of flaws are

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

required in order to preserve the quality of the programme and get rid of any potential risks to people, threats to life, or other disasters [5, 6]. In most cases, flaws in the software system can be found by employing software quality assurance (SQA) procedures such as static code inspection, testing, and peer review. However, software developers and testers have limited access to SQA resources and finances; therefore they are unable to conduct extensive software testing to guarantee that the system is free from errors. In addition, because of the large complexity of the software system, it is impossible to review and test each individual line of code. The goal of software fault prediction is to determine and discover portions of the software that include code in which the likelihood of a software error occurring is higher [7].

Below figure 1 represents the "system that predicts reliability" which utilizes the historical failure data of projects. The contribution of this thesis is to propose such a system which is used to predict software reliability of target projects. The below mentioned are the inputs and expected outputs from the system.

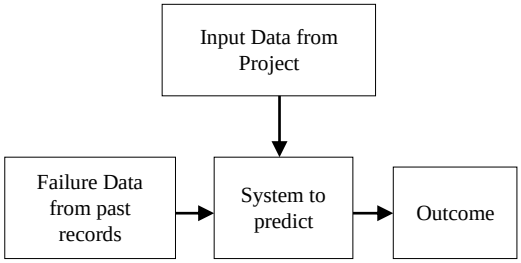

Figure 1: System for Reliability Prediction

The majority of errors in software are due to design flaws, which manifest themselves when a software engineer either fails to properly interpret a specification or simply makes a mistake while doing so. It is claimed that between 60 and 90 percent of today's computer faults are brought on by software malfunctions. These failure predictions have been studied in the context of fault-prone modules, self-healing systems, developer information, maintenance models, and so on; however, there are a lot of things that need to be explored for the fault severity in software development, such as modelling and weighting of the impact of different types of faults occurring in different types of software systems. The creation of high assurance systems begins with the establishment of performance requirements, followed by a focus on reliability. On the basis of the failure analysis, it has been shown to be an effective tool for identifying and preventing defects requirements early on in the software lifecycle. By adopting a generic taxonomy fault, one is able to more effectively avoid repeating mistakes from the past and create needs specifications that contain less general failures. If there are fewer failures in the software definition, in terms of the criteria for performance and reliability, then the resulting systems will have a high level of quality and security. The purpose of this paper is to provide an overview of the mechanisms involved in fault detection as well as approaches for the prevention of defects that can be followed in the process of developing quality software [8].

The remainder of this work is broken up into seven distinct sections. The section 2 is dedicated to provide a discussion of the mechanisms for detecting software faults and preventing software faults. The benefits of fault avoidance as well as its limitations are discussed in section 3. Section 2 presents works that are relevant to the topic and the conclusion is provided in section 5.

2. Software Fault Detection

Any error, fault, or imperfection in a work activity for a software product or software process that is caused as a direct result of an error, fault, or failure is referred to as a failure. According to the definition provided by the IEEE Standards, an error is "a human activity that leads to erroneous results." A fault can be defined as an incorrect decision made when attempting to comprehend the information provided in order to solve the problems or complete the application process. A single mistake may result in one or more faults, and a number of errors may ultimately lead to a failed attempt. In order to prevent this kind of error in software products, faults detection activities are carried out during each and every phase of the software development life cycle (figure 2). These activities are prioritised according to the level of importance they hold [9].

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

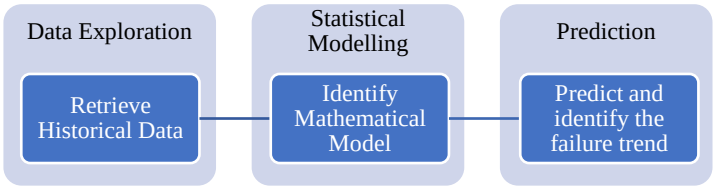

Figure 2: Different Activities in Software Fault Prediction

A. Detection Using Automated Static Analysis

The majority of Automated Static Analysis (ASA) detection is carried out for Manual Code analysis, which is one of the oldest practises that are still practised. However, automated tools are increasingly being used, particularly for the basic troubles related to non-compliance faults, possible memory leaks, variable usage, and other such issues.

B. Detection Using Graph mining

Graph Mining is an approach that helps detect defects that may not be crashing in their natural state. It is based on the concept of dynamic control flow.

C. Detection Using Classifiers

Classifiers built on the clustering algorithm, decision tree, or neural network can be utilised in order to differentiate aberrant events from normal ones for the purposes of detection. Labelling defective tracks whenever a flaw is spotted is another method that can be used to produce classifiers. The Naive Bayes and bagging classifiers are two that are used frequently. Bayesian classification is a form of supervised learning that may also be used as a statistical tool for classifying data. presenting an underlying probabilistic model that, when combined with a reasoned model for evaluating the probability of events, enables us to take into account the uncertainty involved.

D. Detection Using Pattern Mining

Pattern-based detection is also known as classifier-based detection; however, it employs one-of-a-kind iterative patterns for the purpose of categorization sequential data via software analysis of trace for fault diagnosis. A network of distinguishing characteristics extracts recurring sequences of occurrences from the execution traces of the programme when it is initially run.

E. Software Fault Prevention Mechanism

The process of developing software was fraught with many errors, which became apparent over time. It is a fallacy to suppose that errors are introduced at the start of the cycle and then eliminated during the remainder of the development process [10]. The flaws are present throughout the entirety of the development process. As a result, the prevention of errors should be considered an essential component in the process of increasing the quality of software. The method of fault prevention is an aspect of quality improvement that seeks to determine the factors that most often lead to faults and then make adjustments to the relevant procedures in order to stop the same kind of error from happening again. Additionally, it improves the quality of a software product while simultaneously cutting overall expenses, as well as saving time and resources. This guarantees that a project is able to maintain a balance between its time, cost, and quality [11]. The goal of fault prevention is to locate problems early on in a product's life cycle, then devise strategies to stop those problems from occurring in the future. This will ensure that the problem never manifests itself again.

F. Importance of Fault Prevention

The method of fault prevention is an aspect of quality improvement that seeks to determine the factors that most often lead to faults and then make adjustments to the relevant procedures in order to stop the same kind of error from happening again. Additionally, it improves the quality of a software product while simultaneously cutting overall expenses, as well as saving time and resources. This guarantees that a project is able to maintain a balance between its time, cost, and

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

quality. The objective of fault avoidance is to spot problems at the start of the life cycle and to stop it from happening again so that the fault cannot occur again.

In the development cycle of any software project, one of the most crucial activities is fault avoidance. The majority of a software project team will concentrate on bug fixing and finding errors. As a result, the prevention of faults is frequently an overlooked component.

G. Activities in Fault Prevention

Fault Identification: - A defect might be a pre-planned activity with the goal of publicising the particular problems that have been discovered. In general, errors can be discovered during activities such as design review, code inspection, graphical user interface review, function testing, and unit testing that are carried out during various stages of the software development life cycle.

Fault Classification: The general Orthogonal Defect Classification (ODC) method can be used to classify faults [12, 13, 14, and 15] in order to determine the fault group and its type. This can be done to find the fault. The ODC method divides flaws into categories based on when they were discovered and when they were repaired.

Fault Analysis: The continual process of quality improvement through the utilisation of defect data is called fault analysis. In order to make an attempt to determine what may have caused the problem; fault analysis is typically broken down into two categories: blame and direct process improvement initiatives.

Fault Prevention: In every software development project, one of the most crucial activities is error avoidance. Find out what caused the issues in the first place, because the goal of fault prevention is to stop the problems from happening again. Fault Prevention had suffered in the past to analyse the faults and faults in the future to prevent the occurrence of these types include special operations. This analysis was done in order to prevent the occurrence of these faults.

3. Fault Prevention Benefits and its Limitation

There are solutions for fault prevention in network, but they require a high level of checking maturity and discipline, which when combined with the testing effort constitutes the most cost-effective expenditure. It is necessary to have something in place that helps to prevent errors from escaping during the development life cycle, from the design phase to the implementation of code specifications. As a result, testing procedures can be split up into two distinct groups, namely, fault detection technologies and fault prevention technologies [16, 17, 18, 19, and 20]. Spend time and money on preventing errors throughout the development of an application results in significant cost and time saving. Therefore, it is also crucial, as it decreases the number of faults for reconstruction, which in turn delivers a reduction in cost, as well as the fact that it is simple to maintain the port and reuse. Additionally, it is essential for the company to be able to construct high-quality systems in a shorter amount of time while simultaneously providing resources and ensuring the system's dependability [21]. Faults are identified, leading to the implementation of preventative measures that, in turn, lead to an increase in productivity. These measures can be traced back to the life cycle stage at which they were injected. A mechanism can be thought of as a corrective measure that promotes the understanding of lessons acquired from one project to the next. Because there is a dearth of specific domain knowledge in areas where new and varied domain software needs to be developed and implemented. In many instances, the necessary quality requirements that are supposed to be defined are not in the first place. The inspection process is extremely labour-intensive and calls for a high level of ability. It is possible that well-developed quality measurements were not discovered at the time of design in some cases [22, 27].

4. Literature Review

During the software testing process, Y Li et al.[5] presented a time series analysis model for forecasting both the identified number of faults as well as the number of problems that have been rectified. The methodology is demonstrated with examples taken from real datasets.

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

A.G. Singh et al.[6] provided an experimental investigation of the effectiveness of the feature selection methods, specifically gain ratio (GR), info gain (IG), OneR, ReliefF, and symmetric uncertainty (SU), to develop the highly accurate classifier for the goal of enhancing the accurateness of software error detection.

Using the Naive Bayes methodology, Xing et al.[7] proposed both an automatic fault detection framework as well as a fault detection model. Our method is able to obtain the training data and transform them into the detection model by doing an analysis on the service execution logs that are already available. Our model identifies service faults by computing the service posterior probability to each fault type using the Naive Bayes method and the given threshold value. This helps us determine which faults have occurred. The findings of our experiments indicate that using our strategy is an excellent way to identify service issues.

Y.Wang et al. [8] suggest a strategy that utilises algebra graph representation in order to identify pairwise-constraint software flaws. The preliminary findings indicate that the methodology can identify pairwise-constraint software defects prior to the testing of the software.

The state-of-the-art publications are reviewed by S.kong et al. [9], who find that content failures are responsible for the majority of all sorts of software failures, despite the fact that its detection techniques are infrequently explored. In this body of work, we present a novel failure detection indicator for software content failures that is based on the knowledge on the dynamic execution of the software during runtime. In addition to this, we assess the precision of the machine learning models developed for this work.

K.Yeon et al. [10] discusses a method for a fault detection and diagnostic system for a vehicle that is equipped with advanced vehicle E/E systems. This method applies to both the domain control units and the garnish systems. This leads to the discovery of faults with a kind and magnitude that varies over time.

The purpose of this article by T.Alakus et al.[11] is to provide information regarding well-known software quality indicators and their application in the context of software fault prediction studies. In order to accomplish this goal, separate analyses of these three metrics, along with demonstrations of their respective collection methodologies, were included in this paper.

R.Natella et al.[12] examine a number of well-known tools that have been used to detect software ageing in actual, complex software systems. These tools have been adopted by researchers as well as by software developers. For every tool, we do an in-depth analysis of its inner workings, use cases, the signs of ageing that it combats, and the pertinent applications in real-world case studies. We have included tools for native software, such as system software that is written in C and C++. These tools monitor resource consumption at the operating system level and probe software internals, such as heap allocations.

M.Izani et al.[13] proposed a new topology of Cuk rectifier with open circuit fault detection approach. The suggested rectifier features a total of four modes of operation. Therefore, the rectifier continues to operate in closed loop, with the controller of the circuit monitoring the output of the rectifier. This allows the options to be switched between when there is an open circuit fault. Consequently, it is suggested that the controller transition into a different mode based on the malfunction. MATLAB simulation software is used to perform the examination of an open circuit fault tolerant Cuk rectifier as well as its fault detection.

Using the four benchmarks with configuration options interaction faults, Zhao et al.[14] compared the selected sample sizes and the fault-detection capabilities of FWA and t-wise algorithms. They found that FWA can detect a higher number of faults with the less selected sample size. More specifically, FWA can detect high-wise interaction faults with the less selected sample size compared with the 4-wise as well as the 5-wise algorithms.

Table 1: Research Contributions in Software Fault Detection

Ref	Author	Technique used	Accuracy	Result
[1]	Rathore & Kumar	Machine learning was used to find fault detection.	83.5%	Distance measure based on the given dataset characteristics in the approach are presented.

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

[2]	Rathore & Kumar	A study on software fault prediction techniques.	-	The statistical analysis are best suitable features to detect fault.
[3]	Radjenović et al.	Presented a review on fault detection models.	-	Object oriented metrices are best suitable features to detect fault.
[4]	Dhanalaxmi et al.	Presented a review on fault detection models.	-	Recent trend of the latest technologies have been discussed.
[5]	Y Li et al.	Present a time series analysis model for forecasting	89%	Demonstrated with examples taken from real datasets.
[6]	A.G. Singh et al.	Specifically gain ratio (GR), info gain (IG), oner, relieff, and symmetric uncertainty (SU),	88.88%	Enhancing the accurateness of software error detection.
[7]	Xing et al.	Naive baye	86.3%	Identifies service faults by computing the service posterior probability. Indicate that using our strategy is an excellent way to identify service issues.
[8]	Y.Wang et al.	Suggest a strategy that utilises algebra graph representation	-	Indicate that the methodology can identify pairwise-constraint software defects
[9]	S.kong et al.	Failure detection indicator for software content failures that is based on the knowledge on the dynamic execution	99.5%	Novel failure detection indicator
[10]	K.Yeon et al.	Fault detection and diagnostic system	-	Discovery of faults with a kind and magnitude that varies over time.
[11]	T.Alakus et al.	Software quality indicators and their application in the context of software fault prediction	-	Accomplish this goal, separate analyses of these three metrics, along with demonstrations
[12]	R.Natella et al.	Detect software ageing in actual, complex software systems.	91%	Monitor resource consumption at the operating system level and probe software internals, such as heap allocations
[13]	M.Izani et al.	New topology of Cuk rectifier with open circuit fault detection approach.	91.4%	Used to perform the examination of an open circuit fault tolerant Cuk rectifier as well as its fault detection.
[14]	Zhao et al.	Compared the selected sample sizes and the fault-detection capabilities of FWA and t-wise algorithms	-	FWA can detect high-wise interaction faults with the less selected sample size

5. Problems Statement

From the literature review presented above, it is observed that significant work has been done in the area of Software Reliability optimization. Various approaches have been used in development of Reliability growth models like:

- Based on imperfect debugging
- Based on the severity of errors in the software
- Based on Testing Effort

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

- Moving From Single to Multi-Dimensional Framework

Following major issues occurs identified in above mentioned models:

- To identify the parameters or issues that causes failure in software.
- To identify the statistical relationship among these failure causing factors.
- To identify the best suitable parameters that can correctly diagnose the failure causing factors.

The main problem with estimating software reliability is that the data is unbalanced. Dealing with unbalanced, inaccurate data creates havoc for software engineers and academics. The problem with foresight is that it produces a lot of problems. Researchers are looking on machine learning algorithms-based software defect prediction approaches, although they haven't looked into it in detail yet. Reliability Prediction has been the subject of several studies. For anticipating software defects, machine learning and statistical approaches are recommended.

6. Research Objectives

Reliable software must include extra, often redundant, code to perform the necessary checking for exceptional conditions. This fundamental role means that the reliability of systems software is of primary importance. Therefore, enhancing the software reliability is very important before the software is released [23]. The proposed work will make software more reliable.To accomplish the proposed research work, following objectives will be adopted:

- First of all, a detailed study and analysis of various factors and attributes contributing towards System's Reliability.
- To design a machine learning approach for software reliability predictions and identify the fault causing factors.
- To improve the software reliability performance accuracy.

7. Conclusion

There is a fundamental requirement for highly fault-tolerant systems in today's world, as software reliability is gaining more and more attention these days. This survey paper discusses research on fault detection mechanisms as well as fault prevention mechanisms in relation to the most recent trend in the most advanced technologies. These topics are discussed in relation to one another throughout the paper. There are a large number of methods and techniques, and flaw detection and software systems are used to diagnose them. However, not every method and technique are compatible with every system. Determine the organisation of the technological system, the size and complexity of the adaptability and reliability targets, the technological platform, and be driven by the important variables. An automated method to detect a tendency to higher levels in hybrid mining methods and statistical models are in the process of leaning more toward classical systems-oriented alternatives for diagnosis, treatment, and prevention. This approach is being taken because of the complexity of the problem. Fault handling in today's applications is still in the early stages of study, and the service-oriented architecture tries to build level of tolerance as much as possible.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] S. S. Rathore and S. Kumar, "Software fault prediction based on the dynamic selection of learning technique: findings from the eclipse project study," Appl. Intell., vol. 51, no. 12, pp. 8945–8960, 2021, doi: 10.1007/s10489-021-02346-x.

[2] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," Artif. Intell. Rev., vol. 51, no. 2, pp. 255–327, 2019, doi: 10.1007/s10462-017-9563-5.

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

[3] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," Inf. Softw. Technol., vol. 55, no. 8, pp. 1397–1418, 2013, doi: https://doi.org/10.1016/j.infsof.2013.02.009.

[4] B. Dhanalaxmi, G. Apparao Naidu, and K. Anuradha, "A Review on Soware Fault Detection and Prevention Mechanism in Soware Development Activities Related papers A Review on Software Fault Detection and Prevention Mechanism in Software Development Activities," IOSR J. Comput. Eng., vol. 17, no. 6, pp. 25–30, 2015, doi: 10.9790/0661-17652530.

[5] Y. LI, Y. MA, R. PENG, and K. GAO, "Prediction of Software Fault Detection and Correction Processes With Time Series Analysis," in 2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM), 2020, pp. 1–6. doi: 10.1109/APARM49247.2020.9209402.

[6] D. A. A. G. Singh, A. E. Fernando, and E. J. Leavline, "Experimental study on feature selection methods for software fault detection," in 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), 2016, pp. 1–6. doi: 10.1109/ICCPCT.2016.7530156.

[7] X. Xing, J. Luo, Z. Jia, Y. Li, and Q. Han, "Automated Fault Detection for Web Services using Naïve Bayes Approach," in 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), 2019, pp. 336–339. doi: 10.1109/ICSESS47205.2019.9040756.

[8] Y. Wang, X. Chen, W. Zhou, X. Liu, J. Li, and G. Lu, "Using Algebra Graph Representation to Detect Pairwise-Constraint Software Faults," IEEE Access, vol. 8, pp. 184550–184559, 2020, doi: 10.1109/ACCESS.2020.3029094.

[9] S. Kong, M. Lu, B. Sun, J. Ai, and S. Wang, "Detection Software Content Failures using Dynamic Execution Information," in 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2021, pp. 141–147. doi: 10.1109/QRS-C55045.2021.00029.

[10] K. Yeon and D. Lee, "Fault detection and diagnostic coverage for the domain control units of vehicle E/E systems on functional safety," in 2017 20th International Conference on Electrical Machines and Systems (ICEMS), 2017, pp. 1–4. doi: 10.1109/ICEMS.2017.8056361.

[11] T. B. Alakus, R. Das, and I. Turkoglu, "An Overview of Quality Metrics Used in Estimating Software Faults," in 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), 2019, pp. 1–6. doi: 10.1109/IDAP.2019.8875925.

[12] R. Natella and A. Andrzejak, "SAR Handbook Chapter: Experimental Tools for Software Aging Analysis," in 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2020, p. 1. doi: 10.1109/ISSREW51248.2020.00096.

[13] M. S. H. M. Izani, K. S. Muhammad, and R. Baharom, "Open Circuit Fault Tolerant Bridgeless Cuk Rectifier with Fault Detection Technique," in 2021 IEEE Industrial Electronics and Applications Conference (IEACon), 2021, pp. 207–212. doi: 10.1109/IEACon51066.2021.9654750.

[14] J. Zhao, G. Ning, H. Lu, Y. Wang, C. Yan, and J. Zhang, "Poster: A Weight-Based Approach to Combinatorial Test Generation," in 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), 2018, pp. 378–383.

[15] Bushra Khalid, Kapil Sharma. (2015). Ranking of Software Reliability Growth Models Using Bacterial Foraging Optimization Algorithm. International Conference on Computing for Sustainable Global Development. IEEE, 1643-1648.

[16] K. Lu and Z. Ma. Parameter Estimation of Software Reliability Growth Models by A Modified Whale Optimization Algorithm.(2018). International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), 268-271.

[17] L. Zhen, Y. Liu, W. Dongsheng and Z. Wei. (2020). Parameter Estimation of Software Reliability Model and Prediction Based on Hybrid Wolf Pack Algorithm and Particle Swarm Optimization. IEEE Access, 8, 29354-29369.

[18] M. Gheisari et al. (2019). An Optimization Model for Software Quality Prediction With Case Study Analysis Using MATLAB. IEEE Access, 7, 85123-85138.

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022

[19] P. Prashant, A. Tickoo, S. Sharma and J. Jamil. (2019). Optimization of cost to calculate the release time in software reliability using python. International Conference on Cloud Computing, Data Science & Engineering (Confluence), 471-474.
[20] P. Roy, G. S. Mahapatra and K. N. Dey. (2019). Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network. IEEE/CAA Journal of Automatica Sinica, 6(6), 1365-1383.
[21] Ramakanta Mohanthy, Venkatshwarlu Naik, Azmath Mubeen. (2014). Predicting Software Reliability Using Ant Colony Optimization Technique. International Conference on Communication Systems and Network Technologies, 496-500.
[22] R. K. Mohanty, V. Ravi, and M. R. Patra. (2013). Hybrid intelligent Systems for predicting Software reliability," Elsevier, Applied Soft Computing, 13(1), 189-200.
[23] Z. Li, M. Yu, D. Wang and H. Wei. (2019). Using Hybrid Algorithm to Estimate and Predicate Based on Software Reliability Model. IEEE Access, 7, 84268-84283.
[24] Salama, A. A., Smarandache, F., &Kroumov, V., Neutrosophic crisp Sets & Neutrosophic crisp Topological Spaces. Sets and Systems, 2(1), 25-30, 2014.
[25] Smarandache, F. &Pramanik, S. (Eds). (2016). New trends in neutrosophic theory and applications. Brussels: Pons Editions.
[26] Alhabib, R., The Neutrosophic Time Series, the Study of Its Linear Model, and test Significance of Its Coefficients. Albaath University Journal, Vol.42, 2020, (Arabic version).
[27] Kumar A, Dubey S, Arshad M, Saxena S, Sinha SK, Dixit P, Arjaria A. Enhanced Cloud Data Storage Security by Using Hadoop. InProceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020 Mar 30.

DOI: https://doi.org/10.54216/IJWAC.060203
Received: September 18, 2022 Accepted: November 10, 2022