Evaluating Quality of Software Systems by the Confidence and Prediction Intervals of Regressions for RFC, CBO and WMC Metrics

SERGIY PRYKHODKO
Department of Software for Automated Systems,
Admiral Makarov National University of Shipbuilding,
Heroes of Ukraine Ave., 9, Mykolaiv, 54007,
UKRAINE

Abstract: - We have proposed to apply the confidence and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC at the app level to evaluate the quality of software systems from the point of view of their object-oriented design (OOD). A modified technique for evaluating the quality of software systems has been introduced. We have given the example of using the modified technique to detect the software quality of open-source Java systems.

Key-Words: - quality, software system, confidence interval, prediction interval, nonlinear regression, software metric, normalizing transformation.

Received: April 9, 2024. Revised: September 11, 2024. Accepted: October 13, 2024. Published: November 25, 2024.

1 Introduction

As we know, “software quality is given high priority”, [1]. However, despite the existing methods of software quality assessment, “there is still a lack of an effective estimation method for overall quality”, [2]. Also, the importance of the problem of evaluating the quality of software systems is evidenced by publications in recent years, [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20].

At the same time, “The backbone of any software system is its design” [21], including object-oriented design (OOD). To analyze the object-oriented system, special sets of metrics are used, for instance, CK [22] and MOOD, [23]. However, only the CK metrics are designed to measure the three non-implementation steps of OOD in Booch’s definition, [22].

Nowadays, software metrics, [5], [6], [8], [17], including RFC (response for a class) at the app level [5], [6], are used for detecting the quality of software systems. Also, we know that RFC depends on the metrics CBO (coupling between object classes) and WMC (weighted methods per class). Such dependency in the form of a linear regression was proposed, [24].

Although machine learning algorithms are becoming increasingly popular for software quality evaluation, [11], [12], [13], [14], [15], [16], methods of regression analysis have not yet reached their full potential, [1], [5], [6], [20]. In [1], the authors combined multiple linear regression and a fuzzy comprehensive evaluation method to build a quality evaluation algorithm. In [20], the authors used the linear regression algorithm for predicting the defect density in software apps and concluded existing approaches, including Case-Based Reasoning, are less precise than the Linear Regression methodology.

There is currently a known use of the technique [5] based on the confidence and prediction intervals of nonlinear regression for the RFC metric at the app level to evaluate the quality of open-source apps developed in Java [5] and Kotlin [6]. However, the metrics CBO and WMC, like RFC, also should be considered as the dependent variables that characterize the quality of software systems. That is why we proposed to modify the technique [5] to evaluate the quality of software systems from the point of view of their OOD. A modification is based on the confidence and prediction intervals of nonlinear regressions for RFC, CBO, and WMC at the app level. To build the nonlinear regression models, confidence, and prediction intervals of nonlinear regressions for RFC, CBO, and WMC, we apply the appropriate techniques based on multivariate normalizing transformations, [25].

2 Problem Formulation

Suppose given the original sample as the three-dimensional non-Gaussian data set: actual values of the RFC, CBO, and WMC metrics from N software systems. Suppose that there are two transformations: a bijective three-variate normalizing transformation of a non-Gaussian random vector $\mathbf{P}=\{RFC, CBO, WMC\}^T$ to a Gaussian random vector $\mathbf{T}=\{Z_{RFC}, Z_{CBO}, Z_{WMC}\}^T$ that is given by:

$$\mathbf{T}=\boldsymbol{\psi}(\mathbf{P}) \quad (1)$$

and the inverse transformation for (1):

$$\mathbf{P}=\boldsymbol{\psi}^{-1}(\mathbf{T}), \quad (2)$$

$\boldsymbol{\psi}$ is a vector of normalizing transformation (1), $\boldsymbol{\psi}=\{\psi_{RFC}, \psi_{CBO}, \psi_{WMC}\}^T$.

It is required to build three nonlinear regression models in the form $RFC=F_1(CBO, WMC, \varepsilon_1)$, $CBO=F_2(RFC, WMC, \varepsilon_2)$, and $WMC=F_3(RFC, CBO, \varepsilon_3)$, respectively, using transformations (1) and (2). Here ε_j is the error term that is the Gaussian random variable to describe residuals, $\varepsilon_j \sim N\left(0, \sigma^2_{\varepsilon_j}\right)$, σ_{ε_j} is the standard deviation, $j=1,2,3$.

Also, it is required to build the confidence and prediction intervals for the above three nonlinear regressions for the RFC, CBO, and WMC metrics to evaluate the quality of software systems from the point of view of their OOD.

3 Problem Solution

To evaluate the quality of software systems, we modify the technique for detecting software quality based on the confidence and prediction intervals of nonlinear regression for the RFC metric at the app level, [5]. The need for a modification is primarily due to that the other two metrics CBO and WMC, like RFC, should also be considered as dependent variables. Before using a modified technique, it is necessary to build nonlinear regression models, confidence, and prediction intervals. To construct them, you can use the appropriate techniques based on multivariate normalizing transformations, [25].

The modified technique follows six steps.

Step 1. Normalize the RFC, CBO, and WMC values (three-dimensional data point i) for the software system (system i) by the three-variate normalizing transformation, which has been used for finding the confidence and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC at the system level (app level).

Step 2. Calculate the squared Mahalanobis distance (SMD) for the three-dimensional normalized data point (point i).

Step 3. Check whether the SMD test statistic for the three-dimensional normalized data point (point i) is greater than a quantile of the corresponding distribution for this statistic. If yes then stop and go away (we cannot use the modified technique for point i) else go to step 4.

Step 4. Calculate borders of the confidence and prediction intervals of nonlinear regressions for the RFC, CBO, and WMC metrics at the system level for the three-dimensional data point (point i).

Step 5. Detect where the three-dimensional data point (point i) falls. If the data point (point i) for the software system is inside all confidence intervals of nonlinear regressions for the metrics RFC, CBO, and WMC, then stop (the software system has medium quality) else go to step 6.

Step 6. If the data point (point i) for the software system is between the upper borders of confidence intervals and the lower borders of prediction intervals for all three metrics, then the software system has high quality else the software system has low quality.

In step 1, we recommend using multivariate transformations, for instance, the Box-Cox [26] or Johnson, [27]. The choice of the multivariate transformation will depend on the data set for the metrics RFC, CBO, and WMC.

To calculate the SMD for a three-dimensional normalized data point (point i) in step 2, we apply the following formula:

$$d_i^2=\left(\mathbf{T}_i-\overline{\mathbf{T}}\right)^T \mathbf{S}_N^{-1}\left(\mathbf{T}_i-\overline{\mathbf{T}}\right), \quad (3)$$

where $\overline{\mathbf{T}}$ is the sample mean vector, $\overline{\mathbf{T}}=\{\overline{Z}_{RFC}, \overline{Z}_{CBO}, \overline{Z}_{WMC}\}^T$; $\mathbf{S}_N$ is the sample covariance matrix:

$$\mathbf{S}_N=\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{T}_i-\overline{\mathbf{T}}\right)\left(\mathbf{T}_i-\overline{\mathbf{T}}\right)^T . \quad (4)$$

In step 3, we apply a test statistic for value d_i^2 as follows, [28]:

$$N(N-3)d_i^2/3\left(N^2-1\right), \quad (5)$$

which has an approximate F distribution with a 3 and $N-3$ degrees of freedom and α significance level. According to [29], we take α as 0.005. We use the F distribution quantile $F_{3,N-3,0.005}$ with a 3 and $N-3$ degrees of freedom and 0.005 significance level to compare with (5).

We use the appropriate techniques based on multivariate normalizing transformations [25] to build models and intervals (confidence and prediction) of nonlinear regressions for the metrics RFC, CBO, and WMC. According to [25], we can build the confidence intervals of nonlinear regressions for the metrics RFC, CBO, and WMC as:

$$\psi_Y^{-1}\left(\hat{Z}_Y \pm t_{\alpha/2,\nu} S_{Z_Y} \left\{\frac{1}{N} + \left(\mathbf{z}_X^+\right)^T \mathbf{S}_Z^{-1} \left(\mathbf{z}_X^+\right)\right\}^{1/2}\right), \quad (6)$$

where ψ_Y is the normalizing transformation component for dependent variable Y; $\hat{Z}_Y$ is a prediction result by a linear regression equation $\hat{Z}_Y = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2$ dependent on predictors Z_1 and Z_2 for the normalized data, which are transformed by the three-variate normalizing transformation; $t_{\alpha/2,\nu}$ is a student's t-distribution quantile with a $\alpha/2$ significance level and ν degrees of freedom; $\nu = N-3$; $\mathbf{z}_X^+$ is a vector with components $Z_{1_i} - \bar{Z}_1$, $Z_{2_i} - \bar{Z}_2$ for i-row; $\bar{Z}_j = \frac{1}{N}\sum_{i=1}^{N} Z_{j_i}$, $j = 1,2$; $S_{Z_Y}^2 = \frac{1}{\nu}\sum_{i=1}^{N}\left(Z_{Y_i} - \hat{Z}_{Y_i}\right)^2$; $\mathbf{S}_Z$ is the 2×2 matrix:

$$\mathbf{S}_Z = \begin{pmatrix} S_{Z_1Z_1} & S_{Z_1Z_2} \\ S_{Z_1Z_2} & S_{Z_2Z_2} \end{pmatrix}, \quad (7)$$

where $S_{Z_qZ_r} = \sum_{i=1}^{N}\left[Z_{q_i} - \bar{Z}_q\right]\left[Z_{r_i} - \bar{Z}_r\right]$, $q, r = 1,2$.

To build the confidence interval of nonlinear regression for the metric RFC by (6), we need to substitute RFC, ψ_{RFC}, $\hat{Z}_{RFC}$, Z_{CBO}, Z_{WMC}, $\bar{Z}_{CBO}$, and $\bar{Z}_{WMC}$ instead of Y, ψ_Y, $\hat{Z}_Y$, Z_1, Z_2, $\bar{Z}_1$, and $\bar{Z}_2$, respectively. To construct the confidence interval of nonlinear regression for the metric CBO by (6), we need to substitute CBO, ψ_{CBO}, $\hat{Z}_{CBO}$, Z_{RFC}, Z_{WMC}, $\bar{Z}_{RFC}$, and $\bar{Z}_{WMC}$ instead of Y, ψ_Y, $\hat{Z}_Y$, Z_1, Z_2, $\bar{Z}_1$, and $\bar{Z}_2$, respectively. To build the confidence interval of nonlinear regression for the metric WMC by (6), we need to substitute WMC, ψ_{WMC}, $\hat{Z}_{WMC}$, Z_{RFC}, Z_{CBO}, $\bar{Z}_{RFC}$, and $\bar{Z}_{CBO}$ instead of Y, ψ_Y, $\hat{Z}_Y$, Z_1, Z_2, $\bar{Z}_1$, and $\bar{Z}_2$, respectively.

The prediction interval of the nonlinear regression is constructed analogously (6) with the only difference that 1 more must be added to the sum in curly brackets (6).

4 An Example of Problem Solution

We give an example of using the modified technique to detect the software quality of open-source Java systems. To build models, the confidence and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC at the system level by (6) for our example, we use the data of RFC, CBO, and WMC of 46 open-source Java-systems hosted on GitHub from [5]. In [5], the data was obtained using the CK tool and cleaned from the three-variate outliers. The data of RFC, CBO, and WMC metrics of 46 open-source Java systems was supplemented by others of the same metrics from [30] for three popular open-source Java systems of some versions: FreeMind 0.9.0Beta17, jEdit (2.6final and 3.0final), and TuxGuitar 1.3.0. Also, we added the data other three versions of the above systems hosted on GitHub: FreeMind 1.1.0Beta2, jEdit 5.5.0, and TuxGuitar 1.5.2src. Thus, we had the data of the metrics RFC, CBO, and WMC from 53 software systems. Like [5], the data was cleaned from two three-variate outliers (FreeMind 1.1.0Beta2 and TuxGuitar 1.3.0). In the following, we used 51 data points.

As in [5], to normalize the data, we applied the three-variate Box-Cox transformation (BCT) with components:

$$Z_j = \begin{cases} \left(X_j^{\lambda_j} - 1\right)/\lambda_j, & \text{if} \quad \lambda_j \neq 0; \\ \ln\left(X_j\right), & \text{if} \quad \lambda_j = 0. \end{cases} \quad (8)$$

Here Z_j is the Gaussian variable and λ_j is a parameter of BCT, $j = 1,2,3$. The variable Z_{RFC} is defined analogously (8) with the only difference that instead of Z_j, X_j, and λ_j should be put Z_{RFC}, RFC, and λ_{RFC}, respectively. The variables Z_{CBO} and Z_{WMC} are defined similarly. The parameter estimates of the three-variate BCT for the data are calculated by the maximum likelihood method according to [29] and are $\hat{\lambda}_{RFC} = 0.194965$, $\hat{\lambda}_{CBO} = 0.851253$, $\hat{\lambda}_{WMC} = -0.567096$.

We built the nonlinear regression models for the metrics RFC, CBO, and WMC based on the three-variate BCT in the form [5]:

$$Y = \left[\hat{\lambda}_Y\left(\hat{Z}_Y + \varepsilon\right) + 1\right]^{1/\hat{\lambda}_Y}, \quad (9)$$

where $\hat{Z}_Y$ is a prediction result by the linear regression equation $\hat{Z}_Y = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2$ dependent on predictors Z_1 and Z_2 for the normalized data, which are transformed by the three-variate normalizing transformation; ε is a Gaussian random variable, $\varepsilon \sim N(0, \sigma_\varepsilon^2)$.

To build the nonlinear regression model for the metric RFC by (9), we need to substitute *RFC*, $\hat{\lambda}_{RFC}$, $\hat{Z}_{RFC}$, Z_{CBO}, Z_{WMC}, ε_1, and σ_{ε_1} instead of *Y*, $\hat{\lambda}_Y$ $\hat{Z}_Y$, Z_1, Z_2, ε, and σ_ε, respectively. To build the nonlinear regression model for the metric CBO by (9), we need to substitute *CBO*, $\hat{\lambda}_{CBO}$, $\hat{Z}_{CBO}$, Z_{RFC}, Z_{WMC}, ε_2, and σ_{ε_2} instead of *Y*, $\hat{\lambda}_Y$, $\hat{Z}_Y$, Z_1, Z_2, ε, and σ_ε, respectively. To build the nonlinear regression model for the metric WMC by (9), we need to substitute *WMC*, $\hat{\lambda}_{WMC}$, $\hat{Z}_{WMC}$, Z_{RFC}, Z_{CBO}, ε_3, and σ_{ε_3} instead of *Y*, $\hat{\lambda}_Y$, $\hat{Z}_Y$, Z_1, Z_2, ε, and σ_ε, respectively. The parameter estimates of the nonlinear regression models for the metrics RFC, CBO, and WMC are shown in Table 1.

Table 1. The parameter estimates of the nonlinear regression models

No	*Y*	b_0	b_1	b_2	σ_ε	*MMRE*	*PRED*
1	*RFC*	-3.69701	0.11637	4.59287	0.2743	0.1346	0.8627
2	*CBO*	8.09952	4.37867	-11.4975	1.6823	0.1974	0.7451
3	*WMC*	0.99535	0.12980	-0.00864	0.0461	0.1949	0.7059

To assess the predictive accuracy of nonlinear regression models for RFC, CBO, and WMC in the form (9), we utilized standard metrics, namely MMRE and PRED(0.25). The acceptable values of MMRE and PRED(0.25) are not more than 0.25 and not less than 0.75, respectively. Table 1 contains the MMRE and PRED(0.25) values for the above models. These values indicate the satisfactory quality of the models.

To calculate SMD for the three-dimensional normalized data point (point *i*) in step 2 of the considered example of the modified technique, we need to use the following values in (3): $\bar{Z}_{RFC} = 3.731$ $\bar{Z}_{CBO} = 8.244$, $\bar{Z}_{WMC} = 1.409$, and the matrix inverse (4)

$$\mathbf{S}_N^{-1} = \begin{pmatrix} 13.561 & -1.578 & -62.283 \\ -1.578 & 0.3604 & 4.144 \\ -62.283 & 4.144 & 479.825 \end{pmatrix}.$$

In step 3, we use the *F* distribution quantile with 3 and 48 degrees of freedom and 0.005 significance level $F_{3,48,0.005} = 4.85$.

To calculate the borders of the confidence interval of nonlinear regression for the RFC metric, we need to use the following values in (6) and (7): $S_{Z_Y} = 0.2799$, $\bar{Z}_1 = 8.244$, $\bar{Z}_2 = 1.409$, and

$$\mathbf{S}_Z^{-1} = \begin{pmatrix} 0.003466 & -0.06087 \\ -0.06087 & 3.79940 \end{pmatrix}.$$

To calculate the borders of the confidence interval of nonlinear regression for the metric CBO, we need to use the following values in (6) and (7): $S_{Z_Y} = 1.7170$, $\bar{Z}_1 = 3.731$, $\bar{Z}_2 = 1.409$, and

$$\mathbf{S}_Z^{-1} = \begin{pmatrix} 0.13041 & -0.86547 \\ -0.86547 & 8.47418 \end{pmatrix}.$$

To calculate the borders of the confidence interval of nonlinear regression for the metric WMC, we need to use the following values in (6) and (7): $S_{Z_Y} = 0.0471$, $\bar{Z}_1 = 3.731$, $\bar{Z}_2 = 8.244$, and

$$\mathbf{S}_Z^{-1} = \begin{pmatrix} 0.10738 & -0.02040 \\ -0.02040 & 0.006365 \end{pmatrix}.$$

In all cases of the considered example for calculating borders of the confidence intervals we need to use the following values in (6): $t_{0.05/2,48} = 2.011$, *N*=51, and $\nu = 48$.

We consider the examples of evaluating the software quality of open-source Java systems by the proposed technique. We took the values of the RFC, CBO, and WMC metrics at the app level from three popular open-source Java systems [30]: FreeMind, jEdit, and TuxGuitar. Also, we obtained the values of these metrics using the CK tool for the above software systems. In addition, we took the values of the RFC, CBO, and WMC metrics at the app level from four other open-source Java systems hosted on GitHub: Apache Commons Lang, Hosebird Client, gwt-bootstrap, and itcoinj. Apache Commons Lang (commons-lang) is a package of Java utility classes for the classes that are in Java.lang's hierarchy. Hosebird Client (HBC) is a Java HTTP client for consuming Twitter's real-time Streaming API. Gwt-bootstrap is a GWT Library that provides the widgets of Bootstrap, from Twitter. The bitcoinj library is a Java implementation of the Bitcoin protocol, which allows it to maintain a wallet and send/receive transactions without needing a local copy of Bitcoin Core. Table 2 shows the metrics RFC, CBO, and WMC from the above apps, and test statistic (TS) (5).

The quality evaluation results from Table 2 are slightly different from the results from [5]. This can be explained primarily by the modified technique (unlike the technique from [5]) considers the other two metrics CBO and WMC, like RFC, as the dependent variables.

Table 2. The quality evaluation results

i	App name	*RFC*	*CBO*	*WMC*	TS (5)	quality
1	TuxGuitar 1.5.2-src	15,45	8,54	14,52	0.52	low
2	jEdit 5.5.0	26,49	7,91	39,03	3.38	low
3	jEdit 3.0final	10.38	4.29	14.00	1.28	high
4	jEdit 2.6final	8.84	4.24	9.27	1.68	low
5	FreeMind 0.9.0Beta17	13.29	5.31	12.16	1.88	low
6	commons-lang 4x	25.39	13.80	42.11	0.97	low
7	bitcoinj	37.16	17.81	65.84	1.43	medium
8	gwt-bootstrap	10.62	7.52	12.95	0.55	high
9	HBC	13.10	10.53	13.74	0.13	high

Also, we tried to use the modified technique example to evaluate the quality of three software systems (A, B, and C), for which the quality is classified in NASA's research as low, high, and medium, respectively [24]. We could not evaluate the quality of these systems by the modified technique example since their relevant values of the test statistic (5) for the normalized metrics RFC, CBO, and WMC are greater than 4.85. These results may be explained by the system A is commercial software, system B is NASA software, and system C is developed in C++.

5 Discussion

To evaluate the quality of software systems, we propose the modified technique based on the confidence and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC. This choice is due to the following. Firstly, according to [22], the CK metrics are designed to measure the three non-implementation steps in Booch's definition of OOD. These are the metrics WMC, DIT, NOC, RFC, CBO, and LCOM, which define the OOD complexity in the above steps. In particular, the metrics RFC and CBO define the OOD complexity due to the relationships between classes, [31]. And, as we know, the OOD complexity affects the quality of software systems.

Finally, the above metrics together characterize the OOD complexity and quality of software systems that require the use of multivariate analysis methods, such as multivariate statistical analysis. One of them is regression analysis. In this case, as a rule, nonlinear regression analysis should be used since only in special cases can the use of a linear regression model be theoretically justified for estimating software metrics.

We apply the three-variate Box-Cox normalizing transformation to build the nonlinear regression models, the confidence and prediction intervals for the nonlinear regressions for the metrics RFC, CBO, and WMC by [25] since, firstly, according to the Mardia test [32], the distribution of the three-dimensional normalized data is Gaussian and, secondly, the residuals distribution of corresponding linear regression models for normalized data is Gaussian.

To build the confidence and prediction intervals for the nonlinear regressions for the metrics RFC, CBO, and WMC for evaluating the quality of software systems, we used a 0.05 significance level, as the appointed one usually, although this value may be discussed.

Preliminary, we have studied the stability of the quality evaluation results dependent on a significance level value. We evaluated the quality of software systems from Table 2 for two values of significance level: 0.04 and 0.06. The results are the same as for a 0.05 significance level. That indicates the stability of the quality evaluation results at least within a 20 percent change in a significance level.

Concerning the example of using the modified technique to detect the software quality of open-source Java systems two limitations should be acknowledged and addressed concerning the data sample from 51 open-source apps in Java. The first limitation concerns the estimation of the data sample for open-source apps developed in Java only. The evaluation of other data samples, for instance, the industrial systems in Java, may affect the bounds of the confidence and prediction intervals of the nonlinear regressions for the metrics RFC, CBO, and WMC. In such cases, the proposed bounds of the confidence and prediction intervals of the nonlinear regressions for the metrics RFC, CBO, and WMC remain to be confirmed or changed.

The second limitation concerns the sample size, which equals 51. This value cannot be unambiguously considered as the lower size limit of the large sample. Larger sample sizes may lead to a reduction of the widths of the confidence and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC.

The quality of some Java systems from Table 2 is rated as low because the upper bound of the prediction interval is exceeded for only one metric. For instance, TuxGuitar 1.5.2-src, jEdit 5.5.0, and jEdit 2.6final have low quality since the RFC values of these systems exceed the upper bounds of the prediction intervals on 11, 18, and 6 percent, respectively. These results can be explained by the above systems having such classes for which the RFC values are greater than 100 (an acceptable maximum limit [33]). For instance, there are two such classes in TuxGuitar 1.5.2-src: org.herac.tuxguitar.app.action.installer.TGActionInstaller and org.herac.tuxguitar.android.action.installer.TGActionInstaller for which the RFC values equal 252 and 178, respectively. In this case, it is necessary to decide whether these classes are difficult to understand due to the large number of methods in every class's response set, and if necessary, reduce their number.

Also, the quality of the above systems can be improved by improving the relationships between classes. In our opinion, the modified technique allows us to assess how balanced OOD of a software system using the metrics RFC, CBO, and WMC. And, as we know [34], “Systems engineering seeks a safe and balanced design in the face of opposing interests and multiple, sometimes conflicting constraints.”

The given example of the modified technique use is illustrative and demonstrates its capabilities. In the future, it is necessary to build corresponding models, the confidence, and prediction intervals of nonlinear regressions for the metrics RFC, CBO, and WMC based on various data sets.

6 Conclusion

We have proposed to apply the confidence and prediction intervals of nonlinear regressions for the RFC, CBO, and WMC metrics for evaluating the quality of software systems from the point of view of their OOD. To estimate the confidence and prediction intervals of nonlinear regressions for the RFC, CBO, and WMC metrics need to use multivariate normalizing transformations. In this case, we have used the three-variate Box-Cox transformations.

We have introduced the modified technique for evaluating the quality of software systems. We have given the example of using the modified technique to detect the software quality of open-source Java systems.

Moving forward, we plan to develop examples of applying the modified technique that does not have the above limitations due to the programming language and the sample size.

References:

[1] C. Gao, W. Luo, J. Wang, and Y. Zhang, Software quality evaluation model based on multiple linear regression and fuzzy comprehensive evaluation method, *Proceedings of the 2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, Wulumuqi, China, 2022, pp. 383-389. DOI: https://doi.org/10.1109/DSA56465.2022.00058.

[2] C. Chen, M. Shoga, and B. Boehm, Exploring the dependency relationships between software qualities, *Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Sofia, Bulgaria, 2019, pp. 105-108. DOI: https://doi.org/10.1109/QRS-C.2019.00032.

[3] A. Madaehoh and T. Senivongse, OSS-AQM: An open-source software quality model for automated quality measurement, *Proceedings of the 2022 International Conference on Data and Software Engineering (ICoDSE)*, Denpasar, Indonesia, 2022, pp. 126-131. DOI: https://doi.org/10.1109/ICoDSE56892.2022.9972135.

[4] T. B. Alakus, R. Das, and I. Turkoglu, An overview of quality metrics used in estimating software faults, *Proceedings of the 2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Turkey, 2019, pp. 1-6. DOI: https://doi.org/10.1109/IDAP.2019.8875925.

[5] S. Prykhodko and N. Prykhodko, A technique for detecting software quality based on the confidence and prediction intervals of nonlinear regression for RFC metric, *Proceedings of the 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2022, pp. 499-502. DOI: https://doi.org/10.1109/CSIT56902.2022.10000532.

[6] S. Prykhodko and N. Prykhodko, Estimating quality of open-source Kotlin-based apps by the confidence and prediction intervals of nonlinear regression for RFC metric, *Proceedings of the 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*, Lviv, Ukraine, 2023, pp. 1-4. DOI: https://doi.org/10.1109/CSIT61576.2023.10324187.

[7] A. Jabborov, A. Kharlamova, Z. Kholmatova, A. Kruglov, V. Kruglov, and G. Succi, Taxonomy of quality assessment for intelligent software systems: A systematic literature review, *IEEE Access*, vol. 11, 2023, pp. 130491-130507. DOI: https://doi.org/10.1109/ACCESS.2023.3333920.

[8] I. M. A. Wikantyasa, A. P. Kurniawan, and S. Rochimah, CK metric and architecture smells relations: towards software quality assurance, *Proceedings of the 2023 14th International Conference on Information & Communication Technology and System (ICTS)*, Surabaya, Indonesia, 2023, pp. 13-17. DOI: https://doi.org/10.1109/ICTS58770.2023.10330874.

[9] A. Almogahed, M. Omar, N. H. Zakaria, G. Muhammad, and S. A. AlQahtani, Revisiting scenarios of using refactoring techniques to improve software systems quality, *IEEE Access*, vol. 11, 2023, pp. 28800-28819. DOI: https://doi.org/10.1109/ACCESS.2022.3218007.

[10] D. K. K. Shyamal, P. P. G. D. Asanka, and D. Wickramaarachchi, A comprehensive approach to evaluating software code quality through a flexible quality model, *Proceedings of the 2023 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, Kelaniya, Sri Lanka, 2023, pp. 1-8. DOI: https://doi.org/10.1109/SCSE59836.2023.10215004.

[11] O. Bombiri, P. Poda, and T. F. Ouedraogo, Application of machine learning in software quality: a mini-review, *Proceedings of the 2023 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD)*, Bobo-Dioulasso, Burkina Faso, 2023, pp. 1-7. DOI: https://doi.org/10.1109/MNE3SD57078.2023.10079800.

[12] M. Jukaria, O. H. Saif, M. Enas, H. M. J. Abdul, H. Laith, A. Sajjad, and Ali SaadAlwan, Exploring perspectives, issues, and practices in the testing and quality validation of AI software, *Proceedings of the 2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India, 2024, pp. 768-772. DOI: https://doi.org/10.1109/ICACITE60783.2024.10617318.

[13] R. Li, Software Quality Testing Framework based on Machine Learning Analysis, *Proceedings of the 2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, Lalitpur, Nepal, 2024, pp. 396-401. DOI: https://doi.org/10.1109/ICMCSI61536.2024.00063.

[14] Y. Qi, Research on quality prediction of airborne software based on neural network optimized by ant colony algorithm, *Proceedings of the 2024 IEEE 4th International Conference on Power, Electronics and Computer Applications (ICPECA)*, Shenyang, China, 2024, pp. 1338-1343. DOI: https://doi.org/10.1109/ICPECA60615.2024.10471180.

[15] D. Choudhury and D. Gupta, Investigation on integration of machine learning techniques into LC/NC platforms for code review, quality assessment, and error detection automation, *Proceedings of the 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2024, pp. 1-7. DOI: https://doi.org/10.1109/ICRITO61523.2024.10522267.

[16] V. K. Swain, R. Panigrahi, K. K. Sahu, and N. Padhy, Revealing MOVE METHOD: Machine learning-driven real-time refactoring identification for enhanced software quality, *Proceedings of the 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, Bhubaneswar, India, 2024, pp. 572-578. DOI: https://doi.org/10.1109/ESIC60604.2024.10481609.

[17] S. Jin, Z. Li, B. Chen, B. Zhu, and Y. Xia, Software code quality measurement: implications from metric distributions, *Proceedings of the 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*, Chiang Mai, Thailand, 2023, pp. 488-496. DOI: https://doi.org/10.1109/QRS60937.2023.00054.

[18] F. Basciani, D. Di Pompeo, J. Di Rocco, and A. Pierantonio, A customizable approach to assess software quality through multi-criteria decision making, *Proceedings of the 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*,

L'Aquila, Italy, 2023, pp. 264-271. DOI: https://doi.org/10.1109/ICSA-C57050.2023.00063.

[19] F. Témolé and D. Atanasova, Role, importance and significance of software quality, *Proceedings of the 2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, Opatija, Croatia, 2023, pp. 1658-1663. DOI: https://doi.org/10.23919/MIPRO57284.2023.10159733.

[20] N. Saravanan, C. Dharanya, M. Dhina, E. Ranjth Kumar, and L. Shanmathi Devi, A novel approach to predict the defect density in software application using linear regression algorithm, *Proceedings of the 2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, Coimbatore, India, 2024, pp. 1-5. DOI: https://doi.org/10.1109/ICSTEM61137.2024.10560850.

[21] F. B. Abreu, M. Goulilo, and R. Esteves, Toward the design quality evaluation of object-oriented software systems, *Proceedings of the 5th International Conference on Software Quality*, Austin, Texas, USA, October 1995, pp. 1-4, [Online]. https://www.academia.edu/3853615/Toward_the_Design_Quality_Evaluation_of_Object_Oriented_Software (Accessed Date: June 21, 2024).

[22] S. R. Chidamber and C. F. Kemerer, Towards a metrics suite for object oriented design, *ACM SIGPLAN Notices*, Vol. 26, Issue 11, 1991, pp. 197–211. DOI: https://doi.org/10.1145/118014.117970.

[23] F. Brito e Abreu and W. Melo, Evaluating the impact of object-oriented design on software quality, *Proceedings of the 3rd International Software Metrics Symposium*, Berlin, Germany, 1996, pp. 90-99, DOI: https://doi.org/10.1109/METRIC.1996.492446.

[24] V. Laing and C. Coleman, *Principal components of orthogonal object-oriented metrics*, White Paper Analyzing Results of NASA Object-Oriented Data. SATC, NASA, 2001, [Online]. https://www.semanticscholar.org/paper/Principal-Components-of-Orthogonal-Object-Oriented-Laing-Coleman/10260d67fa5ed4ecea9dd4398c3f236c52cabf6b (Accessed Date: June 21, 2024).

[25] S. Prykhodko and N. Prykhodko, Mathematical modeling of non-Gaussian dependent random variables by nonlinear regression models based on the multivariate normalizing transformations, *Proceedings from MODS'2020: Mathematical Modeling and Simulation of Systems. Advances in Intelligent Systems and Computing*, Vol. 1265, Springer, Cham, 2021, pp. 166-174. DOI: https://doi.org/10.1007/978-3-030-58124-4_16.

[26] S. Velilla, A note on the multivariate Box-Cox transformation to normality. *Statistics and Probability Letters*, 17, 1993, pp. 259–263. DOI: https://doi.org/10.1016/0167-7152(93)90200-3.

[27] P. M. Stanfield, J. R. Wilson, G. A. Mirka, N. F. Glasscock, J. P. Psihogios, and J. R. Davis, Multivariate input modeling with Johnson distributions, *Proceedings of the 28th Winter simulation conference WSC'96*, 8-11 December 1996, Coronado, CA, USA, ed. S.Andradóttir, K.J.Healy, D.H.Withers, and B.L.Nelson. IEEE Computer Society Washington, DC, USA, 1996, pp.1457-1464, [Online]. https://www.informs-sim.org/wsc96papers/209.pdf (Accessed Date: June 21, 2024).

[28] A. A. Afifi and S. P. Azen, *Statistical analysis: a computer-oriented approach*, New York, London, Academic Press, 1972. DOI: https://doi.org/10.1016/C2013-0-10273-5.

[29] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Pearson Prentice Hall, 2007, [Online]. https://books.google.com.ua/books/about/Applied_Multivariate_Statistical_Analysi.html?id=gFWcQgAACAAJ&redir_esc=y (Accessed Date: June 21, 2024).

[30] AJ. Molnar, A. Neamţu, and S. Motogna, Evaluation of software product quality metrics, In E. Damiani, G. Spanoudakis, L. Maciaszek. Eds. *Evaluation of Novel Approaches to Software Engineering. ENASE 2019. Communications in Computer and Information Science*, Vol. 1172, Springer, Cham, 2020, pp. 163-187. DOI: https://doi.org/10.1007/978-3-030-40223-5_8.

[31] S. R. Chidamber and C. F. Kemerer, A metrics suite for object-oriented design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, pp. 476-493. DOI: https://doi.org/10.1109/32.295895.

[32] K. V. Mardia, Measures of multivariate skewness and kurtosis with applications, *Biometrika*, 57, 1970, pp. 519–530. DOI: https://doi.org/10.1093/biomet/57.3.519.

[33] Response for Class, [Online]. https://objectscriptquality.com/docs/metrics/response-for-class (Accessed Date: June 23, 2024).

[34] SEH 2.0. Fundamentals of Systems Engineering, [Online]. https://www.nasa.gov/reference/2-0-fundamentals-of-systems-engineering/ (Accessed Date: June 23, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Prof. Sergiy Prykhodko is the only author of this article. He made an alone contribution to this research at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflicts of interest to declare that are relevant to the content of this article.