A Five-Factor Nonlinear Regression Model for JAVA Applications Size Estimation

Oleksandr Oriekhov[1,*,†], Tetyana Farionova[1,*,†], Liubava Chernova[1] and Mykhailo Vorona[1]

[1] *Admiral Makarov National University of Shipbuilding, Ukraine, Heroes avenue, 9, Mykolaiv, 54007, Ukraine.*

Abstract

The research proposes a five-factor nonlinear regression model for JAVA applications size estimation at the early stages of project planning for further usage in parametric models for effort estimation of software development. Accurate software development effort estimation is necessary for project planning to manage risk assessment, identify potential planning gaps, enhance the efficiency of the software development process, resource allocation, and costs. JAVA is one of the widely used programming languages in the world and is actively used in the development of various software projects. The aim of the research is to improve the accuracy and reliability of JAVA-applications size estimation at the early stages of software project planning. To achieve this goal, existing equations and models for JAVA-application KLOC estimations were reviewed and compared. A dataset of 571 open-source JAVA applications code metrics was collected using the CK static code analysis tool, and it was split up into learning and validation samples for model construction and validation. Firstly, the five-factor regression model is constructed using the total number of actual classes and interfaces metrics and averages of VMQ, TFQ, and CBO per class on the basis of a multivariate Box-Cox normalizing function. The model is constructed through an iterative process by detecting and removing anomalies from the sample. The constructed model is compared to the existing models by the standard regression model quality criteria such as the coefficient of determination, $MMRE$, and the $PRED(0.25)$. The estimates of criteria R^2, $MMRE$, and $PRED(0.25)$ for the latest iteration on the learning sample are 0.9759, 0.1276, and 0.9008 respectively, and the estimates on the basis of initial learning and validation samples exceeded the thresholds, which indicates good accuracy and reliability of the constructed regression model. The forecast interval was constructed for the regression and compared with four-factor nonlinear regression. The study confirms that the accuracy and reliability of KLOC estimation for JAVA applications have been successfully improved.

Keywords

Software project management, application size estimation, nonlinear regression model, JAVA application, normalizing function, non-Gaussian data, multivariate Box-Cox normalizing function, multicollinearity, code metrics.

1. Introduction

The effort estimation of software development (SDEE) is one of critical factors in effective project management. Accurate and reliable SDEE is necessary for project planning to manage risk assessment, identify potential planning gaps, enhance the efficiency of the software development process, resource allocation, and costs. Valid estimates allow reducing uncertainty, enabling project managers to optimal resource allocation and reduce unforeseen challenges. When estimations are close to actual software development efforts, the risk is reduced for the objective project goals, as a result, it leads to more predictable and controlled software development lifecycles. The application size can be represented as functional points (FP) or number of code lines (KLOC - kilo lines of code). Both variants have their own pros and cons. Compared to functional points, usage of KLOC application size considers important parameters such as environmental factors, including

ITPM-2025: VI International Workshop "IT Project Management", May 22, 2025, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

oleksandr.oriekhov@nuos.edu.ua (O. Oriekhov); tetyana.farionova@nuos.edu.ua (T. Farionova); liubava.chernova@nuos.edu.ua (L. Chernova); mykhailo.vorona@nuos.edu.ua (M. Vorona)

0000-0002-0001-0140 (O. Oriekhov); 0000-0003-3384-4712 (T. Farionova); 0000-0001-7846-9034 (L. Chernova); 0000-0003-4288-0096 (M. Vorona)

programming languages and software categories. Moreover, the KLOC metric is highly used in parametric models for SDEE, for example COCOMO, COCOMO II, SLIM, etc. [1,2].

Over the past 25 years, statistical data on software project success, reported by [3], indicate a moderate positive trend in the share of successfully completed projects. In 1994, only 16% of software projects were successfully implemented, meeting deadlines, aligning with budget, and achieving all requirements. By 2020, the share of successfully implemented projects had risen to 35%. Also, the percentage of failed projects reduced from 31% to 19% for the period from 1994 to 2020. Meanwhile, the share of challenged projects, which had issues with deadlines exceeding, budget planning, or failing to meet the declared requirements - has shown a minor downward trend of approximately 4%. Besides, the research [3] demonstrates a strong correlation between project size and the probability of failure or challenges and risks during software project development. Larger software projects are more difficult to manage because they require bigger resource allocation, wider coordination, and they have a higher probability of undefined obstacles. The CHAOS Report findings suggest that large software projects require more accurate estimation and risk management strategies to improve their chances of success.

The programming language JAVA is one of the most popular in software development [4] for diverse domains, including web-platforms, utility software, enterprise solutions, and information systems. The KLOC estimation of JAVA applications is necessary for effective project management to use resources efficiently, control costs, reduce risks, and increase confidence that JAVA applications are delivered on time and within budget.

Despite the rapid growth of the information technology industry, research on software project success rate shows that accurate SDEE remains a challenge. Recent studies indicate that improving the reliability and accuracy of application size estimation is possible by considering such environmental factors as a programming language when using parametric models such as COCOMO II, COSYSMO, and others [1,2] which are based on parameter of code lines. To ensure a high level of accuracy in application size estimation, it is essential to develop mathematical models that consider specific characteristics of programming languages, including JAVA.

2. Literature Review

There are regression equations and nonlinear regression (NR) models [5-12] have been constructed to estimate the size of JAVA applications on basis of open-source code metric samples. The models depend on certain code metrics that are possible to obtain from a UML class diagram, such as the total number of classes (CLASS), the total number of methods (visual, public, static, etc), the total number of class fields (private, public, protected, visual), CBO (coupling between objects), lack of cohesion (LCOM), response for class (RFC), etc. Different combinations of code metrics, sample size, quality, and representation of the general population impact differently on the accuracy, reliability, and robustness of application size estimation. Overall, the estimation process uses metrics of code from UML class diagrams, by applying regression models for KLOC estimation.

The detailed review of previously constructed models for JAVA-application size estimation is given in [10-12] on the total sample of JAVA application metrics with size 571 rows. The models [5-9] were compared by statistical quality assessment criteria for regression models, for instance the coefficient of determination R^2, a mean magnitude of relative error $MMRE$ and percentage of prediction for magnitude of relative error (MRE) level 0.25 $PRED(0.25)$ [13]. It shows the three-factor linear regression (LR) [5, 6] cannot be used because of the deprecated learning sample, non-Gaussian distribution of the learning and validation samples and heteroscedastic nature of the sample. The three-factor NR model [7] also uses the sample of software code metrics from [5, 6], the JAVA web-based one-factor NR model on the basis of CLASS metric [8], and a four-factor NR model on the basis of CLASS, SMQ (static methods quantity), LCOM and TFC [9], fail to exceed satisfactory accuracy according to the estimates of regression model quality criteria, or it's impossible to obtain estimates on a extended sample of the metrics because of the restrictions of normalizing functions.

The improved regression models for JAVA-application KLOC estimation are devoted in [10-12]. The models from [10] uses restricted set of independent factors which does not allow to exceed required accuracy thresholds. The three-factor NR model on the basis of CLASS, RFC, and aVMQ (average visual methods quantity) [11] and the four-factor NR model on the basis of CLASS, RFC, aCBO (average CBO), and aVMQ [12] have good accuracy by R^2, $MMRE$ and $PRED(0.25)$ criteria, but the models are based on RFC metric. The drawback of the RFC is that it requires knowledge about all the individual messages initiated by a given class, as this metric calls for the computation of all methods potentially executed in response to a message received by an object of a given class [14]. It requires information about internal class logic and algorithms, which UML class diagram does not provide, as it lacks information about internal method calls and algorithms, and the metric considers all types of methods - public, protected, and private. In comparison with public and protected methods, which are defined as part of the class interface for interaction, private methods and their relationships with other methods are typically determined under development rather than the system design stage.

The literature analysis proves, the estimating JAVA-applications size at the early stages of software project development is a demanding scientific and practical task, that proves necessity to construct an appropriate NR model to improve the estimation of the size of JAVA applications using appropriate normalization functions on the basis of quantitative software metrics, which should be available in UML class diagrams. It is necessary to compare the constructed model with the existing models [11, 12] using R^2, $MMRE$, and $PRED(0.25)$ quality criteria to prove the accuracy and reliability of JAVA application size estimation on the independent validation sample.

3. Objectives of the Research

The aim of the research is to improve the reliability and accuracy of JAVA-applications size estimation at the early stages of software development project planning on the basis of quantitative software metrics from a conceptual data model by constructing a NR model.

The object of the research is the process of JAVA-applications size estimation.

The subject of the research is NR models for estimating the size of JAVA applications.

4. Materials and Methods

4.1. Nonlinear Regression Model Building Methodology and Methods

JAVA code metrics generally have a non-Gaussian distribution, which restricts the possibility of using linear regression models for accurate estimation. A fundamental requirement for applying linear regression models is that the regression residuals ε must have a Gaussian distribution, and the sample must have homoscedastic nature.

One of the most efficient methodologies to process non-Gaussian data are based on applying a invertible normalizing transformation to convert the data into a normalized sample for further processing. Various methods can be used for this purpose, including decimal logarithm, square root transformation, univariate and multivariate Box-Cox transformations, and univariate or multivariate Johnson transformation. These normalization transformations allow the construction of linear regression models based on the transformed normalized data, which can be transformed into NR models through inverse transformation.

The NR model constructing methodology is based on statistical analysis methods [15] and it is based on detecting and removing anomalies in NR analysis of non-Gaussian sample and includes invertible normalizing transformation functions, squared Mahalanobis distance (SMD) anomalies detection method, verification of Gaussian distribution of the regression residuals ε, and detecting the forecast interval. The methodology recommends detecting and removing only one anomaly iteratively once it is detected. In case if an anomaly is detected, the methodology starts from the first stage using the modified sample without the detected anomalies from the previous iteration. Otherwise, the NR model is successfully built. The following stages are to construct the NR model.

The methodology begins with applying a normalizing transformation function to a non-Gaussian sample. There is the invertible normalizing function of a non-Gaussian random vector of the sample $P = \{Y, X_1, X_2, \ldots, X_k\}^T$ into a Gaussian random vector $T = \{Z_Y, Z_1, Z_2, \ldots, Z_k\}^T$ is given by

$$T = \psi(P), \tag{1}$$

where k is the number of regression factors, and the inverse function of (1) is given by

$$P = \psi^{-1}(T), \tag{2}$$

where ψ is a vector of invertible normalizing functions, $\psi = \{\psi_Y, \psi_1, \psi_2, \ldots, \psi_k\}^T$.

To normalize the multivariate non-Gaussian sample, we selected the multivariate invertible Box-Cox transformation, which is given by

$$Z_j = \begin{cases} (X_j^{\lambda_j} - 1)/\lambda_j, & if \lambda_j \neq 0 \\ ln(X_j), & if \lambda_j = 0 \end{cases}, \tag{3}$$

where λ_j - Box-Cox transformation function parameter; j changes from 1 to k; X_j – non-Gaussian random variable that is normalized; Z_j – Gaussian random variable.

The maximum likelihood method with logarithmic likelihood function was chosen to estimate the parameters of the normalization function (1) [16].

In the 2nd stage, the Gaussian distribution of normalized data is checked with the Mardia test [16] which uses the measurement of multivariate skewness ($\beta_{1,k}$) and kurtosis ($\beta_{2,k}$) of the multivariate sample.

$$\beta_{1,k} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[(X_i - \overline{X})^T S_N^{-1} (X_j - \overline{X}) \right]^3, \tag{4}$$

$$\beta_{2,k} = \frac{1}{N} \sum_{i=1}^{N} \left[(X_i - \overline{X})^T S_N^{-1} (X_i - \overline{X}) \right]^2, \tag{5}$$

where X is a k-dimensional vector of a random variable, $X = (X_1, X_2, \ldots, X_k)$ and S_N is a biased variance matrix of the multivariate random variable X which is given by

$$S_N = \frac{1}{N} \sum_{i=1}^{N} (X_i - \overline{X})(X_i - \overline{X})^T, \tag{6}$$

where $\overline{X}$ is a sample's means vector of the independent variable, $\overline{X} = (\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_k)^T$.

The test statistic for $\beta_{1,k}$ has the form

$$\tfrac{N}{6} \beta_{1,k} \leq \chi^2, \tag{7}$$

where χ^2 is an approximated value of the Chi-Square χ^2 distribution and with degrees of freedom $k(k+1)(k+2)/6$ and α is a significance level which is 0.005 for the test.

For $\beta_{2,k}$, the test statistic is the $1 - \alpha$ quantile of the normal distribution $\mathcal{N}$ with the math expectation $\mu = k(k+2)$ and variance $\sigma^2 = 8k(k+2)/N$

$$\beta_{2,k} \leq \mathcal{N}_{1-\alpha}(\mu, \sigma^2). \tag{8}$$

If the test conditions exceed expectations, the multivariate sample is considered as Gaussian. Otherwise, anomalies of the sample are detected using the SMD method [15] method. The SMD is elements on the main diagonal of the d^2 matrix with the size $N \times N$

$$d^2 = (Z_i - \overline{Z})^T S_N^{-1} (Z_i - \overline{Z}), \tag{9}$$

where S_N is a biased variance matrix of the sample (6), and Z is a Gaussian random variable. The elements of the main diagonal of $d_i^2, i = 1,2,\ldots,N$ matrix are detected as anomalies if the elements exceed the threshold of the Chi-Square χ^2 distribution quantile for the significance level α.

The 3rd stage is dedicated to a linear regression model constructing for the Gaussian sample which is given by the formula

$$Z_y = \hat{Z}_Y + \varepsilon = \hat{b}_0 + \hat{b}_1 Z_1 + \hat{b}_2 Z_2 + \ldots + \hat{b}_k Z_k + \varepsilon. \tag{10}$$

where ε is the Gaussian random variable, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$; $\hat{b}_0, \hat{b}_1, \hat{b}_2, \ldots, \hat{b}_k$ - the LR model estimators of parameters (10). The estimated values are calculated by the method of least squares.

In the 4th stage, the normality distribution of the LR residuals ε are checked with Pearson χ^2 criteria for significance level $\alpha = 0.01$. If the residual random variable is not normally distributed, one row Z_i should be removed with the biggest absolute value of the residual ε_i.

In the 5th stage, the NR model is obtained by applying the inverse transformation function to (2) to the LR models (10):

$$Y = \psi_Y^{-1}(\hat{Z}_Y + \varepsilon) = \psi_Y^{-1}(\hat{b}_0 + \hat{b}_1 \psi_1(X_1) + \hat{b}_2 \psi_2(X_2) + \ldots + \hat{b}_k \psi_k(X_k) + \varepsilon), \tag{11}$$

where ψ^{-1} is the inverse Box-Cox transformation function.

The 6th stage is dedicated to the forecast interval $\hat{Y}_{FI}$ constructing for the NR model (11). The forecast interval is based on the LR model (10) and is given by

$$\hat{Y}_{FI} = \psi^{-1}(\hat{Z}_Y \pm t_{a/2,v} S_{Z_Y} \{1 + \frac{1}{N} + (Z_X^{+})^T S_Z^{-1} (Z_X^{+})\}^{1/2}), \tag{12}$$

where $t_{\alpha/2,v}$ is a quantile of T-Student distribution with $v = N - k - 1$ degrees of freedom and $\alpha/2$ - significance level; $S_{Z_Y}^2 = \frac{1}{v} \sum_{i=1}^{N} \left(Z_{Y_i} - \hat{Z}_{Y_i}\right)^2$; Z_X^{+} is a vector of central moments of the sample' predictor variables, which is given by $\left\{Z_{1_i} - \overline{Z}_1, Z_{2_i} - \overline{Z}_2, \ldots, Z_{k_i} - \overline{Z}_k\right\}$; S_Z is $k \times k$ matrix

$$S_Z = \left[S_{Z_q} S_{Z_r}\right], \tag{13}$$

where $S_{Z_q} S_{Z_r} = \sum_{n-1}^{N} \left(Z_{q_i} - \overline{Z}_q\right)\left(Z_{r_i} - \overline{Z}_r\right)$, $q, r = 1, 2, \ldots, k$.

After the forecast interval is constructed, the predicted variable is checked if it is in the interval. If a dependent value is out of the forecast interval, it is considered as an anomaly and must be removed from the learning sample.

4.2. Collecting and Processing of JAVA-Applications Code Metrics

The authors have collected a dataset containing code metrics from 571 open-source JAVA applications hosted on the GitHub platform (https://github.com). The code metrics were extracted using the CK static code analysis tool (https://github.com/mauricioaniche/ck). The dataset includes the following metrics: lines of code per project (KLOC), total number of application classes (CLASS), string value of class types (TYPE), VMQ, TFQ, and CBO. The data is processed, and average values of the metrics per class are obtained, such as aVMQ, aCBO, aTFQ, etc. CK tool extracts abstract, open, final, inner classes, and interfaces under the one code metric name CLASS. All existing models [5-12] do not consider using actual quantities of classes and interfaces for the model building. Therefore, the general CLASS metric is split up into 2 different metrics: total number of actual classes (CLS) and total number of interfaces (INFC) by TYPE metric and all values of INFC metrics are incremented to make it suitable for Box-Cox transformation function. For the math model construction of JAVA-application code size estimation (KLOC), CLS, INFC, VMQ, TFQ, and CBO metrics are chosen as independent factors. The multivariate sample is randomly split up into learning and validation samples with sizes 286 and 285 multivariate points respectively. The considered metrics, except KLOC, can be obtained at the early stages of project planning from the conceptual data model of the application.

In the next stage, the multivariate learning sample was analyzed for multicollinearity using variance inflation factors (VIFs) to assess the relationships between independent variables. For multivariate data with k independent factors X_i where $i = 1, 2, \ldots, k$, the VIFs are represented in the diagonal elements of the inverse covariance $k \times k$ matrix. If a VIF value exceeds the threshold value of 10, it indicates a significant multicollinearity issue, while a value is close to 10, it suggests a potential risk of multicollinearity in mathematical model construction, in case of iterative anomalies removing [17]. For the CLS, INFC, VMQ, TFQ, and CBO factors, the calculated VIFs were 9.3, 2.4, 4.9, 7.2, and 18.6, respectively, confirming the presence of multicollinearity among certain independent factors. Using the NR model construction methodology described in Section 4.1 on the basis of Box-Cox normalizing function, the sample was normalized, and the anomalies were iteratively removed. The anomaly removal process was completed in 32 iterations, on the last iteration VIFs were 22.1, 4.3, 15.1, 11.6 and 31.2 which confirms that the VIFs for CLS, INFC, VMQ, TFQ, and CBO increase during the iterative anomalies' removal process. To avoid problems with multicollinearity, the absolute values of the metrics VMQ, TFQ, and CBO were replaced with their average per CLASS values: aVMQ, aTFQ, and aCBO. For the first iteration, the VIFs coefficients were 2.1, 2.3, 1.7, 1.1, and 1.6 for the metrics CLS, INFC, aVMQ, aTFQ, and aCBO respectively, confirming the absence of multicollinearity between the independent factors. Similarly, the sample was iteratively cleaned up from anomalies in 35 iterations. For the last iteration, the VIFs coefficients were 3.3, 3.3, 1.3, 1.3, and 1.3 confirming the absence of multicollinearity during iterative anomalies removing.

5. Experiment

5.1. Constructing the Five-Factor Nonlinear Regression Model

To enhance the accuracy of early JAVA-application KLOC estimation, a five-factor NR model is built by the above methodology from 286 (learning sample) applications hosted on Github. The model is constructed in 35 iterations using the multivariate learning sample of KLOC, CLS, INFC, aVMQ, aTFQ, and aCBO metrics using the multivariate Box-Cox normalizing function.

Before analyzing the six-dimensional learning sample for multivariate anomalies. We checked the Gaussian distribution of the multivariate data. Multivariate normality Mardia test were applied, which is based on multivariate skewness (β_1, k) and kurtosis (β_2, k). This test shows that, the distribution of the six-dimension data X_1 (CLS), X_2 (INFC), X_3 (aVMQ), X_4 (aTFQ), X_5 (aCBO), and Y (KLOC) of learning sample is not Gaussian, because multivariate skewness estimate $N\beta_1/6 = 12393.52$ is exceeded Chi-Square quantile value 86.99 for 56 degrees of freedom and significance level $\alpha = 0.005$ and the estimate of multivariate kurtosis $\beta_2 = 393.95$ is exceeded the value of Gaussian distribution quantile which is equal to 50.77 for mean 48 and variance 1.34.

We used the statistical methodology based on multivariate normalizing transformation and the SMD for normalized data. The six-variate Box-Cox transformation (3) was iteratively applied to the learning sample for normalizing. The parameter estimates of the six-variate Box-Cox transformation are calculated by the maximum likelihood method (4,5). There were 28 anomalies iteratively detected and removed from the learning sample using the SMD method because their d_i^2 values were greater than the threshold value 18.55 of the Chi-Square χ^2 for the significance level $\alpha = 0.005$. Once anomalies were found, an anomaly with max absolute value was removed, and the model constructing process started from the beginning. The estimates of the six-variate Box-Cox normalizing transformation are $\hat{\lambda}_Y = -9{,}165027 \cdot 10^{-3}$, $\hat{\lambda}_{X_1} = 7.384604 \cdot 10^{-3}$, $\hat{\lambda}_{X_2} = 2.758452 \cdot 10^{-2}$, $\hat{\lambda}_{X_3} = -7.881872 \cdot 10^{-2}$, $\hat{\lambda}_{X_4} = 0.377377$ and $\hat{\lambda}_{X_5} = 0.707875$ for the latest iteration.

On the latest iteration, the six-dimension learning sample ($N = 252$) checked Gaussian distribution by a multivariate normality Mardia test. Accordingly the test, distribution of six-dimension data is Gaussian, because the multivariate skewness estimate $N\beta_1/6 = 85.92$ is not exceeded Chi-Square quantile value 86.99 for 56 degrees of freedom and significance level $\alpha = 0.005$

and the estimate of the multivariate kurtosis $\beta_2 = 47.01$ does not exceed the value Gaussian distribution quantile which is equal to 50.86 for mean 48 and variance 1.23.

In the third stage, the five-factor linear regression is constructed (10) using the learning sample. The estimators $\hat{b}_0$, $\hat{b}_1$, $\hat{b}_2$, $\hat{b}_3$, $\hat{b}_4$, and $\hat{b}_5$ are calculated by the least square method and the estimates are -4.202982, 0.921816, $1.390376 \cdot 10^{-2}$, 0.933162, 0.231490 and $-2.686832 \cdot 10^{-2}$, respectively for the latest iteration.

And in the fourth stage the regression residuals ε were verified on the normality of the distribution in the LR model for 258 rows of the learning sample. The observed frequency distribution of the residual values in (10) resembles Gaussian distribution, after 3 residuals were removed from the sample, because the evaluated values of the Chi-Square test with 3 residuals were higher than quantile 15.09 of the Chi-Square distribution for 6 degrees of freedom and significance level $\alpha = 0.01$.

In the fifth stage, the NR model (11) is constructed by applying inverse transformation to (2) to the LR model (10). Then, in the sixth stage, the forecast interval is constructed of NR model by (12). In this stage, the 3 anomalies were detected and removed from the learning sample. For the latest iteration, the inverse covariance matrix of (13) is

$$S_Z^{-1} = \begin{vmatrix} 1.4271 \cdot 10^{-2} & -1.0049 \cdot 10^{-2} & -1.9412 \cdot 10^{-3} & -1.8556 \cdot 10^{-3} & -8.7586 \cdot 10^{-4} \\ -1.0049 \cdot 10^{-2} & 8.9002 \cdot 10^{-3} & -3.9592 \cdot 10^{-4} & 2.2489 \cdot 10^{-3} & 4.8178 \cdot 10^{-4} \\ -1.9412 \cdot 10^{-3} & -3.9592 \cdot 10^{-4} & 7.2923 \cdot 10^{-2} & -1.2086 \cdot 10^{-2} & -6.0266 \cdot 10^{-3} \\ -1.8556 \cdot 10^{-3} & 2.2489 \cdot 10^{-3} & -1.2086 \cdot 10^{-2} & 2.0962 \cdot 10^{-2} & -3.6828 \cdot 10^{-3} \\ -8.7586 \cdot 10^{-4} & 4.8178 \cdot 10^{-4} & -6.0266 \cdot 10^{-3} & -3.6828 \cdot 10^{-3} & 7.5035 \cdot 10^{-3} \end{vmatrix}$$

For the obtained forecast interval, the values of normalized sample mean $\overline{Z}_1$, $\overline{Z}_2$, $\overline{Z}_3$, $\overline{Z}_4$, and $\overline{Z}_5$ are 6.59985, 4.12350, 1.46601, 0.90387, and 3.58194, respectively. The $t_{\alpha/2,v} = 2.5960$ for significance level $\alpha = 0.01$ and 246 degrees of freedom; $S_{Z_Y} = 0.154424$.

The constructed NR model is limited to estimating KLOC JAVA-application with the following restrictions on factors: the interval for $X_1$is from 33 to 10610, X_2 is from 1 to 1562, X_3 is from 2.34 to 12.14, X_4 is from 0.5384 to 5.64 and X_5 is from 2.1 to 10.43.

The constructed model's accuracy is verified with R^2, $MMRE$, and $PRED(0.25)$ regression model quality criteria on the learning sample ($N = 252$) without 34 anomalies, which were detected and removed during the iterative model constructing process. The estimates of R^2, $MMRE$, and $PRED(0.25)$ are 0.9759, 0.1276, and 0.9008 respectively, which indicates a high level of the prediction accuracy of the model.

Moreover, to estimate the size of JAVA-applications, we constructed a four-factor NR with factors of total classes and interfaces number X_1 (CLASS), X_2 (aVMQ), X_3 (aTFQ), and X_4 (aCBO) on the basis of multivariate Box-Cox normalizing transformation for the same learning sample. The NR model with four factors is based on the multivariate Box-Cox transformation, and has the form (11) but with the following estimates of the parameters: $\hat{\lambda}_Y = 4.072805 \cdot 10^{-3}$, $\hat{\lambda}_{X_1} = 4.1431 \cdot 10^{-2}$, $\hat{\lambda}_{X_2} = -0.143596$, $\hat{\lambda}_{X_3} = 0.334377$, and $\hat{\lambda}_{X_4} = 0.651922$ and parameters of four-factor linear regression model for the normalized sample are $\hat{b}_0 = -3.918828$, $\hat{b}_1 = 0.782652$, $\hat{b}_2 = 1.037247$, $\hat{b}_3 = 0.283308$, and $\hat{b}_4 = -4.834321 \cdot 10^{-2}$.

For the latest iteration, the inverse covariance matrix of (13) is given by

$$S_Z^{-1} = \begin{vmatrix} 1.8518 \cdot 10^{-3} & -2.7074 \cdot 10^{-3} & 8.7934 \cdot 10^{-4} & -3.3994 \cdot 10^{-4} \\ -2.7074 \cdot 10^{-3} & 9.8625 \cdot 10^{-2} & -1.6543 \cdot 10^{-2} & -8.4414 \cdot 10^{-3} \\ 8.7934 \cdot 10^{-4} & -1.6543 \cdot 10^{-2} & 2.2426 \cdot 10^{-2} & -3.8873 \cdot 10^{-3} \\ -3.3994 \cdot 10^{-4} & -8.4414 \cdot 10^{-3} & -3.8873 \cdot 10^{-3} & 9.1539 \cdot 10^{-3} \end{vmatrix}$$

For the obtained forecast interval, the values of normalized sample mean $\overline{Z}_1$, $\overline{Z}_2$, $\overline{Z}_3$, and $\overline{Z}_4$ are 7.433473, 1.385643, 0.883213, and 3.356313, respectively. The $t_{\alpha/2,v} = 2.5955$ for significant level $\alpha = 0.01$ and 251 degrees of freedom; $S_{Z_Y} = 0.169793$.

The constructed model's accuracy was verified with R^2, $MMRE$, and $PRED(0.25)$ regression model quality criteria on the learning sample without 30 anomalies, which were determined and excluded during the model constructing process. The estimates of R^2, $MMRE$, and $PRED(0.25)$ are 0.9719, 0.1345, and 0.8750 respectively which indicates a high prediction accuracy of the model.

5.2. Comparing the Quality and the Accuracy of Size Estimation of the Regression Models

The obtained five-factor NR model is verified with the regression model's quality criteria to assess the predictive accuracy and reliability using the initial learning and the validation samples, and compared with existing three-factor NR [11] and four-factor NR on the basis of RFC [12]. The estimates of R^2, $MMRE$, and $PRED(0.25)$ are displayed in Table 1.

Table 1
Comparison of NR models by quality criteria

NR models	Learning sample			Validation sample		
	R^2	MMRE	PRED	R^2	MMRE	PRED
Three-factor NR on RFC basis [11]	0.9073	0.1645	0.7692	0.9016	0.1617	0.8175
Four-factor NR on RFC basis [12]	0.8242	0.1621	0.8042	0.8981	0.1536	0.8211
Four-factor NR (11)	0.9303	0.1713	0.8146	0.9177	0.1821	0.7579
Five-factor NR (11)	0.9257	0.1536	0.8427	0.9185	0.1757	0.7719

The constructed five-factor model validation accuracy estimates of $MMRE$ and $PRED(0.25)$ for the initial learning sample and R^2 for the validation sample demonstrate that the usage of the average values of VMQ, TFQ, and CBO metrics allows us to achieve better results in estimating the KLOC of JAVA-applications in comparison to all existing and previously constructed regression models which use RFC metric. The remaining R^2 $MMRE$ and $PRED(0.25)$ estimates prove the good quality of the constructed five-factor regression mode.

The forecast interval of the five-factor regression is compared with the four-factor regressions. The comparison indicates the five-factor NR interval is 21.4% shorter than the RFC based four-factor NR forecast interval and is 9.8% shorter than the four-factor NR (CLASS, aVMQ, aTFQ, aCBO) (12) on the basis of the learning sample. The forecast intervals are visualized in Figure 1, comparison of the four-factor nonlinear model on RFC basis, the four-factor nonlinear model (11), and the five-factor nonlinear model (11).

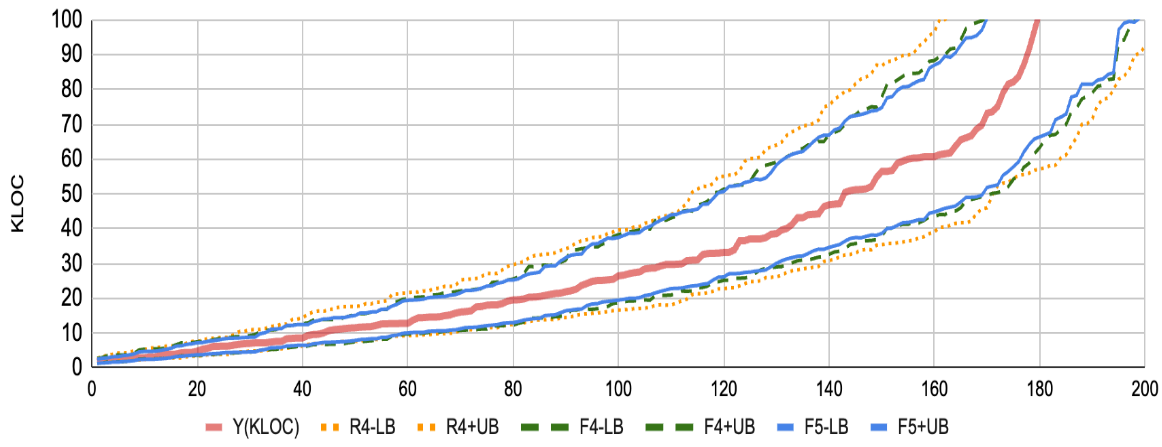

Figure 1: Lower and upper bounds on the compared regression models.

The forecast interval estimates are sorted ascendingly order. The lower and upper bounds are dotten lines with names R4-LB and R5+UB for four-factor regression on the RFC basis, are dashed lines with names F4-LB and F4+UB for four-factor regression (11), and are solid lines with names F5-

LB and F5+UB for five-factor regression (11). Actual JAVA-application KLOC size is solid red line. The chart confirms that the width of the constructed five-factor nonlinear regression is shorter in comparison with the existing models.

6. Conclusion

The obtained five-factor nonlinear regression model on the basis of multivariate Box-Cox normalizing transformation significantly enhances the accuracy and reliability of JAVA-applications size estimation compared to existing models by using five independent factors: CLS, INFC, aVMQ, aTFQ, and aCBO, which are available in UML class diagram. The model is built without the quality metric RFC, which cannot be accurately obtained from the UML class diagram on the early stage of the project planning. Splitting the total sum of classes and interfaces into separate metrics allows to improve the accuracy of the model by R^2, $MMRE$ and $PRED(0.25)$ criteria estimates from the learning and the validation samples compared to three- and four-factor regression models. The estimates of R^2, $MMRE$, and $PRED(0.25)$ exceed the thresholds, and there are 0.9257, 0.1536, and 0.8427 respectively for the initial learning sample and 0.9185, 0.1757, and 0.7719 for the validation sample, and the forecast interval is 21.4% shorter in comparison with the four-factor nonlinear regression on the basis of CLASS, RFC, aVMQ and aCBO metrics and 9.8% shorter compared to the four-factor nonlinear regression on the basis of CLASS, aVMQ, aTFQ, and aCBO metrics, which indicates good accuracy and reliability of the model.

The scientific novelty of the research is that the five-factor nonlinear regression model for multivariate non-Gaussian data is firstly constructed using Box-Cox six-variate transformation using quantitative metrics, which are available on early project planning stage from UML-class diagram. Firstly, the total number of class entities is split up into the actual number of classes and the number of interfaces for regression model construction. Firstly, the multicollinearity issue is solved in the iterative regression model constructing process on the basis of CLS, INFC, and averages of VMQ, TFQ, CBO. The model, compared to other nonlinear regression models, has a higher value of determination coefficient R^2, a lower values of the mean relative error $MMRE$, higher values of the $PRED(0.25)$, and the forecast interval is 21.4% shorter in comparison with the four-factor nonlinear regression on the basis of CLASS, RFC, aVMQ and aCBO metrics and 9.8% shorter compared to the four-factor nonlinear regression on the basis of CLASS, aVMQ, aTFQ, and aCBO metrics. The findings demonstrate that, splitting up the sum of classes and interfaces metric into 2 separated metrics allows to build nonlinear regression models with a higher accuracy and reliability, accordingly to the estimates of the quality criteria and forecast interval width.

The practical significance of the obtained results allows us to recommend the obtained model for use in practice for further software development effort estimation using parametric models such as COCOMO II, COSYSMO, etc. The proposed model is implemented as a software service that can be used by project managers for JAVA-applications SDEE at early stages of project planning to reduce costs and manage risks.

Prospects for further research. The obtained five-factor NR model could be improved by extending learning sample with code metrics from commercial projects.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o and Grammarly tools in order to: Grammar and spelling check. After using these tool(s)/service(s), the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. McConnel, Software Estimation: Demystifying the Black Art, Microsoft Press, Redmond, Washington, USA, 2006.

[2] S. W. Munialo, A Review of Agile Software Effort Estimation Methods, volume 5 of International Journal of Computer Applications Technology and Research., Association of Technology and Science, 2016, 612-618. doi:10.7753/IJCATR0509.1009.

[3] The Standish Group, Chaos report 2015, 2015. URL: https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

[4] TIOBE, TIOBE Index, 2024. URL: https://www.tiobe.com/tiobe-index/.

[5] H. B. K. Tan, Y. Zhao, H. Zhang, Estimating LOC for information systems from their conceptual data models, in: Proceedings of 28th. International Conference on Software Engineering, 2006, pp. 321-330. doi:10.1145/1134285.1134331.

[6] H. B. K. Tan, Y. Zhao, H., H. Zhang, Conceptual Data Model-Based Software Size Estimation for Information Systems, volume 19 of ACM Transactions of Software Engineering and Methodology, 2009. doi:10.1145/1571629.1571630.

[7] N. V. Prykhodko, S. B. Prykhodko, A non-linear regression model for estimation of the size of JAVA enterprise information systems software, volume 85 of Modeling and Information Technologies, 2018, pp. 81-88. URL: http://nbuv.gov.ua/UJRN/Mtit_2018_85_14.

[8] L. M. Makarova, N.V. Prykhodko, O. O. Kudin, Constructing the non-linear regression model for size estimation of WEB-applications implemented in JAVA, volume 69 of Herald (Kherson National Technical University), 2019, pp. 145-153.

[9] S. B. Prykhodko, N. V. Prykhodko, T. G. Smykodub, Four-factor nonlinear regression model to estimate the size of open source JAVA-based applications, volume 70 of Scientific Notes of Taurida National V.I. Vernadsky University. Series: Technical Sciences, 2020, pp. 157-162. doi:10.32838/2663-5941/2020.2-1/25.

[10] O. Oriekhov, T. Farionova, Mathematical models for the size estimating of JAVA applications, volume 89 of Herald (KNTU), 2024, pp. 196-203, doi:10.35546/kntu2078-4481.2024.2.28.

[11] O. Oriekhov, T. Farionova, L. Chernova, Three-factor nonlinear regression model of estimating the size of JAVA-software, in Proceeding of 12th. Information Control Systems & Technologies, Odesa, Ukraine, 2024. URL:https://ceur-ws.org/Vol-3790/paper44.pdf.

[12] O. Oriekhov, The four-factor nonlinear regression model for early JAVA-applications size estimation, in: N. Aksak, D. Antonov, ICST-2024: Advances in Information Control Systems and Technologies, Liha Press, Lviv, Ukraine, 2024, pp. 360-379. doi:10.36059/978-966-397-422-4.

[13] D. Port, M. Korte, Comparative studies of the model evaluation criterions MMRE and PRED in software cost estimation research, Proceedings of the 2nd. ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, New York, 2008, pp. 51–60. doi:10.1145/1414004.1414015.

[14] R. Subramanyam, M. Krishnan, Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects, volume 29 of IEEE Transactions on Software Engineering, pp. 297- 310. doi:10.1109/TSE.2003.1191795.

[15] S. Prykhodko, N. Prykhodko, Mathematical Modeling of Non-Gaussian Dependent Random Variables by nonlinear Regression Models Based on the Multivariate Normalizing Transformations, in S. Shkarlet, A. Morozov, A. Palagin, volume 1265 of Mathematical Modeling and Simulation of Systems (MODS'2020). Advances in Intelligent Systems and Computing, 2021, pp. 166-174. doi:10.1007/978-3-030-58124-4_16

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

[17] I. Olkin, A. R. Sampson, Multivariate Analysis: Overview, in N. J. Smelser, P. B. Baltes, International encyclopedia of social & behavioral sciences, 1st. ed., Elsevier, Pergamon, 2001, pp. 10240–10247.