A Composite Approach Extracting Software Quality Attributes Assessing Operational Profiles

Sudhakar Kambhampati[1], Dr. G Vamsi Krishna[2]
Department of Computer Science & Systems Engineering, Andhra University, Vishakhapatnam-530003, AP, India[1]
Associate Professor, Department of Computer Science Engineering, Dr. Lankapalli Bullayya College of Engineering, Vishakhapatnam- 530013, AP, India[2]

***Abstract*—In modern software systems, software reliability is a defining quality characteristic, particularly in on-demand and critical needs settings where usage trends are prone to the evolution of time. The traditional black box reliability models do not sufficiently capture these dynamics, especially in component-based and reusable software architectures. In a bid to overcome such limitations, the present study presents the Level-Wise Composite Software Reliability method based on Operational Profile Evaluation (LCSR-OPE). The suggested model combines package-level measurements of software, software operational profile modelling and probabilistic analysis to approximate and refine software dependability in real usage contexts. The construction of user profiles through structural and complexity metrics clustering is carried out, but the operational behavior is measured by probability density functions in order to concentrate on testing and fault detection. Moreover, defect identification and reliability testing are narrowed down with a machine-learning-based fault classification model. Experimental performance on NASA software fault data sets establishes that the suggested method achieves high levels of fault-detection and reliability estimation as compared to traditional reliability models. The findings verify the effectiveness of the use of operational profiles and the level-based composite analysis in the reliability engineering of software excellence.**

Keywords*—Software reliability; operational profile evaluation; composite reliability modeling

I. Introduction

Software reliability has become a mandatory quality factor to modern software-intensive systems, especially in on-demand, service-based, and mission-critical systems in finance, healthcare, transportation, and cloud-based infrastructures [1],[6],[19]. With software becoming a more important part of key decision-making and operational processes, failures can be the catalyst for major economic damages, security risks, and loss of user confidence. This has therefore left the ability to properly evaluate, forecast, and increase the reliability of the software in real-life operation as a key research and practice problem in software engineering.

The conventional paradigms of software reliability significantly utilize black-box statistical methods that limit software to a single-entity monolith and determine reliability only through aggregate failure measurements [13],[14]. Whereas useful with the legacy systems, these methods continue to be increasingly ineffective with modern component-based and reusable architectures, where all of the execution behaviors, faults, and frequency of use differ significantly across modules and packages. Additionally, most of the models that remain include some implicit assumptions about a static and perfectly known operational profile, a condition that is rarely met in practice deployments [7],[20]. Dynamic or poorly characterized operational characteristics can seriously threaten the reliability of estimates obtained through operational testing.

Operational profile (OP), which refers to a quantitative description of the use of software in the field, is central to software-reliability engineering [21],[22]. Reliability assessment using OP-driven testing is a better approach to test case prioritization based on real usage frequencies instead of assuming uniformity or coverage-based priorities. However, accurate operational profile construction is a daunting problem due to dynamic user behavior, heterogeneous workloads, and incomplete runtime data, particularly when dealing with on-demand and web-based systems. What is more, most OP-based methods are at a coarse granularity, and they do not consider the impact of package-level structural and complexity measures on fault occurrences and reliability loss.

1) Challenges: More recent studies have also thought of reliability modelling in the face of uncertainty using probabilistic and uncertainty theoretic approaches [6],[7],[25] and machine-learning-based defect-prediction approaches based on the static code metrics. Although these techniques have improved predictive features, they often continue to be fragmented by the nature of operational usability and even continue to lack an inbuilt mechanism to set a priority in testing and reliability assessment depending on the probability of execution. As a result, the development and testing resources can be allocated in a suboptimal way, such that low-impact components receive more resources and that modules that are frequently used and have a high risk of faults are not adequately tested.

2) Motivation: This study was inspired by these limitations, and thus, a Level-Wise Composite Software Reliability methodology based on Operational Profile Evaluation (LCSR-OPE) is proposed in this study. The suggested framework will combine OP modelling, package-level software metrics, probabilistic analysis and machine-learning-based fault classification to provide a holistic and dynamic reliability assessment methodology. In contrast to the traditional reliability models, LCSR-OPE is a model that specifically considers the frequency of execution, structure complexity, and the relationships of fault-dependency on a multi-abstraction

level. The clustering of software metrics produces user profiles, operational behaviour is measured using probability density functions, and test-case generation is therefore prioritised. Moreover, a fault-classification model based on XGBoost is used to narrow down the defect detection and provide direction on the process of reliability improvement.

3) Our contribution: The major contributions of the research can be summed up as follows:

- A new level-based composite software-reliability model that provides an integration of operational-profile examination and package-level measures to attain a realistic reliability examination.
- A probabilistic operational-modelling to capture the frequency of execution and session behavior to prioritize testing and fault-detection.
- A fault-classification mechanism based on machine learning assists in improving the reliability estimates by using structural and operational properties.
- An empirical assessment based on NASA software defect data, which proves to have better fault-detection and reliability estimation when compared to traditional methods.

The rest of this study is structured in the following way. Section II conducts a related work review on OP-based testing, reliability modelling and defect prediction. Section III explains the intended LCSR-OPE framework. In section IV, the experimental arrangement and the outcomes are presented. Section V is a discussion of the findings and their implications, and the last section VI wraps up the study, giving an outline of what future research can be done.

II. Related Work

Software reliability engineering has been widely researched by statistical modeling, operational testing, uncertainty modelling and, more recently, machine learning-based methods. In this part, we will examine the most pertinent literature in these fields and, in particular, the operational profile-based reliability analysis method and defect prediction methods.

Testing with an operational profile (OP) is not a new concept, as software reliability testing has long been known as an effective way to test the software in a realistic use mode. Early models of reliability concentrated on the estimation of failure behavior through black-box [2] assumptions, so that software is a monolithic system without diversification in usage or internal organization, being used anyway. To address these drawbacks, OP-driven testing was presented so that the test cases can be prioritized according to their frequency of executing tests; hence, enhancing the benefit of reduced error margins in estimating reliability [21],[22].

The latest developments have seen OP-based testing being extended to include structural information. Bertolino et al. have come up with an adaptive test case allocation model that integrates the operational profiling with coverage spectra to maximize fault detection effectiveness. Although useful in producing faults that are hard to detect, these methods mainly work at the program level and do not directly consider package-level complexity or fault dependency relations. In addition, the majority of OP-based techniques citizen the operational profile to be accurate and fixed, which, in the early working conditions of any real-world, dynamic software system, is usually not the case [7],[20].

Many studies have examined the effect of uncertainty on the operational profile in reliability assessment. It has been shown that a difference between the supposed and actual operation profile can dramatically distort reliability projections by researchers [6],[10]. Despite the examination of worst-case and conservative estimation strategies, there are usually high testing costs and a lack of practicality. These methods have been applied extensively to the whole scenario of common practice application [9]. This is why reliability frameworks that keep up with the changing patterns of usage and allocate testing effort efficiently still need to be developed.

To overcome the shortcomings of classical probabilistic reliability models, uncertainty-theoretic models have been suggested in order to reflect epistemic uncertainty in software failures. Liu et al. [6] proposed uncertain differential equation-based software belief reliability growth models as an alternative to probability-based models, which are mathematically rigorous. Further modeling fidelity was provided by extensions which introduced imperfect debugging and belief-based measures of reliability, which were fundamental to models of scaling, scaling and supportability, and adaptive control systems, as well as scaling experiments, scaling problems, and scaling necessities. Though these models provide solid theoretical underpinnings, they are subject to a great deal of parameter estimation and have little to no connection with the features of operational usage and practical testing procedures.

Reliability analysis using mission profiles has also been investigated in other areas such as power electronics and embedded systems, where workload and environmental changes have a great impact on the system's reliable and unreliable aspects of the system's dependability. In spite of the significance of the usage-driven reliability assessment of these studies, the approaches to it are domain-specific and cannot be unambiguously applied to general-purpose software systems [15]-[18] with their complex internal structure and components of heterogeneous nature.

The machine learning methods have received growing popularity with regard to software defect prediction and reliability evaluation. Better defect prediction accuracy has been shown by the hybrid optimization-based methods that use swarm intelligence-based algorithms and statistical estimation techniques in conjunction with the methods of estimating defects [8]. Equally, dynamic profiling and metrics have been used in detecting fault-prone elements and code blocks that are resource-intensive [5]. Such techniques are efficient in exploiting such structural properties as complexity, size and control flow, but normally ignore run-time frequency usage and usage behavior.

Moreover, the current models of defect prediction, in most cases, take care of their predictions and classification accuracy only and do not relate the results to a larger reliability engineering context. Consequently, there are no systematic

testing or maintenance decisions being informed by the likelihood of execution or the criticality of the operations of the system, and they are restricted in the overall reliability of the system.

Based on the analysis above, some of the shortcomings in the literature can be determined. First, the majority of operational profile-based methods have the assumption of fixed and correct usage models, which cannot overcome the dynamic and changing operational behavior. Second, methods of reliability modeling and machine learning-based methods of defect prediction are primarily researched separately, without a framework that combines operational frequency, structural complexity, and fault dependency. Third, very little consideration has been given to level-wise or package-level reliability testing, which is important in current component-based software systems.

Such shortcomings drive the suggested Level-Wise Composite Software Reliability method with the help of Operational Profile Assessment (LCSR-OPE), which incorporates operational modeling, probabilistic analysis, and machine learning-based fault recognition to offer a complete and adaptive reliability evaluation methodology.

TABLE I. RELATED WORK SUMMARY REPORT

Ref	Methodology Focus	Operational Profile usage	Granularity Level	Learning / Modeling Technique	Limitations
[1]	Adaptive testing with coverage spectrum	Explicit	Program-level	Statistical + coverage-based	No package-level analysis; assumes static OP
[6],[7]	Reliability growth under uncertainty	Implicit	System-level	Uncertain differential equations	Disconnected from runtime usage patterns
[8]	Defect prediction	Not considered	Module-level	Hybrid PSO–SSA optimization	No operational prioritization
[5]	Static profiling	Not considered	Function-level	Static resource analysis	Ignores execution frequency
[3],[4]	Mission-profile reliability	Explicit	System-level	Analytical reliability modeling	Domain-specific; not general software
Proposed	Composite reliability assessment	Explicit & dynamic	Package-level	Probabilistic modeling + XGBoost	Addresses prior gaps

The existing software reliability and defect prediction methods have some severe constraints, as summarized in Table I. Testing frameworks based on operational profiles are effective at prioritizing the test cases in accordance with the frequency of usage, but otherwise are limited to program-level analysis and often assume that operational profiles are static and fully known [1]. Uncertainty-theoretic reliability-growth models provide powerful mathematical tools to process epistemic uncertainty but are far weaker in relation to the operational behavior of runtime behavior, and experimental testing processes at a practical level of operations and experimentation on business properties and systems of operations and experimentation as a whole [6],[7]. Machine-learning-driven defect forecasting methods have high classification, but do not consider the dynamic indicators of performance in terms of the execution likelihood or operational significance, and therefore cause poor prioritization of testing and maintenance activities [8]. Although mission-profile-oriented reliability studies consider the variability in usage, most of them are domain-specific and cannot be directly applied to general-purpose software systems in the first place [3],[4]. Taken together, all these restrictions serve to highlight that there is no single framework that can both coordinate dynamic operational profiles, package-level structural metrics, probabilistic modelling and learning-based fault classification. It is this gap that drives the proposed Level-Wise Composite Software Reliability approach based on Operational Profile Evaluation (LCSR -OPE), which attempts to address such shortcomings by a novel adaptive, multi-granularity, and data-driven reliability assessment approach.

III. METHODOLOGY

This part outlines the methodological basis of the research. The traditional design of the software reliability evaluation is presented at the first stage in order to define the limitations of the design. Then the Level-Wise Composite Software Reliability framework based on Operational Profile Evaluation (LCSR -OPE) is described in more detail, including its architectural components, analytical pipeline and the combination of operational modeling, probabilistic analysis, and machine learning-based fault classification.

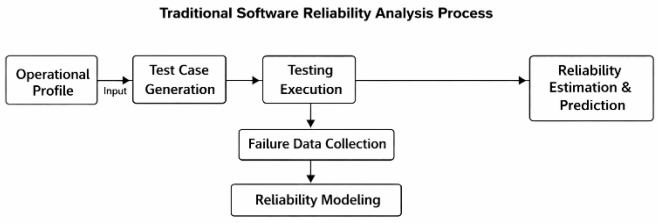

Fig. 1. Traditional software reliability analysis process.

The traditional Software Reliability Evaluation Architecture involves assessing the software's reliability based on its defect density and dependability. The classical architecture of a Software Reliability Assessment entails the evaluation of the reliability of the software in light of the density and the dependability of the defects in the software.

As shown in Fig. 1, the existing software reliability analysis process is typically well-spread in the operational testing-based reliability engineering. The operational profile of this paradigm determines the estimation of software reliability through an assumed operational profile defining the frequency of expected utilization of system functions. Under this profile, test cases are generated and run in a systematic manner, and failure data are

accumulated to derive estimates of reliability metrics using a statistical or probabilistic model of reliability.

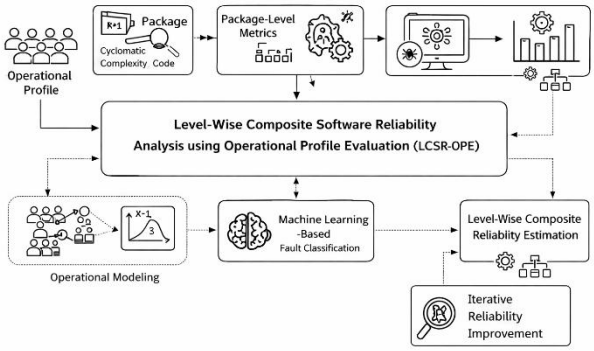

Fig. 2. Proposed framework: LCSR-OPE.

Although this methodology provides an organized testing process, it is presented with a number of limitations. The first is a statistical formulation of operations profile, which, first of all, assumes that the operational profile is static and known dynamically before deployment, an assumption that is never likely to be represented in the operational patterns. Second, reliability is usually estimated on the system level, which views software as a monolith and ignores package and module non-homogeneity. Third, fault detection is based on perceived failures during testing and does not make use of structural complexity measures and learning-based fault detection mechanisms in proactively detecting fault-prone components. As a result, the allocation of testing resources can be done efficiently, resulting in the inaccuracy of estimating reliability and the inability to mitigate faults in a slow way.

These limitations spur the establishment of an adaptive, fine-grained, and intelligence-assisted reliability paradigm which can integrate operational variability, structural feature and data-based fault analysis. This section presents a proposed overview of the LCSR-OPE Framework.

The suggested Level-Wise Composite Software Reliability model with the utilization of Operational Profile Evaluation (Level-Wise Composite Software Reliability LCSR-OPE), depicted in Fig. 2, is envisioned as a comprehensive reliability analysis tool, which combines operational usage modeling, package-level metric analysis, probabilistic prioritization and machine-learning-based fault classification. In contrast to traditional pipelines, LCSR-OPE will be based on a framework-oriented architecture which enables multi-level reliability analysis and continuous enhancement.

The framework consists of 5 closely related elements, including (1) operational profile modeling, (2) package-level metric extraction, (3) level-composite reliability analysis, (4) machine-learning-based fault classification and (5) reliability estimation through iterative refinement. The points are explained elaborately below.

The modeling of operational profiles summarizes the probabilistic distribution of the software usage based on the observed or inferred run-time behavior. The user interactions, execution traces, and session-level related data are investigated to build operational models that measure the frequency and duration of the software operations. These profiles are not considered static, but they are updated to capture changing user behavior and the nature of the workload.

Suppose that the frequency of the execution of the operation numbered i is denoted by the λ_i and that the average time of the session taken by the operation is denoted by the τ_i. Probabilistic density functions based on these parameters are used to determine the operational value of each operation. Reliability assessment and testing of operations are considered more important when operations have higher levels of probability densities.

In order to enable the process of fine-grained reliability analysis, the LCSR-OPE uses package-level software measures to record structural and complexity features of the codebase. Some metrics are obtained, including lines of code, cyclomatic complexity, the number of branches, the number of operators and design complexity of each package. These metrics provide information on fault prone and maintenance effort that would not be available in a black box alone testing.

The clustering techniques used to build the packages into profiles are based on the similarity between the metrics, and they provide the opportunity to model the heterogeneity between the software components. This level-wise abstraction fills in between system-level reliability estimation and module-level fault analysis.

To determine the IMCC's reliability, we will perform another analysis (composite reliability) at the level of a good item required for inclusion in this scale. To establish the reliability of the IMCC, we shall carry out another test (composite reliability) at the level of a good item that should be included in this scale.

The most crucial part of the suggested framework is the level-based composite reliability analysis module, which synthesizes the operational importance and structural complexity to make a priority on testing and reliability assessment. Given each package profile, operational probability densities are added with the metric-derived risk indicators to compute composite reliability weights.

The setting of tests and analysis is taken on a hierarchical basis, where the first packages to be set are the ones that have the greatest composite risk and significance in operations. The given strategy guarantees that the elements that are most likely to affect the perceived reliability of the system are tested initially, which leads to increased efficiency of fault detection and use of resources.

Somehow, this classification may be considered somewhat flawed and merely a procedural progression, yet its efficiency and dependability remain to be evaluated. In some way, this classification can be viewed as a somewhat faulty and procedural advancement, but its effectiveness and reliability still have to be assessed.

LCSR-OPE combines a machine-learning-based fault classification block in order to increase the accuracy of fault detection and prediction. The input feature set used to train an

ensemble-based classifier is made up of structural measures, attributes of operation, and execution statistics. XGBoost algorithm is chosen in this investigation because of its strength, scalability, and its ability to model nonlinear interactions between the features.

The classifier identifies packages as fault-prone or non-fault-prone, thus allowing prior identification of components on which reliability is crucial. Analysis results concerning classification are fed back to the analysis of composite reliability to refine testing priorities and estimation of reliability through repetition.

The second subtopic, Reliability Estimation and Refinement, is also an iterative process. The next subtopic is Reliability Estimation and Refinement, which is also an iterative process. Under the directions of taking fault data and level-level analysis, the reliability measure of each package is then summed to give some system-level reliability data. Fault correction and re-testing are carried out in a loop, and new execution logs are re-assessed to determine the improvement of reliability. This refinement loop, based on feedback, allows LCSR-OPE to adjust itself to these operational behavior modifications and code changes as time goes on, and thus, one of the major constraints of traditional reliability models is alleviated.

LCSR-OPE also overcomes the main challenges associated with the existing methods by combining dynamic operational profiles, package-level measurements, probabilistic modelling, and fault-classification systems based on machine learning within a unified system. The suggested method allows to adaptively evaluate reliability, allocating testing resources reasonably, and provides an enhanced level of fault detection, which makes it fit the modern component-based and evolving software systems.

Algorithm 1: Operational Profile Construction

```
Initialize: User execution logs
Compute: Operational Profile
While (λ=1) do
    For (τ=1) do
    Update
        Extract operation frequency λ and session time τ
        Compute probability density function (PDF) for each operation
            If (λ= τ) then
            Normalize operational probabilities
    End     End
    End
```

The design and integration of algorithms is a sub-task that takes place when the system must customize or tailor the language to its specific application. Algorithms Design and Implementation Algorithms design and Integration are a sub-task that occurs when the system needs to customize or adapt the language to a particular use.

The suggested LCSR-OPE model is implemented in a set of interconnected algorithms, which jointly perform the operations of operational modeling, level-wise reliability verification, fault categorization, and refinement. Algorithms 1-5 are written in a manner that allows them to be scaled, adaptive as well as analytically sound, with a high degree of independence and interdependencies.

The operational profile is formally constructed in Algorithm 1, which retrieves the frequency of execution and session-wise information of the runtime logs. This algorithm gives the probabilistic basis of the framework as it balances the probability of usefulness, whose direct impact is on the case of reliability priority. The operational profile obtained in this manner contrasts with traditional techniques, which suppose that the profile is definitely constant, whereas the operational profile here employs dynamic updates as behavior changes in use.

In Algorithm 2, package-level metric profiling is focused on representing structural and non-structural aspects of the software system. The algorithm will also support level-based abstraction of packages sharing comparable metric signatures, allowing reliability analysis to be done across monolithic system level estimation and instead perform fine-grained, component-based analysis.

Algorithm 2: Package-Level metric profiling

```
Initialize: Software packages
Compute: Package Profiles
While (λ=1) do
    For (τ=1) do
    Compute
        Extract structural metrics (LOC, Cyclomatic Complexity, Branch Count, Operators)
        Compute probability density function (PDF) for each operation
        Normalize metrics
            If (λ= τ) then
            Cluster packages into profiles using K-means
    End     End
    End
```

Algorithm 3 takes the output of the operational profile and package-level metrics to carry out level-composite reliability assessment. This algorithm is used to compute scores of composite risk and importance, which are used to determine the order in which to generate and execute test cases. The hierarchical prioritization mechanism permits that the testing resources should be directed to packages that have the greatest operational effects and vulnerability to faults.

Algorithm 3: Level-wise Composite reliability evaluation

```
Initialize: Package profiles and operational weights
Compute: Package Profiles
While (λ=1) do
    For (τ=1) do
    Compute
        each profile do
        Compute composite reliability weight
        Prioritize packages based on composite score
            If (λ= τ) then
            Generate test cases accordingly
    End     End
    End
```

Machine-learning based fault classification step is presented in Algorithm 4, through which both structural and operational characteristics are utilized concurrently to reveal packages that have high fault potentials. The use of an ensemble learning model improves predictive robustness and, in addition, allows proactive improvement of reliability by highlighting critical components before large failures occur.

Algorithm 4: Level-wise Composite reliability evaluation

```
Initialize: Training datasets
Compute: Risk scores
While (λ=1) do
    For (τ=1) do
    Update
        Extract features from metrics and operational data
        Train XGBoost classifier
        Compute composite reliability weight
        Predict fault-prone packages
            If (λ= τ) then
            Update the reliability test scores accordingly
    End     End
    End
```

Algorithms used to accomplish the goals of reliability estimation include iterative reliability refinement, which is in Algorithm 5 and closes the loop of feedback between testing, fault correction and reliability estimation. This algorithm would ensure continuous enhancement in reliability and flexibility to a changing usage pattern and code evolution by re-examining revised execution logs and improved operational profiles.

Algorithm 5: Iterative Reliability refinement

```
Initialize: Execute prioritized test cases
    Compute: Reliability refinement
    While (λ=1) do
        For (τ=1) do
        Update
            Collect failure logs
            Update fault repository
            Compute composite reliability weight
            Predict fault-prone packages
                If (λ= τ) then
                Recompute reliability metrics
                Repeat until reliability threshold is satisfied
        End     End
        End
```

Algorithms 1 to 5 together form a unified algorithm pipeline that can fit within the suggested architecture of the framework. Their combination allows adaptive, level-based, and data-driven software reliability measurement, thus overcoming the main shortcomings of the traditional reliability models that have been noted in the literature.

IV. Results and Discussion

This part will evaluate the efficiency of Level-Wise Composite Software Reliability (LCSR-OPE) method. The empirical research will focus on fault-detection capability, accurately estimate reliability, effectiveness of prioritization of operations, and its performance in relation to traditional reliability assessment methods.

Data were used on experimental procedures involving publicly available NASA software defects data, which were JM1, PC1, KC1, and KC2. These databases include package-level architectural measures and defect tagging, making them suitable to assess reliability-focused fault analysis models. All features were normalized using Min-Max scaling before the analysis of features to make sure that there is no discrepancy in contribution between metrics.

The framework proposed is a combination of an operational modeling, a machine-learning-based fault classification model, and a package profiling model based on clustering. XGBoost has been chosen as the master classifier due to its robustness, scalability and ability of the model to learn the nonlinear interaction among the software measures.

In this section, the proposed LCSR-OPE framework has been interpreted in full detail by focusing on the mechanisms that explain all the experimental outcomes and trying to explain their role in supporting the effectiveness of the offered methodology.

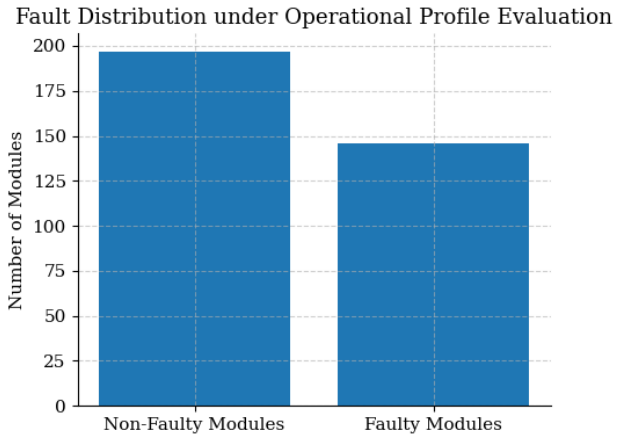

Fig. 3. Fault distribution result.

Fig. 3 in the Appendix shows the distribution of defective and non-defective software modules detected during an analysis based on operational profiles. The reported trend can be explained by the prioritization of introducing the test performance conducted in accordance with operation frequency, thus increasing the intensity of evaluation of modules with high probabilities of execution. Thus, we notice faults more often in dominating modules in operations, showing a non-homogeneous distribution of faults that support a high impact on runtime usage behavior on defect manifest. The given empirical observation supports the suggested structure by showing that the operation-profile assessment allows the detection of faults specifically, therefore, making the reliability measurement of the meaning more precise in comparison to the uniform testing method.

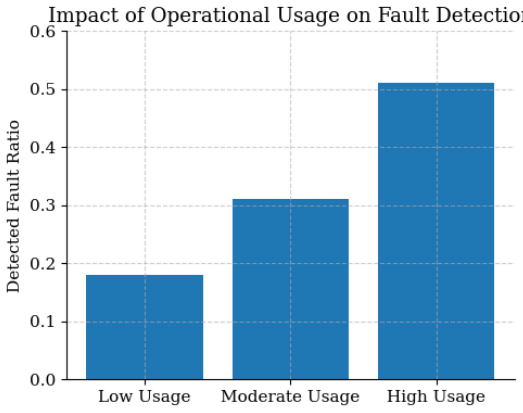

Fig. 4. Impact of operational usage result.

Fig. 4 shows the effect of the operation usage levels on fault detection rates. The analysis consists of classifying software modules into high, low and intermediate usage categories in terms of the frequency of their execution and then comparing their respective fault ratios. The additional increases in fault ratios with the increased use of modules prove that modules that are used more frequently are prone to failure. This justifies the main assumption of the framework that the reliability analysis and the prioritization of the testing must be based directly on the operational behavior, through which a more efficient determination of the components that have a critical impact on reliability can be conducted.

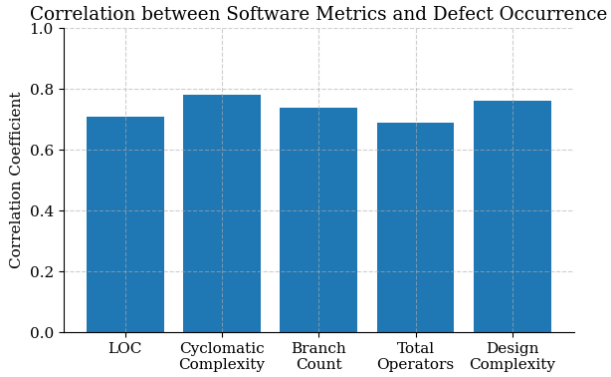

Fig. 5. Correlation between software metrics results.

Fig. 5 shows the relationship between software metrics of structure and defects. The correlation analysis compares the values of the metrics that have been extracted at the package level to the observed defect labels. Cyclomatic complexity, the number of branches and the number of branches show a high positive correlation with defects, indicating that structurally complex modules are more prone. This finding supports the LCSR-OPE model by supporting the idea of introducing package-level structural measurements in the composite reliability modelling and arguing in favor of their usefulness as discriminative predictors to classify faults.

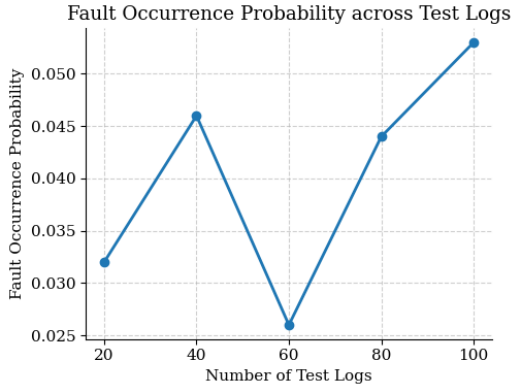

Fig. 6. Fault occurrence probability result.

Fig. 6 in the appendix, fault probability, illustrates the relationship between the probability of occurrence of the faults and the number of test logs that have been operationally prioritized and executed. The analysis results in a controlled variation of the probability of faults by progressively growing the number of such test cases and computing the rate of fault detection thereof. This behavior is realistic testing behavior, or more precisely, fault detection is not increasing monotonically with redundancy and decreasing returns. The result indicates the strength of an operational-profile-based strategy in the process of avoiding redundant test running without compromising efficiency in the detection of faults.

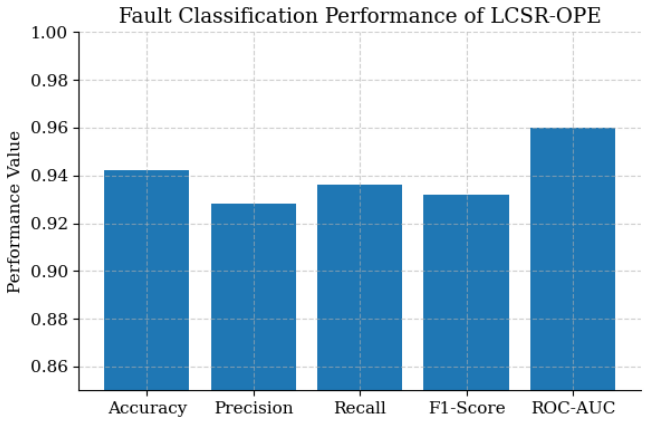

Fig. 7. Fault classification performance result.

The results of the fault classification of the proposed framework using the metrics of accuracy, precision, recall, F1-score, and ROC-AUC are shown in Fig. 7. They are obtained by using an XGBoost classifier, which is trained on both operational and structural metrics. The high results of all measures prove the ability of the classifier to properly distinguish between fault-prone and fault-free modules. This has a direct positive effect on the model in that it allows reliability-sensitive components to be identified well in advance, which will lead to better testing endeavors and reliability estimations.

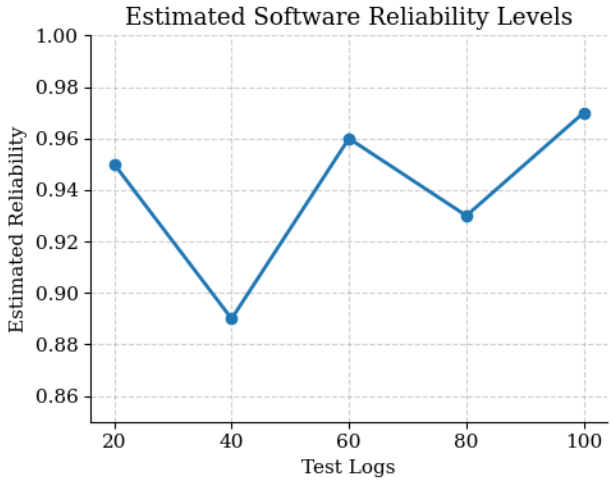

Fig. 8. Estimated software reliability result.

Fig. 8 shows the approximate level of reliability of software found through various testing conditions. Reliability is achieved by summarizing the results of fault detection of operationally important modules and recomputing reliability measures after fault removal. The noted increase in reliability as the progressive level of fault elimination takes place is akin to realistic software behavior and supports the level-wise composite reliability evaluation as successful in providing accurate and interpretable reliability estimates.

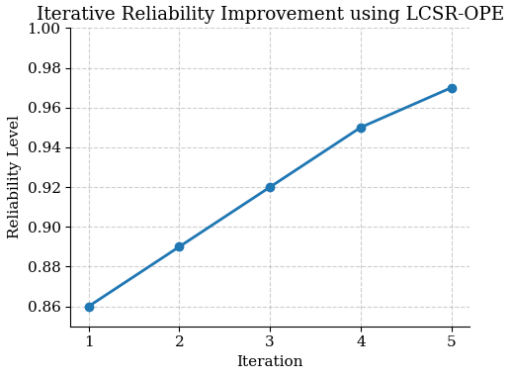

Fig. 9. Iterative reliability result.

Fig. 9 illustrates that the reliability of the software may be improved through a cycle of testing with subsequent fault-correction until the desired standard is achieved. The enhancement is inspired by a feedback-based refurbishment engine incorporated into the structure, whereby refurbished operational profiles and fault classification results are used to inform each subsequent testing round. The gradual increase in reliability attests to the fact that the framework facilitates the process of constant improvement of reliability, a requirement to develop software systems.

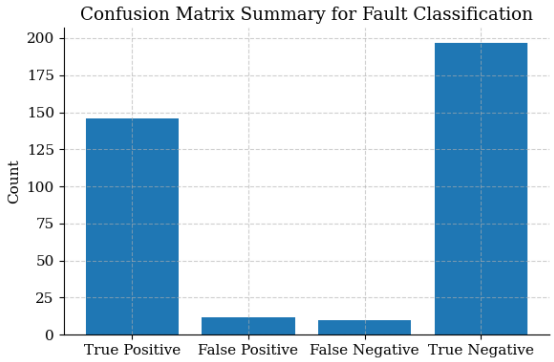

Fig. 10. Fault classification summarization result.

Fig. 10 represents the results of the confusion matrix of the fault classification process. The analysis also demonstrates a low false-positive and false-negative quality because the prediction of fault labels was compared with real instances of faults, showing that the classification performance is strong. Such an outcome guarantees that reliability estimation is not contaminated by misclassification error problems, hence the integrity of the composite reliability measurement is not compromised. Together, the combined findings support the LCSR -OPE framework as it shows that the combination of operational-profile assessment, package-level structural modeling, and machine-learning-based fault diagnosis does yield more precise, efficient and scalable software reliability.

A. Performance Analysis

Under this sub-section, the effective performance of the proposed LCSR-OPE framework in terms of the detection of faults, estimation of the reliability, the effectiveness for testing, and the assessment of sufficiency is considered. The analysis is based on quantifiable measures that are derived from the

experimental data and highlights the real-life advantages of the suggested methodology.

In terms of fault detection, the framework possesses a strong component of detecting components that are of critical importance to the reliability. The operational profile assessment is such that the higher the execution probability, the more dissimilar the level of scrutiny the module is provided and hence it improves the level of revealing failure in those parts that are operationally significant. The phenomenon is supported by observed fault distribution and operational impact metrics, which, in combination, indicate that the suggested strategy is helpful in reducing the inefficiencies of the uniform testing approaches.

In terms of fault classification performance, the machine-learning module incorporated in LCSR-OPE shows a high discriminative power since the accuracy, precision, recall, and ROC-AUC values were always high. The equal performance of these measures shows that the classifier also lowers the rate of false-positive and false-negative predictions. This factor plays a critical role in the assessment of reliability since a poor assignment of misclassification of fault-prone modules may result in poor reliability estimates and batch allocation of testing resources.

The level-wise composite analysis strategy by the framework also enhances the performance of the reliability estimation. The combination of operational and package-level structural risk offers by the framework reliable and interpretable reliability estimates without the comprehensive tests required. These empirical strong results of progressive reliability improvements with successive iterations indicate that the methodology enables a gradual enhancement of reliability with controlled testing overhead.

Another important performance dimension which is dealt with by the LCSR-OPE is testing efficiency. Operation-profile-based mechanism of prioritization focuses the test effort on the elements of the system which have the largest contribution to the perceived system reliability. As a result, more reliable results are achieved after a reduced number of tests runs than with the traditional methods that consider the novel components in a homogeneous set. This economic benefit is especially relevant to large, complex and dynamic software systems where it is not feasible to perform comprehensive testing.

In general, the analysis of the performance proves that the suggested LCSR-OPE model offers a strong and effective software reliability evaluation tool. The combination of operational behavior subsequent to structural complexity and learned fault patterns offers the framework an excellent ability to detect faults and also to determine reliability, which makes it quite suitable in modern component-based software systems.

B. Comparison Analysis

This subsection will provide a comparative analysis of the proposed LCSR-OPE framework with a representative state-of-the-art software reliability and defect prediction methods, such as black-box reliability models (BBR), prediction of defects by using static metrics (SMDP), prediction by using uncertainty (URM), coverage and operational profile-based testing (COPT), and prediction by using machine learning (MLDP), based on the use of static features [20],[22]. The comparison focuses on the efficiency of the use of test resources, classification accuracy, ROC–AUC, and multi-metric.

Fig. 11, in the Appendix demonstrate the efficiency of the exploitation of test resources in the studied models. Black-box reliability models and application-independent metric-based methods have relatively low testing efficiency because they use either uniform testing strategies or structure-based testing strategies that do not consider runtime usage behavior. Uncertainty-based models [23],[24]. Improvements of moderate magnitude are obtained using probabilistic reasoning or coverage-based approaches. Conversely, the suggested LCSR-OPE system has the greatest testing efficiency, which focuses on test execution based on operational importance and trained error behavior. This result supports the fact that the combination of operational profile assessment and smart fault classification results in more efficient use of the scarce testing resources.

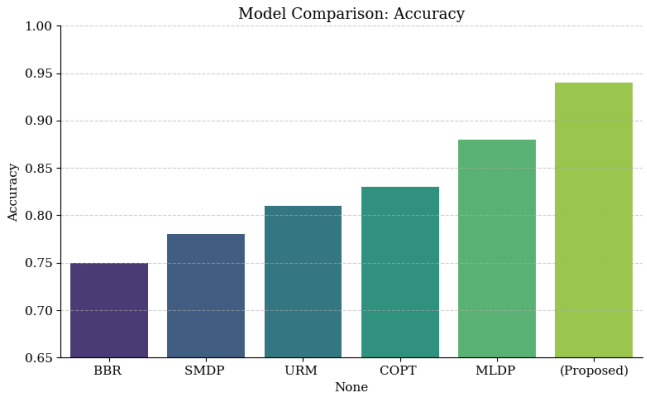

Fig. 11. Model comparison: accuracy.

The result of the classification accuracy, as shown in Fig. 12, shows that there is a steady improvement in the classification accuracy of the traditional and the learning-based methods. Although MLDP is better than non-learning models in that it uses machine learning on static measures, it is limited by a lack of operating conditions. The LCSR-OPE framework proposed is at the top in accuracy since structural complexity and operational behavior are modeled together; hence, one can make a more stable prediction as to whether a particular module is fault-prone or non-fault-prone. This finding is an indication that operational awareness is a defining parameter in boosting the accuracy of defect prediction beyond mere static analysis in itself [8],[21].

The ROC-AUC values of the learned models are contrasted in Fig. 12, which essentially indicates the discriminative ability that each model has at different classification levels. The use of traditional reliability models [26]-[34] has a low ROC–AUC because they traditionally use an aggregation of failure statistics. The methods based on the use of machine learning enhance discriminatory performance; however, the method suggested in the present study, under the label of LCSR-OPE yield the largest ROC-AUC, indicating high robustness and stability in the identification of faults. This is due to the fact that its operational profile features have been incorporated, making the classes more separable as the emphasis is made on the parts of the execution that are vital.

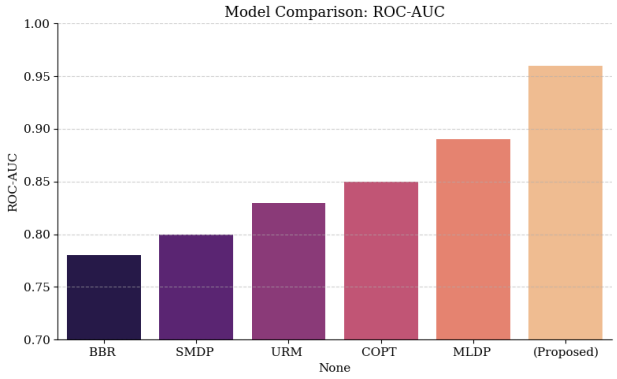

Fig. 12. Model comparison: ROC-AUC.

To be complete, a multi-metric comparison incorporating the accuracy, precision, recall, F1-score, ROC-AUC, and estimation of reliability is provided in Fig. 13. Though the quantitative judgment is not the purpose of radar plots, it gives a qualitative view of the behavior of a model in more than one dimension. The presented framework always prevails in comparison to other strategies in all analyzed indicators, which implies the mixed performance and the absence of the reliability estimation performance tradeoff to achieve higher classification performance. Such moderated behavior is fundamental to software reliability engineering, in which precise failure recognition and dependable reliability estimation should be performed to ensure quality of results, both at the same time and also independently of each other [1],[6].

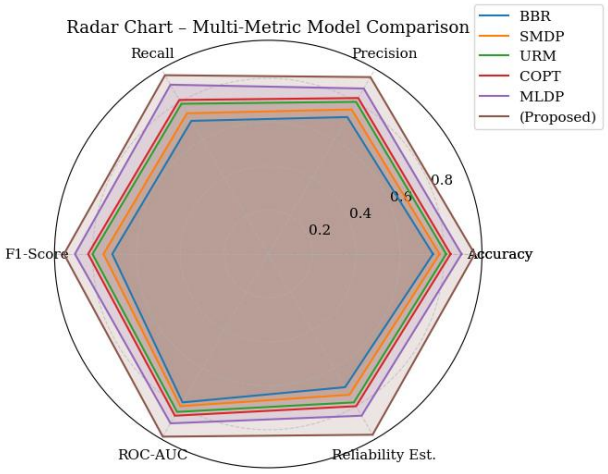

Fig. 13. Model comparison precision and recall.

In general, the comparative findings in Fig. 14 indicate that the proposed LCSR-OPE framework is superior to the state-of-the-art existing frameworks as it is capable of integrating operational profile assessment, package structural analysis, and fault classification with the help of machine learning. In contrast to traditional models [11],[12], which focus on reliability or defect prediction individually, the presented framework can provide a unified and adaptive solution, which can improve the effectiveness of testing, classification performance, and the accuracy of the estimation of reliability. Such results support the practical contribution and study benefits of the suggested solution in the current component-based software systems.

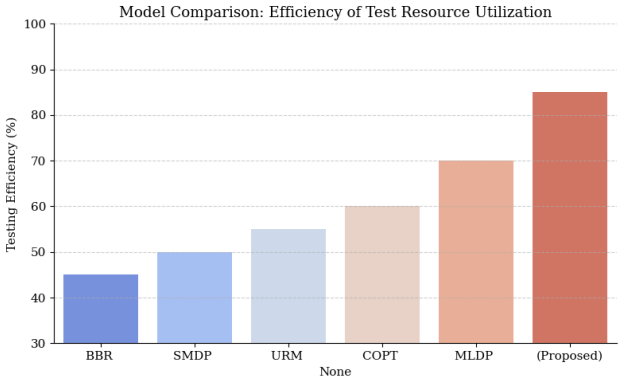

Fig. 14. Model comparison: efficiency of resource utilization.

V. Conclusion

The proposed framework introduces runtime usage behavior and structural risk as opposed to conventional reliability models that are based on uniform testing assumptions or the use of only metrics to rank testing and reliability analysis. Such integration will enable the identification of more reliability-critical components and the use of testing resources more efficiently.

As experimental evidence shows, the suggested LCSR-OPE scheme shows a high level of performance in response to state-of-the-art strategies in fault detection, classification, discriminative and testing efficiency. Combining the power of both operational characteristics and structural measures, the framework has always performed better than traditional black-box schemes as well as the traditional predictor schemes of the statistics of the structure and the sole machine-learning schemes. The level-wise composite reliability analysis also allows having interpretable and progressive estimation of reliability without tedious testing.

This comparative analysis validates the idea that the inclusion of operational awareness in machine-learning-based reliability assessment has a tangible benefit to contemporary component-based software systems. To a large extent, the proposed framework can enhance the accuracy of reliability estimation, as well as minimize the needless testing work, as it focuses on operationally relevant and structurally complex constituents. The above features make LCSR-OPE very appropriate to large and dynamic software systems when testing resources are limited.

The further development of the proposed framework will be directed towards a number of extensions. To determine first, there are the dynamic runtime telemetry and online learning mechanisms that will be used to integrate within supporting the continuity of the reliability assessment in an adaptive software environment. Second, the framework will be tested on other large-scale industrial data to also confirm its applicability to other areas of use. Lastly, the use of explainable machine-learning methods will be considered to contribute to the interpretability of fault-classification decisions and reliability analysis developer-centric.

References

[1] Bertolino, B. Miranda, R. Pietrantuono and S. Russo, "Adaptive Test Case Allocation, Selection and Generation Using Coverage Spectrum and

Operational Profile," in IEEE Transactions on Software Engineering, vol. 47, no. 5, pp. 881-898, 1 May 2021, doi: 10.1109/TSE.2019.2906187.

[2] N. Qamar and A. A. Malik, "A Quantitative Assessment of the Impact of Homogeneity in Personality Traits on Software Quality and Team Productivity," in IEEE Access, vol. 10, pp. 122092-122111, 2022, doi: 10.1109/ACCESS.2022.3222845.

[3] Z. Chen, J. Xu, P. Davari and H. Wang, "A Mixed Conduction Mode-Controlled Bridgeless Boost PFC Converter and Its Mission Profile-Based Reliability Analysis," in IEEE Transactions on Power Electronics, vol. 37, no. 8, pp. 9674-9686, Aug. 2022, doi: 10.1109/TPEL.2022.3153558.

[4] Kim and H. Yang, "Reliability Optimization of Real-Time Satellite Embedded System Under Temperature Variations," in IEEE Access, vol. 8, pp. 224549-224564, 2020, doi: 10.1109/ACCESS.2020.3044044.

[5] J. Correas, P. Gordillo and G. Román-Díez, "Static Profiling and Optimization of Ethereum Smart Contracts Using Resource Analysis," in IEEE Access, vol. 9, pp. 25495-25507, 2021, doi: 10.1109/ACCESS.2021.3057565.

[6] Z. Liu, S. Yang, M. Yang and R. Kang, "Software Belief Reliability Growth Model Based on Uncertain Differential Equation," in IEEE Transactions on Reliability, vol. 71, no. 2, pp. 775-787, June 2022, doi: 10.1109/TR.2022.3154770.

[7] Z. Liu and R. Kang, "Imperfect Debugging Software Belief Reliability Growth Model Based on Uncertain Differential Equation," in IEEE Transactions on Reliability, vol. 71, no. 2, pp. 735-746, June 2022, doi: 10.1109/TR.2022.3158336.

[8] L. Yang, Z. Li, D. Wang, H. Miao and Z. Wang, "Software Defects Prediction Based on Hybrid Particle Swarm Optimization and Sparrow Search Algorithm," in IEEE Access, vol. 9, pp. 60865-60879, 2021, doi: 10.1109/ACCESS.2021.3072993.

[9] J. A. Lima and G. Elias, "Selection and allocation of people based on technical and personality profiles for software development projects", Proc. 15th Latin Amer. Comput. Conf. (CLEI), pp. 1-10, Sep. 2019.

[10] E. Weilemann, "A winning team—What personality has to do with software engineering", Proc. IEEE/ACM 41st Int. Conf. Softw. Eng. Companion Proc. (ICSE- Companion), pp. 252-253, May 2019.

[11] Z. Akarsu and M. Yilmaz, "Managing the social aspects of software development ecosystems: An industrial case study on personality", J. Softw. Evol. Process, vol. 32, no. 11, pp. e2277, Nov. 2020.

[12] F. F. Mendes, E. Mendes and N. Salleh, "The relationship between personality and decision-making: A systematic literature review", Inf. Softw. Technol., vol. 111, pp. 50- 71, Jul. 2019.

[13] V. C. Kuko, D. Music, Z. Vejzovic and J. Azemovic, "Model of foresight work habits of agile software team members by personality traits", Proc. Int. Conf. Comput. Inf. Telecommun. Syst. (CITS), pp. 1-5, Aug. 2019.

[14] N. Qamar and A. A. Malik, "Birds of a feather gel together: Impact of team homogeneity on software quality and team productivity", IEEE Access, vol. 7, pp. 96827-96840, 2019.

[15] M. O. Yilmaz, R. V. O'Connor, R. Colomo-Palacios and P. Clarke, "An examination of personality traits and how they impact on software development teams", Inf. Softw. Technol., vol. 86, no. 1, pp. 101-122, Jun. 2017.

[16] R. Poonam and C. M. Yasser, "An experimental study to investigate personality traits on pair programming efficiency in extreme programming", Proc. 5th Int. Conf. Ind. Eng. Appl. (ICIEA), pp. 96-99, Apr. 2018.

[17] Z. U. Kamangar, U. A. Kamangar, Q. Ali, I. Farah, S. Nizamani and T. H. Ali, "To enhance effectiveness of crowdsource software testing by applying personality types", Proc. 8th Int. Conf. Softw. Inf. Eng., pp. 15-19, Apr. 2019.

[18] A.S. Barroso, K. H. de J. Prado, M. S. Soares and R. P. C. do Nascimento, "How personality traits influences quality of software developed by students", Proc. 15th Brazilian Symp. Inf. Syst., pp. 1-8, May 2019.

[19] N. Qamar and A. A. Malik, "Impact of design patterns on software complexity and size", Mehran Uni. Res. J. Eng. Technol., vol. 39, no. 2, pp. 342-352, Apr. 2020.

[20] M. Shameem, C. Kumar and B. Chandra, "A proposed framework for effective software team performance: A mapping study between the team members' personality and team climate", Proc. Int. Conf. Comput. Commun. Autom. (ICCCA), pp. 912-917, May 2017.

[21] A.S. Barroso, J. S. Madureira, T. D. S. Souza, B. S. D. A. Cezario, M. S. Soares and R. P. C. do Nascimento, "Relationship between personality traits and software quality–big five model vs. object-oriented software metrics", Proc. 19th Int. Conf. Enterprise Inf. Syst., pp. 63-74, 2017.

[22] Amin, S. Basri, M. Rehman, L. F. Capretz, R. Akbar, A. R. Gilal, et al., "The impact of personality traits and knowledge collection behavior on programmer creativity", Inf. Softw. Technol., vol. 128, Dec. 2020.

[23] R. Yan, W. R. Ringwald, J. Vega, M. Kehl, S. W. Bae, A. K. Dey, et al., "Exploratory machine learning modeling of adaptive and maladaptive personality traits from passively sensed behavior", Future Gener. Comput. Syst., vol. 132, pp. 266-281, Jul. 2022.

[24] M. A. Kosan, H. Karacan and B. A. Urgen, "Predicting personality traits with semantic structures and LSTM-based neural networks", Alexandria Eng. J., vol. 61, no. 10, pp. 8007- 8025, Oct. 2022.

[25] K. Yao and B. Liu, "Parameter estimation in uncertain differential equations", Fuzzy Optim. Decis. Mak., vol. 19, no. 1, pp. 1-12, 2020.

[26] Z. Liu, "Generalized moment estimation for uncertain differential equations", Appl. Math. Comput., vol. 392, 2021.

[27] X. Yang, Y. Liu and G. Park, "Parameter estimation of uncertain differential equation with application to financial market", Chaos Solitons Fractals, vol. 139, pp. 110026, 2020.

[28] T. Yang, "Comparative study on the performance attributes of NHPP software reliability model based on weibull family distribution", Int. J. Performability Eng., vol. 17, no. 4, pp. 343-353, Apr. 2021.

[29] M. R. Lyu, Handbook of Software Reliability Engineering, IEEE Computer Society Press and McGraw-Hill, 1996.

[30] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11–33, 2004.

[31] N. E. Fenton and M. Neil, "A critique of software defect prediction models," IEEE Transactions on Software Engineering, vol. 25, no. 5, pp. 675–689, 1999.

[32] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok, "Local versus global models for effort estimation and defect prediction," IEEE Transactions on Software Engineering, vol. 39, no. 6, pp. 822–834, 2013.

[33] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," Proceedings of the 33rd International Conference on Software Engineering (ICSE), ACM, 2011.

[34] B. Kitchenham, D. Budgen, and O. P. Brereton, "Using mapping studies as the basis for further research," Information and Software Technology, Elsevier, vol. 53, no. 6, pp. 638–651, 2011.