IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

A Research Study to Forecast Software Quality Using Machine Learning

K. Rama Mohana Rao[1], K. Nikitha[2], G. D. V.Prasad[3], Y. Santhi Sri[4], B. Rahul Chowdary[5]

Department of Information Technology, Sasi Institute of Technology & Engineering

1. Abstract:

Software quality estimation is a necessary task at various stages of software development. It can be used to arrange the project's quality assurance processes and to establish benchmarks. Previous studies had used two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) to evaluate the quality of software. Furthermore, C5.0, SVM, and Neutral Network studies were carried out for quality estimation. These investigations have a relatively low degree of precision. In this work, we aimed to improve estimation accuracy by leveraging relevant data from a large dataset. We used a correlation matrix and feature selection method to improve accuracy.

Keywords: Software engineering, software measurement, software metrics, software quality, machine learning, algorithms.

Additionally, we have experimented with more modern methods that have proven effective for many prediction tasks. To forecast software quality and determine the correlation between quality and development characteristics, machine learning techniques are applied. These algorithms include of CNN, Decision Tree, Random Forest, Bagging classifier, Gradient boosting, Logistic Regression, and Navie-Bayes. The results of the experiment show that machine learning algorithms can estimate the software quality level with accuracy.

2. INTRODUCTION:

Throughout history, technological advancement has never been more rapid than it is today. Along with the innovations that emerge on a daily basis, the software business benefits from this expansion. The rapid development of the software business is unavoidable due to the atmosphere created by people's reliance on technology in all fields. Interest and demand in this field have grown as software is used in numerous sectors to improve people's lives. Furthermore, today's competitive atmosphere draws enterprises into the software industry, regardless of sector. Companies may seek to improve their market share by distinguishing themselves from competitors with innovative software. This has led to the necessity of quality software.

Defects in software applications can arise from requirements analysis, definition, and other software development processes. As a result, there are different stages at which software quality estimation is required [1]. It can be utilised for benchmarking and for

organising project-based quality assurance procedures. Furthermore, one of the key indicators of the software's quality is thought to be the quantity of flaws per unit [2].

Software development requires the activity of software quality estimate at different phases. It can be utilised for benchmarking and for organising the project's quality assurance procedures. Two techniques (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for assessing software quality had been applied in earlier research. Additionally, experiments were conducted with C5.0, SVM, and Neutral Network for quality estimation. The accuracy of these investigations is comparatively low.

In this research, we used a feature selection method and a correlation matrix to achieve greater accuracy. We also tested new strategies that have proven effective for other prediction challenges.
Various machine learning methods, including Navie Bayes, Gradient Boosting, Logistic Regression, Bagging Classifier, Random Forest, Decision Tree, and CNN, are used to forecast software quality and identify correlations with development variables. Experiments demonstrate that machine learning systems can accurately measure software quality.

3.RELATED WORK:

The purpose of software testing is to shield consumers from harm caused by internal program flaws. Software testing can be defined as the process of verifying and attesting to the program's or application's bug-free status. J. Morgan et al. offered an empirical study on software test effort estimation for defense programs [11]. Problems that this model solves with "highly accurate approximations" and a "viable alternative" To overcome these obstacles, software test matrix techniques are applied. The data set utilized in the paper file consists of test attempts made by Software Test. In order to offer a methodology for predicting software test effort using particular software test metrics, an empirical investigation was carried out, as described in the paper. In this study, a novel set of metrics for software testing is proposed. Testing metrics for estimating software testing effort and the suggested approach were discovered to be extremely precise. The proposed method outperforms other current approaches, such the test-point analysis technique, which is intricate and challenging to use and understand. Similarly, Mykhailo Lesinski et al. propose Expanding the monitoring software test environments: real-world examples. [12] This essay explores a number of software testing-related issues and suggests solutions. Among them is the low error detectability, insufficient test coverage and minor flaws that gradually become noticeable. The work employs a range of techniques to deal with issues including viewing test execution from various observational vantage points. Performance metrics, produced events, test oracles.

An Additional Difficult Metamorphic Strength Test: Exposes Information Hidden in Citation Statistics and Journal Impact Factors Zhou Zhi Quan and others [13] In order to improve software quality and testing procedures, it is critical to comprehend the traits and evolution of test failures in continuous integration (CI). This paper tackles this issue. provides a comprehensive, long-term analysis of the success and failure of CI tests utilizing T-Evos to solve issues. The T-Evos data set is used in this work. an extensive collection of environmental changes and test outcomes in continuous

integration situations. The process of T-Evos involves examining the database, which contains details on the test results, structure, code modifications, bug reports, and the sentence-level code environment. The researcher makes inferences about the traits and development of test flops in continuous integration (CI) systems by utilizing statistical analysis and visualization approaches to spot patterns and trends in the data. The research, which is based on an empirical analysis of the T-Evos database, offers some original results and conclusions. Test failures are among the significant outcomes and work completed; the majority of test failures are fixed in a day or less. Our paper's advantages include: Through the use of an extensive dataset of test results and coating evolution in continuous integration scenarios, the research gives a rigorous longitudinal investigation of CI test performance and failure.

Black deep neural network box testing with a variety of test scenarios was suggested by Zohreh et al. [14]. DNS in the real world is a difficulty. Using a black-box technique, the major objective is to identify diversity that can direct DNN testing without depending on the DNNs' experimental performance and outcomes. Cifar-10, MNIST, Moda-MNIST, and SVHN are four popular image recognition databases that were used in the paper's tests. The paper underwent two stages of testing. First, they investigated their capacity to accurately quantify diversity in the input set by choosing and modifying three diversity metrics. The efficacy of the proposed technique in identifying faults in DNN is assessed in the second stage using an error checking exercise. Restrictions The method used by the author is predicated on the idea that geometric variety and faults in DNNs are connected. This presumption, however, does not account for all DNN model flaws.

Ai Liu et al. suggested that formal test-based verification may be made more feasible by controlling the software package's process. [15] The study tackles the problem of enhancing TBFV's capabilities through software implementation. This study presents an axiomatic approach to address implicit processes in software, hence enhancing the capabilities of Test-Based Formal Verification (TBFV). When compared to specification-based testing (SBT), this approach performed better in discovering problems, according to trials conducted to evaluate it. Two quick tests to determine how well the approximation works. The paper's distinctive finding is that the suggested method enhances TBFV flaw detection capabilities while successfully resolving the specified software process. The authors' experiments showed that this method was superior to SBT in terms of flaw detection. The intricacy of the target system utilized for testing and the challenge of creating Hoare's axioms for operation are two of the strategy's limitations.

Similarly, Kun Chiu et al. [16] propose testing under stress with variable conditions to accelerate the discovery of software defects. The recommended method comprises stress testing with influencing variables and developing a mathematical relationship model between the influencing factors and the statistical characteristics of race data. The failure process that follows a data race failure has been attributed to three factors: memory restriction, concurrency level, and parallelism level. This paper's data collection method is statistical analysis. The stress test methodology, which investigates the implications of the suggested contributing

causes for the failure of information processing software and their mathematical relationship models, is used to conduct the tests. The study offers seven intriguing findings that offer practical recommendations for software data analysis, modeling, and assessment. The suggested methodology's shortcomings and potential future paths are mentioned in the study. The suggested methodology might not be appropriate for other kinds of software failures and is only applicable to data race issues.

Xiangying Dang et al. presented enhanced mutation testing utilizing fuzzy clustering and population genetic techniques [17]. Enhancing mutation testing through fuzzy clustering and multipopulation genetic algorithms is a paper-based problem. The test data generation problem is utilized as an optimization of a multipopulation genetic algorithm to generate test results for killing mutations in many groups simultaneously. This set of techniques is applied within the FUZGENMUT, a complete framework. To assess the efficacy of the proposed method, the study runs experiments based on nine distinct measurement programs.Experiments are conducted throughout the study to assess the efficacy of the suggested framework. The study suggests a complete framework called FUZGENMUT that enhances the effectiveness of test data generation for mutation testing by combining population genetic algorithms and fuzzy clustering through private sharing. This research points out a number of drawbacks with the suggested framework, including the requirement for manual cluster parameter configuration and the high computing cost of fuzzy clustering.

Amirhossein Zolf Agharian et al. suggested a research-based test procedure for deep reinforcement learning bots. [18] When implemented in a security context, integrity and adherence to security criteria must be ensured. Using this method, test cases that fulfill security criteria and expand the state space are created, and tests are then conducted on the DRL agent to find any potential flaws. The authors investigated the feasibility of utilizing machine learning models to anticipate defective parts and did experiments to assess the efficacy of the recommended technique in terms of the number of discovered faults compared to random testing. This method effectively explores the state space and generates trials that both increase the state space and meet security criteria by combining directed and randomized research. The suggested method's limitations are acknowledged in the research, including the fact that it is limited to RL agents interacting with stochastic environments and having deterministic rules.

As stated in Safety and Suggested Approximation: Weaknesses in Estimate-Informed Testing by Dow et al. [19]. The authors demonstrated how two malicious tampering attacks—altering the error measure (TEM) and altering the exact net list (TEN)—could impact the classification outcomes of estimation-aware tests. This work uses a 16-bit approximation database, which is subjected to TEN and TEM attacks. The purpose of this work is to suggest two malicious tampering approaches to undermine the integrity of the estimation-aware testing process: altering the error measure (TEM) and the exact net number (TEN). There are a few issues with this work that can be resolved in further studies. Initially, the suggested assault is assessed on the 16-bit approximation promoter; the

impact of the attack on other approximation chains is unknown. Subsequent studies could assess the efficacy of the suggested attack on various kinds of approximation chains. The paper's advantage is that it offers a fresh viewpoint on approximate circuit security and proposes fresh methods of attack that could jeopardize the accuracy of approximate reporting tests.

Paul Tempel and additional individuals. [20] It is suggested to evaluate multimorphic tests empirically. A framework for discovering and assessing test set environments relevant to interest rates is proposed in this research. By finding inadequate configurations and optimizing the systems' configurations, this framework seeks to address the issue of test suite optimization for software systems. The "Multi-Morphic Test" is the procedure that is suggested in the paper. A computer vision competition data set is used in this work. The proposed "Multimorphic Testing" paradigm for assessing the efficacy of software system performance test suites is the paper's primary contribution. The findings demonstrate that the variance score can differentiate between test instances by assigning varying variance scores, indicating whether the variance score can result in higher or lower performance scores.

Anjana Perera et al. suggested an experimental study that would test research-based software utilizing theoretical error predictors. [21] previously looked at how well MOSA and Dyna MOSA detected defects, using noisy statistical tests to make inferences about the data. Pre MOSA is a suggested approach to the issue of enhancing error detection in research-based software testing. The paper makes use of the Defects4J dataset. 420 actual faults from the Defects4J database—acquired through an open-source project—are used in the experimental research. When utilizing appropriate error predictors, experimental evaluations demonstrate that Dyna is more successful than MOSA with large effect sizes. Some of the drawbacks of their study include the use of a retrospective database, which does not fully include clinical studies with restricted control in its database.

Salako Kizito and others. [22] When assessing dependability, it is advised that statistically independent tests be rejected by Bayesian software. In situations where the assumption of statistically independent testing is not satisfied, the paper offers a solution to the software reliability evaluation problem. The study presents a method called conservative Bayesian inference (CBI), which takes into account the uncertainty around the results of statistically significant tests used in the estimation. Provide and present a statistical model that illustrates the virtual system's success or failure. A statistical model for failure/success in dependent systems is presented, and a hypothetical application of conservative Bayesian methods (CBI) for reliability estimation is demonstrated. The paper's primary conclusion is that the CBI technique can yield more cautious and realistic dependability estimations encompassing the unpredictability of conducting separate statistical analyses. The authors highlight one of the CBI approach's drawbacks: it necessitates the collection of prior knowledge, which can be challenging and arbitrary.

A geographical analysis for research-based software testing design was proposed by Neelofar Neelo Long et al. [23]. This paper investigates the efficacy of research-based approaches in software testing and highlights

the advantages and disadvantages of each. It focuses on the issue of assessing this method's effectiveness and locating biases in important databases. In order to assess the efficacy of search-based software testing techniques, the study suggests using Instance Space Analysis. In order to investigate the impact of the features, this approach entails mapping the test method's effectiveness in the test region. The SF110 Corpus and Common Collection are two of the most popular datasets that are utilized in the article to extract CUTs (Code Under Test) for experiments. The paper underwent two stages of testing. They tested the efficacy of cutting-edge software testing techniques based on research on the SF110 Corpus and Cumulative Collection databases in the first phase. They employed Instance Space Analysis in the second stage to assess this method's efficacy in the instance space. The study has certain limitations, as the authors admit. They only assessed a small number of survey-based software testing techniques, which is one of their limitations. One benefit of the suggested Instance Space Analysis approach is that it provides an instance space representation of the portfolio method's instant feature distribution and performance.

Liu Shaoyi and others [24]. The "Vibration-Method," also known as the "V-Method," is a new technique for automatic testing and testing Oracle generation for conformance based on functional situations in formal specifications testing. It addresses the issue of user specifications and potential implementation paths in the application. The "Vibration Method" or "V Method" is the suggested troubleshooting technique to make sure that executable and user issues are satisfied. Several trials carried out by several groups were employed by the researchers to lessen the influence of human variables on the study's findings. Using the FBIDPM and the V-method, two distinct groups created two sets of tests, respectively. They represent the two tests using TEST1 and TEST2, which are two test sets. The article suggests a novel approach. The study suggests a brand-new technique for automatically creating test cases and test oracles from model-based formal specifications: the Vibration Method (V-Method). The intricacy of handling fixed and complex expressions, as well as the challenge of guaranteeing that the execution program's specification maintains the signature, are among the drawbacks of the V-method.

Shujuan Jiang et al. suggested an integration test sequence technique that took the control pair into account. [25] The research aims to solve this issue by presenting an integrated test order method that takes the control bond in the group connection into account. Con CITO, a strategy for creating integration tests, is the suggested approach to solving the issue of taking control relations in intergroup relations into consideration. The suggested ConCITO approach is evaluated in the paper using 10 different software packages as databases. Experiments are carried out in the paper to assess the suggested ConCITO approach. They started by performing a static analysis on ten subject programs in order to determine the transitive and direct links. The paper's novel conclusions and suggestions imply that the Con CITO approach be suggested while taking the control bond into account. This approach seeks to produce more satisfying results than other approaches while lowering production costs for a wide range of applications. Large-scale applications with intricate clustering relationships might not be a good fit for the suggested approach.

For mislocalization, Laleh Sh. Khandehari et al. [26] suggested a combinatorial test-based method. This study's experiments compare two spectrum techniques, Tarantula and Ochiai, using a tool called BEN. This research proposes a collaborative test-based methodology, named BEN, to solve the mislocalization problem. Combination detection and the localization of invalid sentences are the two primary stages of BEN. Four real programs, including flex, grep, zip, and sed, as well as seven smaller applications from the Siemens package from SIR served as the experiment's subject programs. The paper's original results and suggestions demonstrate the efficacy of the BEN combinatorial testing strategy in locating software flaws. BEN needs a limited quantity of produced tests. Compared to other spectrum-based approaches that necessitate tracking every test run, BEN is more efficient since it only requires a small number of tests that are generated in the second step of the technique. The study makes the assumption that the program being tested is deterministic, which means that the same input always yields the same output, with regard to the constraints and potential for future convergence. This method is predicated on the idea that each program defect is independent, which means that the existence of one fault has no bearing on the finding of other defects. The recommended approach has the following benefits: it is practical, scalable, efficient, and effective. It may identify several flaws in a single program and be used in sizable real-world applications.

Durelli, Vinicius H. S. et al. [27] Systematic Mapping Learning is Suggested Machine Learning for Software Testing. The issues with software testing are addressed in this work in significant detail, including how to enhance test selection, test oracle structure, fault localization, and test suite optimization. The methodological quality of primary research is analyzed in the paper, and it is discovered that many of them lack rigorous techniques for experimental design, data gathering, and analysis. In this study, novel results and insights from systematic mapping research at the interface of software testing and machine learning are presented. Recent years have seen a growing commitment to empiricism in certain significant results and research, with enhanced study designs that can more effectively support claims and conclusions from more recent studies. Few primary studies addressed reliability difficulties, and the majority of studies lacked rigor in their experimental design, data gathering, and analysis techniques. Limitations and future directions: The paper points out certain restrictions on the systematic mapping procedure and makes recommendations for future study topics. Future research could benefit from a more strict quality standard, as the study did not include a detailed examination of the source research's standard. For researchers and practitioners interested in investigating the nexus between software testing and machine learning, the study offers a useful starting place overall. However, more work needs to be done to expand the field and overcome the constraints noted in the study. In order to assist academics and practitioners in identifying gaps in the literature and testing software for future research, a systematic gap-mapping study offers an extensive and methodical examination of the body of literature currently available on the subject of utilizing machine learning to test software.

Zhangang Ying and others [28] suggests using a constraint model and software-based self-testing to examine superscalar processors that are out-of-order. The

challenge of developing functional tests for processors—more especially, random superscalar processors—is discussed in the study. It targets out-of-order superscalar processors and offers a novel software-based self-testing solution for processors that use BMC methodologies. When used in functional testing, this method yields extremely high error coverage in crucial components that is in line with the OOE procedure. The suggested approach begins with abstraction-purification at the module level, splitting the model under study to make it smaller. The default BMC solution generates the sequences required to activate the internal processor functions using a derived finite state machine. Extremely high fault coverage in crucial components appropriate for OOE operation is the outcome of the suggested approach. In order to assess how well the recommended method works for creating processor test cases, the study does experiments. The suggested approach was assessed using a set of benchmarks, which included an Alpha 21264 CPU, a PowerPC 604e processor, and a SPARC V8 processor. In order to minimize the size of the research model, the paper's unique results and the suggested strategy combine module-level abstraction-purification and slicing. BMC-based advanced sequencing is then used for each function to guarantee high defect coverage. module accessible. The suggested approach has the capacity to identify issues with OOE processors, which are challenging to test using traditional techniques.Constraints and Potential Limitations of the Suggested Method The suggested approach is ineffective for evaluating combinational circuits and is only applicable to sequential circuits. The suggested approach might not be useful for producing structural tests because it is limited to producing functional tests. The benefit of the proposed approach is that it produces function tests for processors in an efficient manner, leading to a very high error coverage in crucial components that are compatible with OOE operation in functional mode. Compared to alternative test strategies like random testing and internal self-testing (BIST), the suggested approach is more effective in terms of error coverage and test production time.

An integration test sequence technique was provided by Shujuan Jiang et al. [29] in order to take control coupling into account. The study offers an integrated test order technique that takes into account both direct and indirect linkages in order to solve the challenge of evaluating the complexity of control bonds in group relationships. Con CITO, or Control Coupling Test Procedure, is the name of the suggested technique for determining the control coupling's complexity and resolving the issue of creating integration test orders. The suggested ConCITO approach is assessed in this research by experimentation on ten subject applications.

Claudio Mandrioli et al.'s proposed self-adaptive software testing that offers probabilistic assurances for performance KPIs [30] The methods for assessing the adaptation system's adaptation layer's efficacy are covered in this article. The topic is challenging because of the significant degree of uncertainty and variability present in adaptive software. This study aims to develop and justify implicit software performance guarantees, like reaction time and reliability. Monte Carlo (MC) sampling technique. The problem is resolved using the Monte Carlo (MC) method. The maxima can be extracted from the database using a variety

of methods. Peaks Over Threshold and Maxima Block are the most prevalent. Whereas the latter retrieves all values that surpass a preset threshold, the former selects a subset of the database and extracts the highest value from each subset. This constraint stems from the arbitrary selection of test parameters, the assumption of infinite variability of the measured quantity, and unknown, case-dependent experimental confidence. Standard statistical tools are used in this article to examine test findings and offer statistical assurance on software behavior. It contains details on numerous approaches and strategies suggested by professionals to address issues with software testing and monitoring. This covers a range of techniques, including an empirical study that suggests ways to assess the efficacy of software testing by utilizing particular software testing criteria, a paper that looks at the efficiency of research-based techniques for software testing, and suggestions for ways to address the difficulties associated with software testing. underdiagnosis of flaws, lacking testing, and the gradual accumulation of little or unimportant flaws. The drawbacks of a few of these strategies and tactics are also covered.

TABLE I: PERFORMANCE COMPARISON OF SOFTWARE TESTING

Authors	Method	Dataset
Morgan *et al.* [1] [2022]	Software test metrics	test efforts
Mykhailo Lasynskyi *et al*. [2] [2021]	Generated Events, Performance Metrics, Monitoring Test Execution from Different Observation Perspectives	test oracles
Zhi Quan Zhou et al.[3] [2021]	Longitudinal The execution of CI tests and Failure, T-Evos Dataset Analysis, Statistical Analysis, Visualization Techniques, Empirical Study, Key Findings and Takeaways	T-Evos
Zohreh et al. [4] [2023]	diversity metrics	Cifar-10, MNIST, Fashion-MNIST, and SVHN.
Ai Liu et al. [5] [2023]	axiomatic approach	specification-based testing
Kun Qiu et al. [6] [2020]	formulating mathematical relationship models	statistical analysis
Xiangying Dang et al. [7] [2022]	fuzzy Clustering and genetics of many populations	FUZGENMUT
Amirhossein Zolfagharian et al. [8] [2023]	novel search-based testing	DRL

Yuain Dou et al. [9] [2023]	tampering of the exact netlist (TEN) and tampering of the error metric (TEM)	TEN and TEM
Paul Temple et al. [10] [2021]	Multi morphic Testing	computer vision competition
Anjana Perera et al. [11] [2023]	Pre MOSA	Defects4J
Kizito Salako et al. [12] [2023]	conservative Bayesian inference (CBI)	statistical model
Neelofar et al. [13] [2023]	Instance Space Analysis	SF110 Corpus and commons-collections
Shaoying Liu et al. [14] [2020]	Vibration-Method	Universal Card System
Shujuan Jiang et al. [15] [2020]	Con CITO	Con CITO method
Laleh Sh. Ghandehari et al. [16] [2020]	Combinatorial Testing-Based Approach BEN	SIR (Software-artifact Infrastructure Repository)
Vinicius H. S. Durelli et al. [17] [2020]	solution proposals	machine learning
Ying Zhang et al. [18] [2020]	module-level abstraction	BMC
Shujuan Jiang et al. [19] [2020]	Con CITO, which stands for "Control Coupling Integration Test Order	transitive relationship
Claudio Mandrioli et al. [20] [2022]	Monte Carlo Sampling (MC)	Monte Carlo

4.ANALYSIS AND DISCUSSION:

Software testing is the process of testing and verifying that software or programs are error-free, meet technical requirements, and meet effective and meets user criteria efficiently. The software testing process involves not only finding defects in existing software, but also finding ways to upgrade the software so that efficiency, accuracy, and usability. It primarily measures the specifications, functionality and performance of an application or software program. Software testing of the reliability variety assesses the system's ability to perform its tasks consistently and successfully over an extended long period of time. Reliability testing is the identification and resolution of problems that could result in a system failing or becoming unavailable. Software reliability is the probability that a computer program will fail within a specified time under specified conditions. This article explores ways and to make something simpler software testing.

5.CONCLUSION:

In order to standardize software testing procedures, this article suggests a standard

workflow technology. It also builds a workflow management system to enable efficient software process management testing. We focus on integrating the standardization system in the software test system and the actual test process, using the RSA time attack program and workflow test system as the test object. We also extract and describe the test work standards to illustrate the test process. We ran many SRGMs to assess the program's reliability during numerical experiments using two datasets of reported mistakes in actual software testing. We then compared the resilience and prediction performance of these SRGMs. This article's main goal is to make software simpler. Lastly, it's employed to simplify software**.**

6. REFERENCES:

[1] Mandrioli, Claudio, and Martina Maggio. "Testing software that is self-adaptive and offers probabilistic performance guarantees metrics." In European Software Engineering Conference and Symposium on the Foundations of Software Proceedings of the 28th ACM Joint Meeting Engineering, pp. 1002-1014. 2020.

[2] Wang, Junjie, Song Wang, Jianfeng Chen, Tim Menzies, Qiang Cui, Miao Xie, and Qing Wang. "Characterizing crowds to better optimize worker recommendation in crowdsourced testing." *IEEE Transactions on Software Engineering* 47, no. 6 (2019): 1259-1276.

[3] Perera, Anjana, Aldeida Aleti, Burak Turhan, and Marcel Böhme. "An experimental assessment of using theoretical defect predictors to guide search-based software testing." *IEEE Transactions on Software Engineering* 49, no. 1 (2022): 131-146.

[4] Lasynskyi, Mykhailo, and Janusz Sosnowski. "Extending the space of software test monitoring: practical experience." *IEEE Access* 9 (2021): 166166-166183.

[5] Salako, Kizito, and Xingyu Zhao. "The Unnecessity of Assuming Statistically Independent Tests in Bayesian Software Reliability Assessments." *IEEE Transactions on Software Engineering* (2023).

[6] Durelli, Vinicius HS, Rafael S. Durelli, Simone S. Borges, Andre T. Endo, Marcelo M. Eler, Diego RC Dias, and Marcelo P. Guimarães. "Machine learning applied to software testing: A systematic mapping study." *IEEE Transactions on Reliability* 68, no. 3 (2019): 1189-1212.

[7] Neelofar, Neelofar, Kate Smith-Miles, Mario Andrés Muñoz, and Aldeida Aleti. "Instance Space Analysis of Search-Based Software Testing." *IEEE Transactions on Software Engineering* 49, no. 4 (2022): 2642-2660.

[8] Zhang, Ying, Krishnendu Chakrabarty, Zebo Peng, Ahmed Rezine, Huawei Li, Petru Eles, and Jianhui Jiang. "Software-based self-testing using bounded model checking for out-of-order superscalar processors." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, no. 3 (2019): 714-727.

[9] Jiang, Shujuan, Miao Zhang, Yanmei Zhang, Rongcun Wang, Qiao Yu, and Jacky Wai Keung. "An integration test order strategy to consider control coupling." *IEEE Transactions on Software Engineering* 47, no. 7 (2019): 1350-1367.

[10] Ghandehari, Laleh Sh, Yu Lei, Raghu Kacker, Richard Kuhn, Tao Xie, and David Kung. "A combinatorial testing-based approach to fault localization." *IEEE Transactions on Software Engineering* 46, no. 6 (2018): 616-645.

[11] Zhou, Zhi Quan, T. H. Tse, and Matt Witheridge. "Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors." *IEEE Transactions on Software Engineering* 47, no. 6 (2019): 1164-1183.

[12] Qiu, Kun, Zheng Zheng, Kishor S. Trivedi, and Beibei Yin. "Stress testing with influencing factors to accelerate data race software failures." *IEEE Transactions on Reliability* 69, no. 1 (2019): 3-21.

[13] Zolfagharian, Amirhossein, Manel Abdellatif, Lionel C. Briand, Mojtaba Bagherzadeh, and S. Ramesh. "A Search-Based Testing Approach for Deep Reinforcement Learning Agents." *IEEE Transactions on Software Engineering* (2023).

[14] Temple, Paul, Mathieu Acher, and Jean-Marc Jézéquel. "Empirical assessment of multimorphic testing." *IEEE Transactions*

on Software Engineering 47, no. 7 (2019): 1511-1527.

[15] Banerjee, Debdeep, Kevin Yu, and Garima Aggarwal. "Object Removal Software Test Automation." *IEEE Access* 8 (2020): 12967-12975.

[16] Banerjee, Debdeep, Kevin Yu, and Garima Aggarwal. "Object Removal Software Test Automation." *IEEE Access* 8 (2020): 12967-12975.

[17] Sánchez-Gómez, Nicolás, Jesus Torres-Valderrama, Julián Alberto García-García, Javier J. Gutiérrez, and M. J. Escalona. "Model-based software design and testing in blockchain smart contracts: A systematic literature review." *IEEE Access* 8 (2020): 164556-164569.

[18] Li, Nan, Qiang Han, Yangyang Zhang, Cong Li, Yu He, Haide Liu, and Zijian Mao. "Standardization Workflow Technology of Software Testing Processes and its Application to SRGM on RSA Timing Attack Tasks." *IEEE Access* 10 (2022): 82540-82559.

[19] Niu, Xintao, Huayao Wu, Changhai Nie, Yu Lei, and Xiaoyin Wang. "A theory of pending schemas in combinatorial testing." *IEEE Transactions on Software Engineering* 48, no. 10 (2021): 4119-4151.

[20] Sun, Zhe, Chi Hu, Chunlei Li, and Linbo Wu. "Domain ontology construction and evaluation for the entire process of software testing." *IEEE Access* 8 (2020): 205374-205385.

International Research Journal
IJNRD
Research Through Innovation