UDC 004.4.05:004.75:004.8 **doi: 10.26906/SUNZ.2026.2.271**

Dmytro Diachenko, Vladyslav Diachenko

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

MODEL FOR ASSESSING THE RISK OF DEFECTS IN SOFTWARE COMPONENTS OF DISTRIBUTED COMPUTER SYSTEMS

Abstract. Relevance. The relevance of the study is determined by the need to improve the efficiency of automated testing of software in distributed computer systems, which are characterized by a complex structure, dependencies between components, and an increased risk of defects. Existing approaches to defect prediction mostly do not provide comprehensive consideration of the structural characteristics of components, the intensity of their changes, and the parameters of test execution, which complicates the justified planning of testing. Therefore, the development of a model for assessing the risk of defects in software components of distributed computer systems to support the process of prioritizing automated testing is relevant. This makes it possible to focus testing resources on the most critical software components and thereby increase the overall effectiveness of software quality control. **The object of research** is the process of detection and assessment of defects in software components of distributed computer systems in the context of automated testing. **Purpose of the article** is to develop a model for assessing the risk of defects in software components of distributed computer systems to improve the efficiency of automated testing. **Research results.** In this work, a model for assessing the risk of defects in software components of distributed computer systems has been developed, which makes it possible to form an integral defectiveness indicator based on the structural characteristics of the program code and the results of automated testing. Experimental studies using machine learning algorithms have shown that the best results are provided by the CatBoost model, which demonstrated the highest values of ROC–AUC and Precision–Recall characteristics compared to other investigated approaches. The results obtained confirm the possibility of effectively ranking software components by the level of defect risk and using this information for the prioritization of automated testing in distributed computer systems.

Keywords: distributed computer system, prediction, machine learning, automated testing, defect risk assessment, metrics, CatBoost, quality analysis, deep learning, multicomponent system.

Introduction

The modern development of computer systems is characterized by the active implementation of distributed architectures, within which application functionality is implemented by a set of interconnected software components, services, modules, and data exchange interfaces. Such systems are widely used in cloud platforms, corporate information environments, network services, financial technologies, telecommunications infrastructures, and other domains where increased requirements are imposed on the hardware and software complex regarding reliability, continuity of operation, and fault tolerance. The increasing complexity of the internal structure of distributed computer systems, the growth in the number of interconnections between components, and the high intensity of modification of software components lead to an increased probability of defects, which may result in partial or complete degradation of services, violation of data exchange integrity, and a decrease in the overall efficiency of system operation.

One of the key means of ensuring the quality and reliability of distributed computer systems is automated testing, which makes it possible to verify the correctness of component functionality, detect interaction errors between services, control the stability of software implementation after introducing changes, and maintain the required level of confidence in the results of system modification.

At the same time, with the growth of the scale of distributed systems, the number of test scenarios, and the frequency of software component updates, the problem of decreasing efficiency of the full cycle of automated testing arises. Running the entire set of tests for each new change requires significant computational, time, and infrastructure resources, while the use of fixed rules for test selection or prioritization often does not provide sufficient adaptation to the current state of the system.

Under these conditions, the preliminary assessment of the risk of defects in software components of a distributed computer system becomes particularly important, as it allows identifying the most vulnerable components even before the start of the full testing cycle and directing testing resources to those parts of the system where the probability of defects is the highest. This approach creates prerequisites for increasing the efficiency of automated testing by reducing the time to detect critical errors, ensuring justified prioritization of test scenarios, and decreasing the cost of performing regression testing.

A promising tool for solving this problem is the use of machine learning methods, which make it possible to identify hidden patterns in historical data on software code changes, results of previous test runs, characteristics of component structure, parameters of interactions between components, and operational performance indicators of the system. This is particularly important for distributed computer systems, in which defects often arise not only due to local implementation errors but also due to the complex dynamics of component interactions, configuration changes, uneven workloads, and the indirect influence of modifications in related subsystems.

Despite the significant number of studies in the field of software defect prediction, the issue of constructing a model for assessing defect risk specifically for software components of distributed computer systems remains relevant, with the aim of further improv-

ing the efficiency of automated testing. Existing approaches often focus either on general software code quality metrics or on local defect prediction without proper consideration of architectural dependencies, characteristics of the testing process, and the specific features of multicomponent system operation.

Therefore, the development of a model that combines structural, historical, testing, and operational characteristics of components and provides the formation of an integral defect risk assessment based on machine learning methods remains relevant.

The purpose of this work is to develop a model for assessing the risk of defects in software components of distributed computer systems to improve the efficiency of automated testing.

Main part

To solve the problem of improving the efficiency of automated testing of distributed computer systems, it is necessary to consider modern approaches to defect prediction, identify their advantages and limitations, and substantiate the possibility of constructing an original model for assessing the defect risk of software components.

In this regard, it is advisable to analyze modern studies devoted to the application of machine learning and deep learning methods in defect prediction tasks, which form the theoretical basis for the further development of the model, its description, and software implementation.

Paper [1] presents a systematic review of approaches to software defect prediction using artificial intelligence algorithms, which makes it possible to improve the efficiency of testing and software quality assurance processes. The results obtained can also be applied to the analysis of the reliability of software components in complex computer systems, in distributed architectures.

Considerable attention is also paid to the use of machine learning methods for defect prediction, considering their practical value for testing and system maintenance processes.

Paper [2] provides a systematic review of the application of machine learning in defect prediction tasks, focused not only on model accuracy but also on their suitability for use under real production conditions. The obtained results are important for the development of models for assessing the defect risk of components of distributed computer systems, since they confirm the expediency of combining predictive methods with the tasks of improving the efficiency of automated testing. It should be noted that this paper analyzes the most popular machine learning methods, among which decision trees, support vector machines, neural networks, random forests, and nearest neighbors occupy a leading position. At the same time, it is emphasized that the promise of a model is determined not only by classification accuracy but also by the possibility of its interpretation, reproducibility of results, consideration of the cost aspects of implementation, and suitability for application under specific organizational conditions.

Paper [3] presents a systematic review of the application of deep learning methods in defect prediction tasks, summarizing the algorithms used, datasets, approaches to source code representation, and methods for model evaluation. The authors concluded that a significant part of the research in the field of defect prediction is still based mainly on academic or open datasets, while the use of industrial data and the validation of models under real conditions remain limited. For the tasks of assessing the defect risk of components of distributed computer systems, this is especially important, since it confirms the need to develop models focused not only on high predictive accuracy but also on practical suitability for use in the conditions of automated testing of complex multicomponent architectures.

Paper [4] examines the current state of research devoted to software defect prediction using deep learning methods. The authors emphasize that, despite the growing interest in software defect detection, the use of deep learning in this field remains insufficiently systematized and requires the generalization of existing empirical results. The purpose of the study was to summarize scientific works on the application of deep learning to defect prediction from the standpoint of the metrics, models, techniques, datasets, and achieved results used, as well as to compare the effectiveness of deep learning models and classical machine learning methods.

An important area of research is defect prediction based on semantic features of source code. Paper [5] presents a systematic review of approaches to software defect detection based on the use of contextual information of source code, methods of its representation, and deep learning models. The obtained results are important for the development of models for assessing the defect risk of components of distributed computer systems, since they confirm the expediency of using semantic characteristics of software components to improve the efficiency of automated testing.

Paper [6] is devoted to the study of modern defect prediction models with an emphasis on the informativeness and explainability of the obtained results. It presents a systematic review of approaches to software defect prediction, within which the extent to which model results are understandable and suitable for practical use is analyzed. The conclusions obtained are important for the development of models for assessing the defect risk of components of distributed computer systems, since they confirm the expediency of moving from simple binary predictions to more informative models suitable for supporting automated testing.

Paper [7] is devoted to a comparative analysis of machine learning methods for test case prioritization under conditions of continuous testing constrained by strict time resources. The author investigates the influence of the volume of test execution history and the available time budget on the effectiveness of machine learning models in the early detection of regression defects, and also compares the approaches of SVM, ANN, GBDT, and LambdaRank using industrial datasets. The obtained results are important for the de-

velopment of models for assessing the defect risk of components of distributed computer systems, since they confirm the possibility of using historical testing data and machine learning methods to improve the efficiency of automated testing.

Paper [8] is devoted to the development of an approach to test prioritization based on learning-to-rank methods for early detection during regression testing. It proposes the variational approach APFD-Net, aimed at optimizing a differentiated objective function inspired by the APFD metric, which makes it possible to improve the consistency of priority space and enhance the generalization capability of the model. As a result, the authors showed that the proposed approach provides more effective early defect detection compared to traditional test prioritization methods.

Paper [9] is devoted to the development of an approach to test case prioritization in regression testing based on deep learning. It proposes the AnoLSTM-TCP method, which combines an LSTM model with anomaly features formed based on test execution history, test execution duration, and previous results, making it possible to detect temporal dependencies and rank test cases more effectively. As a result, the authors demonstrated that the proposed approach provides higher APFD values compared to well-known prioritization methods, including Random, ROCKET, RETECS, DeepGini, DeepOrder, LSTMTCP, and HyLSTMTCP, which indicates its better capability for early defect detection.

Paper [10] is devoted to a systematic analysis of the application of artificial intelligence methods for processing and analyzing logs in microservice architectures. It summarizes modern approaches to anomaly detection, root cause analysis, error prediction, and analysis of dependencies between services based on machine learning, deep learning, and hybrid learning methods. As a result, the authors concluded that, despite the significant potential of artificial intelligence approaches to improve the efficiency of log analysis in distributed systems, their practical implementation is constrained by problems of scalability, generalization of results, and limited availability of representative industrial datasets.

The conducted analysis of scientific publications indicates that modern studies confirm the prospects of applying machine learning and deep learning methods for defect prediction; however, most of them focus mainly on tasks of software module classification, test case prioritization, or analysis of individual types of data, in particular source code, testing results, or logs of microservice systems.

At the same time, for distributed computer systems the issue of constructing a generalized model that would combine the characteristics of software components, the results of automated testing, parameters of interaction between components, and operational indicators within a single defect risk assessment process remains insufficiently studied.

In addition, the literature analysis showed that a significant part of existing approaches is oriented either toward general open datasets or toward narrow prediction tasks without sufficient consideration of the specifics of multicomponent distributed architectures. This complicates the use of known solutions as a direct basis for improving the efficiency of automated testing in computer systems, where not only prediction accuracy is important but also the possibility of ranking components by risk, considering architectural dependencies, and supporting the subsequent prioritization of test scenarios.

In this regard, there is a need to develop a model for assessing the defect risk of software components of distributed computer systems that would consider the set of historical, structural, testing, and operational characteristics and ensure the formation of an integral risk assessment to support the automated testing process.

Therefore, the next stage of the research is the formulation of the problem and the substantiation of the structure of such a model.

Let a distributed computer system consist of a set of software components.

$$S = \{c_1, c_2, \ldots, c_n\}, \quad (1)$$

where each component c_i implements a separate functionality and interacts with other components through defined mechanisms of data exchange, service calls, or message transmission. For each component, a set of characteristics is formed.

$$X_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}, \quad (2)$$

where the features may belong to the following groups:

– characteristics of software code changes,

– indicators of previous testing results,

– structural parameters of the component,

– parameters of interaction with other components,

– test coverage indicators,

as well as operational characteristics.

It is necessary to construct such a mapping model

$$F : X_i \to R_i, \quad (3)$$

where R_i – component defect assessment c_i, presented in the form of the probability of defect occurrence, an integral risk index, or membership in a certain risk class. The task is to form, based on the available data, an informative and computationally feasible model for defect risk assessment that will allow:

- identifying components with the highest probability of defect occurrence in the current testing cycle.

- considering not only historical test results but also the influence of change intensity, component complexity, and its architectural dependencies.

- providing a basis for further prioritization and selection of test scenarios.

In distributed computer systems, the defectiveness of a software component cannot be considered only as a consequence of local implementation errors. Its level is the result of the influence of a set of factors, among which the intensity of code modifications, architectural complexity, the degree of interaction with other ser-

vices, the quality of existing test coverage, and the history of previous failures play an important role.

The first group of factors includes the characteristics of software code changes. Frequent commitments, a significant volume of modified lines, modification of critical modules, as well as changes in components with many dependencies usually increase the risk of introducing defects.

The second group is related to the results of previous automated tests. If a component or the test scenarios associated with it demonstrate instability, a high frequency of failures, or a history of detected defects, this may be an indicator of increased defectiveness.

The third group covers structural characteristics. These may include cyclomatic complexity, component size, the number of external calls, the number of API endpoints, the volume of business logic, the level of coupling, and the number of dependencies.

The fourth group includes testing characteristics, particularly the level of coverage, the average test execution time, the frequency of repeated runs, and the ratio of success to unsuccessful executions.

The fifth group of factors reflect the operational context:

– the number of incidents,

– the criticality of business functions,

– the frequency of component usage,

– its role in the overall service delivery architecture.

Taking these factors into account, it is advisable to form a comprehensive assessment that would not be reduced to a single indicator but would reflect the multidimensional nature of defectiveness in the components of a distributed system.

The proposed model assumes that the defectiveness of a software component is a latent characteristic that can be estimated based on a set of observable features.

For each component, a feature vector is formed.

$$X_i = \left(H_i, T_i, S_i, D_i, E_i\right), \qquad (4)$$

where H_i – historical characteristics of change, T_i – results of previous testing, S_i – structural characteristics of the component, D_i – parameters of dependencies between components, E_i – operational indicators.

Historical characteristics may include the number of changes over a defined period, the average volume of modifications, the number of authors of changes, and the frequency of updates.

The subset may include the frequency of test failures, the number of successful and unsuccessful runs, the number of flaky cases, the duration of test runs, and the test coverage of the component.

Structural characteristics T_i may include the frequency of test failures, the number of successful and unsuccessful runs, the number of flaky cases, the duration of test runs, and the test coverage of the component.

Structural characteristics S_i describe the complexity of the component, its size, the number of functional branches, and the density of internal dependencies.

Parameters D_i reflect the position of the component in the interaction graph of the system, for example the number of incoming and outgoing connections, centrality in the service graph, and the presence of critical dependencies.

Indicators E_i consider incidents in operation, the importance of the component for user scenarios, and the frequency of requests to the corresponding service. Based on the feature vector, an integral defect assessment is formed.

$$R_i = \alpha_1 H_i^* + \alpha_2 T_i^* + \alpha_3 S_i^* + \alpha_4 D_i^* + \alpha_5 E_i^*, \qquad (5)$$

where $H_i^*, T_i^*, S_i^*, D_i^*, E_i^*$ – normalized aggregated estimates for the corresponding groups of features, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ – weight coefficients that determine the degree of influence of each group of parameters on the final defect risk. In the general case, the values of the coefficients can be specified by experts or determined based on training a machine learning model. If a binary representation of the target variable is used, the model can produce an estimate in the form of the probability of defect occurrence in a component during the next testing cycle. If a multiclass scheme is chosen, the component is assigned to one of the risk classes, for example low, medium, or high. Unlike simplified approaches, in which prediction is based only on the history of previous failures, the proposed model considers both the internal characteristics of the component and its external context within the structure of the distributed system. This makes it possible to increase the informativeness of the assessment and create prerequisites for more justified planning of automated testing. For the practical use of the model, it is advisable to introduce an integral indicator of defect risk Q_i, which is defined as a normalized function of the final assessment R_i:

$$Q_i = \frac{R_i - R_{\min}}{R_{\max} - R_{\min}}. \qquad (6)$$

Then the value of $Q_i \in [0,1]$ can be interpreted as the risk level of the component. Based on threshold values, it is possible to form classes:

$$\mathrm{Risk}(Q_i) = \begin{cases} low, & Q_i < \theta_1, \\ middle, & \theta_1 \le Q_i < \theta_2, \\ high, & Q_i \ge \theta_2. \end{cases} \qquad (7)$$

Such an approach is convenient for further use in decision support systems during automated testing, since it makes it possible not only to rank components by the level of defectiveness but also to form sets of priority objects for in-depth test control. Under conditions of applying machine learning methods, the integral indicator can be formed based on the predicted probability of a component belonging to the class of defective ones. For this purpose, algorithms such as logistic regression, random forest, gradient boosting, XGBoost, or CatBoost can be used.

The advantage of this approach is the possibility of automatically determining the influence of individual features and evaluating their importance in forming the prediction.

The software implementation of the model is advisable to perform in Google Colab based on the open dataset Regenerated PROMISE and BPD Datasets [11], which contains component-oriented metrics and defect labels for several versions of open-source systems. Further research is planned to use the open SQuAD dataset [12].

The analysis of the file structure showed that it contains 36 data subsets formed for 13 open-source software systems, where each subset corresponds to a separate version of the system and represents software components at class level. Such a choice is justified because the dataset is representative for defect prediction tasks, has a unified feature structure across all subsets, and makes it possible to build a reproducible model for assessing the defect risk of components of distributed computer systems.

Unlike datasets focused on pull requests, commit-level analysis, or log analysis, this dataset is directly suitable for component-level modeling, which corresponds to the problem formulation presented in this work. Within the framework of experimental validation, it is advisable to combine all 36 subsets into a single array of observations.

In this case, the model will be trained on 12,690 software components for which six structural-metric characteristics are specified:

– CBO,
– DCC,
– ExportCoupling,
– ImportCoupling,
– NOM,
– WMC,

as well as the target feature Defect.

The meaning of these indicators makes it possible to interpret them as a set of characteristics of coupling, the intensity of interaction between components, structural complexity, and the functional content of a class. Such a structure is sufficient for constructing an initial model for defect risk assessment, since it allows considering not only the internal complexity of a software component but also its connections with other elements of the system.

At the same time, the target variable Defect should be interpreted as a binary indicator of the presence of a defect, while the output of the model should be interpreted as a probabilistic estimate of the defect risk for each component. For the software implementation, it is proposed to use a binary classification scheme with the subsequent interpretation of the predict_proba value as an integral risk index. At the experimental stage, it is advisable to build and compare several machine learning models, in particular logistic regression, support vector machines, random forest, and gradient boosting.

As the main model, it is advisable to choose Random Forest or XGBoost, since they work well with tabular metric data, are resistant to nonlinear dependencies between features, and allow the importance of individual component characteristics to be evaluated. The quality of the model should be evaluated using the metrics Accuracy, Precision, Recall, F1-score, and ROC-AUC, while practical usefulness should be assessed through ranking components according to the obtained risk index.

Thus, the software validation of the model is performed on an open multi-project dataset that contains structural characteristics of software components and labels of their defectiveness. This makes it possible to implement in Google Colab a defect risk assessment model as a tool for supporting automated testing: after training the model, components can be ordered according to the value of the risk index, which creates a basis for further prioritization of tests and concentration of testing resources on the riskiest elements of the system.

The results of the experimental study confirmed the effectiveness of the developed model. The analysis of feature importance showed that the greatest contribution to the formation of defectiveness assessment is made by combining characteristics of component complexity and coupling, metrics of the type ComplexityPlusCoupling, ComplexityTimesCoupling, MethodsTimesCoupling, as well as logarithmized structural indicators.

This indicates the expediency of considering not individual basic metrics but their integrated combinations when constructing the risk model.

The comparative analysis of models showed (Fig. 1) that the CatBoost model demonstrated the best results, for which the values ROC-AUC = 0.9156 and PR-AUC = 0.9116 were obtained. This confirms the high ability of the proposed approach to distinguish between defective and non-defective components and to effectively rank them according to the level of risk.

It is shown that the use of an integral defect risk index makes it possible to identify a group of components with the highest probability of defect occurrence, which creates a basis for prioritizing automated testing.

Fig. 2 presents a comparison of the Precision–Recall characteristics of the studied classification models.

The obtained results show that the proposed model based on CatBoost provides the highest overall prediction quality, which is confirmed by the largest value of the area under the Precision–Recall curve (AP = 0.9116).

The XGBoost model demonstrates a slightly lower result (AP = 0.8556), while Random Forest and Logistic Regression are characterized by even lower efficiency with AP values of 0.8391 and 0.7993, respectively.

The analysis of the curve shapes indicates that as recall increases, precision gradually decreases for all models, which is typical for defect prediction tasks. At the same time, the CatBoost model provides higher precision values over most of the recall range compared to other approaches, which confirms its ability to more accurately identify defective components of the software system.

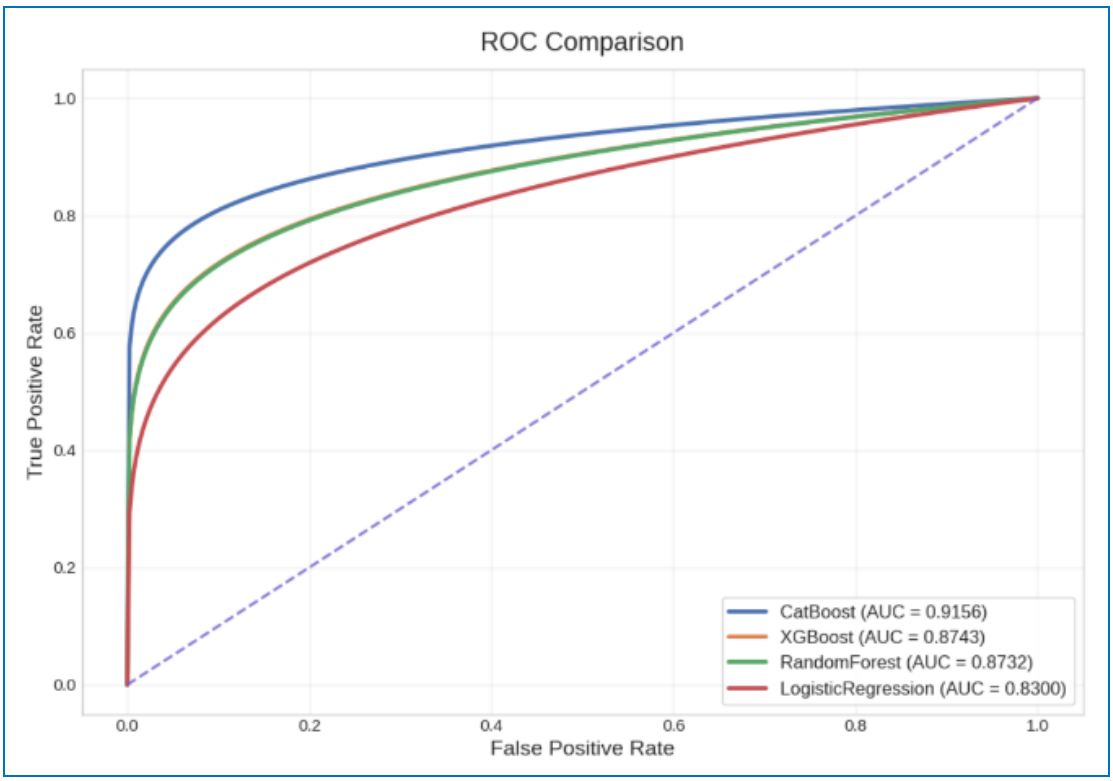

Fig. 1. ROC Comparison

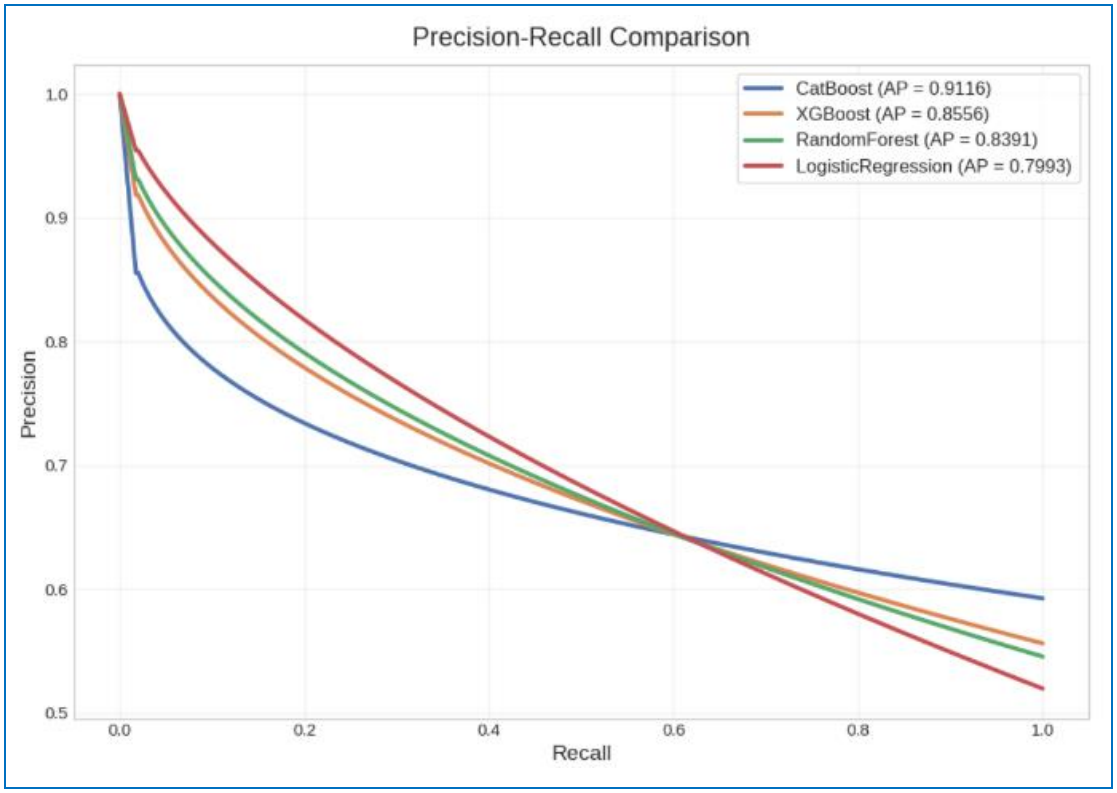

Fig. 2. Precision-Recall Comparison

Fig. 3 presents the distribution of the integral defect risk index of software components formed based on the results of the proposed model. The histogram demonstrates that most components are characterized by low and medium values of the risk index (approximately in the range of 0.2–0.4), while a relatively small proportion of components has increased values of the indicator exceeding 0.5. Such a distribution indicates the ability of the model to effectively differentiate system components by the level of potential defectiveness and to form a high-risk group that should be prioritized during test planning.

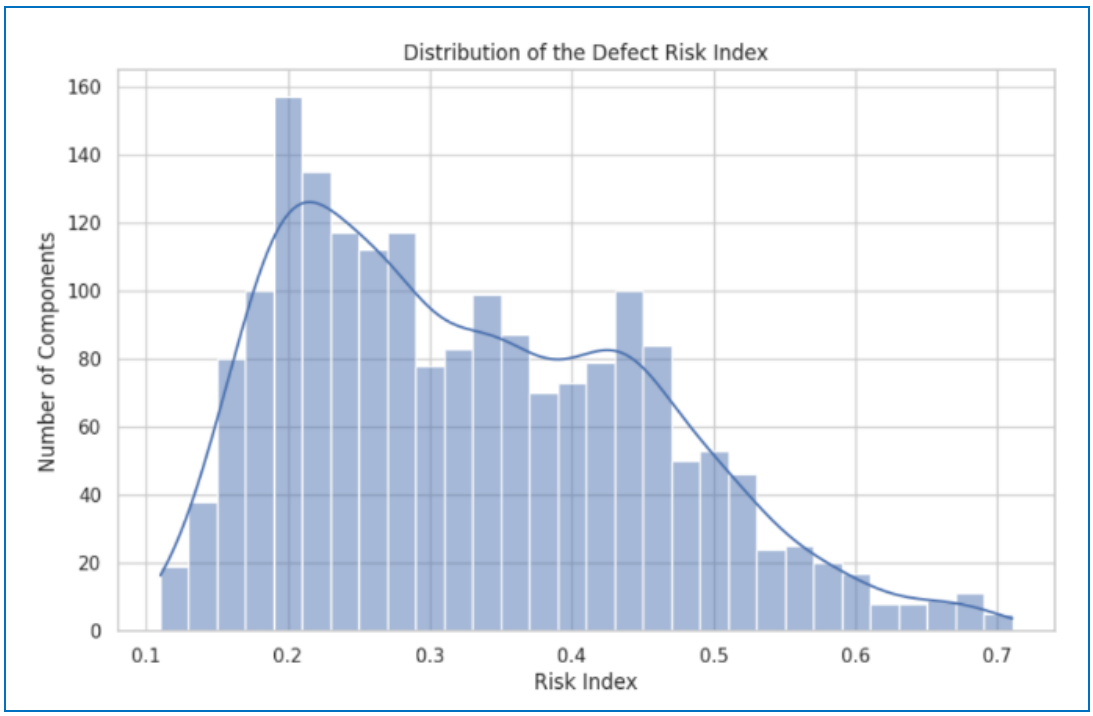

Fig. 3. Distribution of the Defect Risk Index

The obtained results confirm the high ability of the developed model to distinguish between defective and non-defective components.

The presence of a pronounced low-risk region and a smaller group of components with increased index values indicates the correctness of forming the integral risk indicator and its suitability for supporting the process of prioritizing the testing of software components of a distributed system.

Conclusions

During the study, the problem of improving the efficiency of automated testing of software for distributed computer systems was considered. The conducted analysis of modern approaches to defect prediction showed that most existing methods are focused mainly on the use of individual software code metrics or historical failure data, which limits their applicability for supporting the test planning process in complex distributed software systems.

In this work, a model for assessing the defect risk of software components of distributed computer systems was developed, which takes into account the structural characteristics of components, the intensity of changes, inter-component dependencies, and test execution parameters. Based on these data, an integral defect risk indicator is formed, which makes it possible to rank the components of a software system according to the level of potential defectiveness and to use the obtained estimates for the prioritization of automated testing.

The results of the experimental study confirmed the effectiveness of the proposed approach. A comparative analysis of machine learning algorithms showed that the CatBoost model demonstrates the best results, providing the highest values of ROC-AUC and Precision–Recall characteristics compared to the XGBoost, Random Forest, and Logistic Regression models. The analysis of the distribution of the defect risk index showed the possibility of effectively differentiating software components by the level of potential defectiveness and identifying a group of high-risk components.

The obtained results confirm that the use of the developed model makes it possible to improve the efficiency of planning automated testing of software for distributed computer systems by prioritizing the verification of components with the highest probability of defect occurrence.

References

1. Jalaj Pachouly, Swati Ahirrao, Ketan Kotecha, Ganeshsree Selvachandran, Ajith Abraham. A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools, Engineering Applications of Artificial Intelligence, Volume 111, 2022. P. 1–33. https://doi.org/10.1016/j.engappai.2022.104773.
2. Szymon Stradowski, Lech Madeyski. Machine learning in software defect prediction: A business-driven systematic mapping study. Information and Software Technology,Volume 155,2023. P. 1–17. https://doi.org/10.1016/j.infsof.2022.107128.
3. Görkem Giray, Kwabena Ebo Bennin, Ömer Köksal, Önder Babur, Bedir Tekinerdogan. On the use of deep learning in software defect prediction. Journal of Systems and Software, Volume 195, 2023. P. 1–26. https://doi.org/10.1016/j.jss.2022.111537.
4. Zuhaira Muhammad Zain, Sapiah Sakri, and Nurul Halimatul Asmak Ismail. Application of Deep Learning in Software Defect Prediction: Systematic Literature Review and Meta-analysis. Inf. Softw. Technol. Volume 158, 2023. https://doi.org/10.1016/j.infsof.2023.107175.

5. Abdu A, Zhai Z, Algabri R, Abdo HA, Hamad K, Al-antari MA. Deep Learning-Based Software Defect Prediction via Semantic Key Features of Source Code–Systematic Survey. Mathematics. 10(17):3120. 2022. P. 1–26. https://doi.org/10.3390/math10173120.
6. Natalie Grattan, Daniel Alencar da Costa, Nigel Stanger. The need for more informative defect prediction: A systematic literature review. Information and Software Technology. Volume 171, 2024. P. 1-24. https://doi.org/10.1016/j.infsof.2024.107456.
7. Marijan, D. Comparative study of machine learning test case prioritization for continuous integration testing. Software Qual J 31. 2023. P. 1415–1438. https://doi.org/10.1007/s11219-023-09646-0.
8. Peng Tang, Junfeng Wang, Mingxing Liu. Variational learning to rank for Test Case Prioritization via prioritizing metric inspired differentiable loss. Engineering Applications of Artificial Intelligence. Volume 141. 2025. https://doi.org/10.1016/j.engappai.2024.109776.
9. Tapas Kumar Choudhury, Mousumi Behera, Sanjit Kumar Dash, Subhendu Kumar Pani, Jibitesh Mishra, AnoLSTM-A Deep Learning Approach for Test Cases Prioritization, Procedia Computer Science, Volume 258.2025. Pages 1793-1803. https://doi.org/10.1016/j.procs.2025.04.431.
10. Md Arfan Uddin, Shakthi Weerasinghe, Darek Gajewski, Melika Akbarsharifi, Roxana Akbarsharifi, Christopher Stoner, Tomas Cerny, Sen He. Microservice logs analysis employing AI: A systematic literature review, Journal of Systems and Software. Volume 236.2026. P.1-93. https://doi.org/10.1016/j.jss.2026.112786.
11. S. Alhusain, "Predicting Relative Thresholds for Object Oriented Metrics," 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), 2021, pp. 55-63, doi: 10.1109/TechDebt52882.2021.00015.
12. Robredo, M., Esposito, M., Taibi, D., Peñaloza, R., & Lenarduzzi, V. SQuaD: The Software Quality Dataset - Dataset [Data set]. Zenodo. 2025. https://doi.org/10.5281/zenodo.17566691.

Received (Надійшла) 26.01.2026
Accepted for publication (Прийнята до друку) 22.04.2026
Publication date (Дата публікації) 22.05.2026

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Дяченко Дмитро Олександрович – аспірант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Dmytro Diachenko – PhD student, Department of Electronic Computers, Kharkiv National University of Radio Electronics Kharkiv, Ukraine;
e-mail: dmytro.diachenko2@nure.ua ; ORCID Author ID: http://orcid.org/0009-0006-5751-3511.

Дяченко Владислав Олександрович – PhD, комп'ютерна інженерія, старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Vladyslav Diachenko – PhD, Computer Engineering, Senior Lecturer Department of Electronic Computers, Kharkiv National University of Radio Electronics Kharkiv, Ukraine;
e-mail: vladyslav.diachenko@nure.ua; ORCID Author ID: http://orcid.org/ 0000-0003-2725-8784;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57207260441.

Модель оцінки ризику дефектів програмних компонентів розподілених комп'ютерних систем

Д. О. Дяченко, В. О. Дяченко

Анотація. Актуальність. Актуальність роботи зумовлена необхідністю підвищення ефективності автоматизованих випробувань програмного забезпечення розподілених комп'ютерних систем, для яких характерні складна структура, залежності між компонентами та підвищений ризик виникнення дефектів. Існуючі підходи до прогнозування дефектів здебільшого не забезпечують комплексного врахування структурних характеристик компонентів, інтенсивності їх змін і параметрів виконання тестів, що ускладнює обґрунтоване планування випробувань. Тому актуальною є розробка моделі оцінки ризику дефектів програмних компонентів розподілених комп'ютерних систем для підтримки процесу пріоритезації автоматизованих випробувань. Це дозволяє зосередити ресурси тестування на найбільш критичних програмних компонентах і тим самим підвищити загальну результативність контролю якості програмного забезпечення. **Мета статті**: розробка моделі оцінки ризику дефектів програм-них компонентів розподілених комп'ютерних систем для підвищення ефективності автоматизованих випробувань. **Об'єкт дослідження**: є процес виявлення та оцінювання дефектності програмних компонентів розподілених комп'ютерних систем у контексті автоматизованих випробувань. **Предмет дослідження:** моделі та методи оцінки ризику дефектів програмних компонентів розподілених комп'ютерних систем на основі аналізу їх структурних характеристик та результатів автоматизованих випробувань. **Результати дослідження**. У роботі розроблено модель оцінки ризику дефектів програмних компонентів розподілених комп'ютерних систем, що дозволяє формувати інтегральний показник дефектності на основі структурних характеристик програмного коду та результатів автоматизованих випробувань. Проведені експериментальні дослідження із застосуванням алгоритмів машинного навчання показали, що найкращі результати забезпечує модель CatBoost, яка продемонструвала найвищі значення ROC-AUC та Precision–Recall характеристик порівняно з іншими досліджуваними підходами. Отримані результати підтверджують можливість ефективного ранжування програмних компонентів за рівнем ризику дефектності та використання цієї інформації для пріоритизації автоматизованих випробувань у розподілених комп'ютерних системах.

Ключові слова: розподілена комп'ютерна система, прогнозування, машинне навчання, автоматизоване тестування, оцінка ризику дефектів, метрики, CatBoost, аналіз якості, глибоке навчання, багатокомпонентна система.