International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 12, No.1 (Aug-2022)

https://dx.doi.org/10.12785/ijcds/120140

Estimating Java Classes Reusability Using Data Clustering and Static Metrics Analysis

Mariam Amin[1] **and Mustafa Hammad**[1,2]

[1]*Department of Computer Science, University of Bahrain, Sakheer, Kingdom of Bahrain*
[2]*Department of Software Engineering, Mutah University, Al-karak, Jordan*

Received 6 Dec.2021, Revised 10 Mar. 2022, Accepted 15 Jul. 2022, Published 6 Aug. 2022

Abstract: To reuse or not to reuse, that is the question! There is a constant demand in the software development market for new methodologies and practices that would help to lessen the pressure from the software development projects. Software reuse is one practice that helped increase software development productivity and improve the software quality attributes. However, until this day, there is no silver bullet solution that could predict software reuse experience nor estimate software component's reusability. This work proposes a reusability estimation model. The model analyzes the static metric of java classes using data clustering and regression to estimate its reusability. The evaluation results of the proposed model have successfully evaluated java classes reusability with considerably low error rates.

Keywords: Software Reusability, Regression, Software Reuse, Static Metric, Reusability Estimation, Data Clustering

1. Introduction

The widespread use of Open Source Software (OSS) has significantly changed the software development methodologies adopted by software developers. Furthermore, the high availability of OSS has contributed to introducing agility to software development [1]. Nowadays, software developers have replaced developing the whole software from scratch with reusing pre-existed software components. As a result, the software developers started to rely more and more on software reuse. Adapting software reuse has helped reduce the software project schedule and duration. Also, it helped reduce the project cost. However, software reuse has introduced new challenges. One of these challenges is assessing the software component's reusability, especially with the significant growth of OSS.

The assessment of software component reusability is very crucial. The wrong reused component can negatively impact the progress of the software development project and its success. Also, it negatively affects the quality attributes of the developed software. Moreover, retrieving, assessing, and selecting the required software component requires a lot of effort and time. In addition, software developers' skills and experience play a role in the assessment phase.

Therefore, there is a constant demand for assessment models that assess software components without relying on software developers' expertise. This work proposes a model that evaluates the reusability of java classes. The proposal of this model came after studying and analyzing selected static classes metrics using feature selection, data clustering, and regression. The rest of this work is structured as follows. Section 2 reviews some literature related to software reusability assessment models and the utilization of machine learning. Followed Section 3, which presents the used dataset and illustrates the evaluation criteria used to evaluate the performance of machine learning algorithms. The following section that is Section 4, presents the modeling is the proposed model. Then Section 5 evaluates the proposed model. Section 6 presents some threats that may compromise the validity of this study. Finally, Section 7 concludes this work and highlights the future work too.

2. Related Work

Machine learning was leveraged to estimate software reusability. Kaur and Kaushal [2] used fuzzy logic to evaluate the reusability of Aspect-Oriented Systems (AOS) at the package level. The authors validated the proposed model used by analyzing its predictions. The predictions analysis showed that the model performed correctly. The work in [3] selected seven machine learning algorithms to estimate reusability utilizing data, which were extracted from social media repositories system. The experimental results showed that the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) had the highest precision results. A fuzzy logic approach was proposed in [4], which measures the reusability of java classes by assessing their static and dynamic metrics. The findings

E-mail address: [1]*19980608@stu.uob.edu.bh,* [2]*mhammad@uob.edu.bh*

concluded that the dynamic metrics are considered a better reusability prediction factors than the static metrics. Mangayarkarasi [5] proposed an automated approach that could select reusable components by analyzing the software design pattern. The author used two machine learning algorithms. These algorithms are Decision Tree and Neural Network. The study's outcome concluded that the proposed approach is effective with an accuracy of about 93%. In [6] machine learning was leveraged to predict successful software reuse. The empirical results showed that machine leaning have successfully identified successful software reuse with accuracy reached 100%. Also, The work in [7] introduced feature selection techniques in an attempt to improve the prediction of software reuse. They concluded to that feature selection have promoted the prediction model accuracy.

In addition to that, reusability estimation models were proposed for all software reuse approaches. Wangoo and Singh [8] proposed a model (CRePT) that uses K-Nearest Neighbour (KNN) to predict the reuse cost and reuse level for Component-Based Software Development (CBSD). The author evaluated reuse level by the ratio of sizes of total reused components and the application components. The proposed model did not report any defects. However, the precision of the model had not been evaluated. Maggo and Gupta [9] proposed an automated model that assesses and measures the reusability of Object-Oriented (OO) Software Components. The model uses Large Margin Nearest Neighbor (LMNN) to identify the correlation between software metrics and reusability. The evaluation of the proposed model accuracy has reached about 94.2% and an error rate of 5.8%. The work in [10] conducted an exploratory study to identify metrics, which could impact Aspect-Oriented Software Development (AOSD) by utilizing fuzzy logic. The study identified four primary metrics that could affect reusability. These metrics are coupling, size and complexity, Separation of Concern (SoC), and cohesion.

OO Metrics (OOM)s is a quantitative source code measuring technique. Deshpande et al. [11] proposed a machine learning classifier that utilizes seven machine learning algorithms and OOMs to predict software reliability. The evaluation proposed classifier showed that Random Forest had scored the best performance results. Also, the work in [12] used OOM and proposed an estimation model that measures the complexity of C++ and Python software projects. The authors analyzed the OOM of forty C++ and Python programs. The findings showed that Python is more suitable for OO Programming (OOP), and it had a better OOM than C++. Kaur and Sharma [13] experimented prediction of software faults with Artificial Neural Network (ANN). Experimental results showed that the proposed ANN model had scored an accuracy of about 92.5% and performance reached 0.92. Malhotra and Khanna [14] proposed a model that recognizes classes prone to changes by analyzing the OOM and evolution-based metrics. The authors introduced the collected dataset from two Android packages changes logs to seven machine learning models. Also, the experiments evaluated the prediction of the class change prone in four scenarios. The scenarios evaluated the prediction using evolution-based metrics solely, OOM solely, OOM and evolution-based metrics in conjunction, and the classes' internal probability of changes. The experimental results showed that combining OOM and evolution-based metrics had the best prediction results.

Various software reusability estimation techniques assessed Chidamber & Kemerer (CK) metrics. For instance, Padhy et al. [15] proposed a model that utilizes Artificial Neural Networks (ANN), Adaptive Genetic Algorithm (AGA), Extreme learning machines (ELM), Levenberg Marquardt (LM), and CK metrics to estimate the reusability of Web of Service (WoS) software. The evaluation showed that the proposed techniques had outperformed other reusability prediction techniques with an accuracy of 97.71%. The work in [16] used clustering and CK metrics to predict software reusability. The developed prediction model scored high precision results. A comparative study in [17] evaluated the software reusability by assessing CK metrics using four regression algorithms. The experimental results should that a tuned Instance Based K-Nearest Neighbour (IBK) had the best error rate and correlation coefficient. Negi and Tiwari [18] trained the Random Forest algorithm on CK metrics to assess reusability software components. The experimental results concluded that the Random Forest algorithm has a high potential in assessing components' reusability with accuracy reaching about 99% and a correlation coefficient of about 0.85.

Several pieces of literature have tackled software reusability estimation by analyzing software metrics other than CK metrics. Mahmood et al. [19] proposed a software reusability assessment model for Software Product Lines (SPLs). The authors used the Goal, Question, Metric (GQM) approach to extract the input for the proposed reusability assessment model. The evaluation results showed that the proposed assessment had performed very well. Also, the evaluation summed up that the selected attributes have a significant impact on software reusability. The work in [20] has estimation software components reusability for java classes and packages using binning and polynomial regression. The authors collected a dataset that has 28 attributes. These attributes were grouped into six primary groups. These groups are coupling, cohesion, documentation, size, complexity, and inheritance. The evaluation of the estimation model showed that the proposed model had outperformed the estimation model in [19].

Many pieces of literature have proposed GQM models for software quality evaluation. The work in [21] proposed a GQM model that identified the software metrics that could be used to evaluate adaptive systems quality. The goal of the proposed model focuses on the success of completing the project and its phases. Also, improve the development's processes quality, adapt effective testing methodologies,

and improve the effort assessment process. Ito et al. [22] proposed a requirements analysis approach that combined the GQM model and requirements analysis techniques. The GQM was utilized to pinpoint current software issues and identify the metrics to solve these issues. The evaluation of the proposed approach proved the effectiveness of the GQM in identifying the problematic areas and developing appropriate metrics to resolve these problems. In [23], GQM model and machine learning were used to explore the relationship between source metrics and software quality. Also, the authors proposed a quality assessment model based on the extracted metrics. The proposed model was about 75% accurate.

3. Research Background

The used dataset was collected by Papamichail et al. [24]. This dataset was generated by analyzing the most popular project in the maven repository. The dataset consists of static metrics of java classes and packages in addition to the reuse rate of these components. However, this work will focus on the reusability of java classes only. Therefore, the packages' dataset will be neglected. The primary aim of this work is to explore the correlation between the classes' static metrics and their reuse rate. Table I presents the attributes of the used dataset. According to dataset has 28 attributes. All these attributes are numeric. The attribute reuse rate is the primary attribute, which the estimation reusability model will based on. Also, the dataset attributes were classified into six properties. These properties are complexity, size, cohesion, inheritance, coupling, and documentation. Moreover, the dataset has 24794 instances.

Furthermore, in this work, we have utilized machine learning algorithms. Therefore, it is required to use some statistical evaluation criteria to evaluate the performance of the used machine learning algorithms. The performance of the data clustering algorithm was assessed using the Sum of Squared Error (SSE). SSE presents the summation of the squared distance between the observation and the cluster centroid [25]. SSE can be obtained according to the following formula:

$$SSE = \sum_{s=1}^{b} \sum_{j \in C_s} (j - R_j)^2 \quad (1)$$

where b is the number of desired clusters, Cs is the cluster, Rj is the centroid of the cluster Cs and j is the observed data point.

Meanwhile, the Support Vector Regression (SVR) performance was evaluated using the three evaluation criteria. These criteria are the following: noitemsep

- Mean Absolute Error (MAE): This error rate assesses the regression model performance by measuring the differences between the predicted and actual values of the test instances [26]. The given equation computes the MAE

$$MAE = \sum_{s=1}^{j} |y_s - y\prime_s| / j \quad (2)$$

where ys is the actual value, y's is the predicted value, and j is the size of the sample.

- Root Mean Square Error (RMSE): Advantages of the MAE over the root mean square error RMSE in assessing average model performance. However, the RMSE measures the square root of the squared difference total between the predicted and actual values [27]. The following formula calculates the RMSE

$$RMSE = \left[\sum_{s=1}^{j} (y_s - y\prime_s)^2 / j \right]^{-1/2} \quad (3)$$

where y_s is the actual value, y'_s is the predicted value, and j is the size of the sample.

- Pearson Correlation Coefficient (PCC):
This coefficient measures the linear correlation between pair of variables of the same size. The value

TABLE I. Attributes of The Dataset Used in Java Classes' Reusability Estimation

Property	Attribute Name and Description
Size	TLOC: total number of line of codes TLLOC: total number of logical lines LOC: number of line codes LLOC: number of logical lines NG: number of getters TNA: number of attributes TNG: total number of getters TNM: number of methods TNOS: number of statements TNPM: number of public methods
Cohesion	LCOM5: lack of cohesion in methods
Documentation	AD: documentation of the API CD: the density of comments TCD: the total of comments density CLOC: number of comments lines TCLOC: total number of comments lines DLOC: number of documentation lines PDA: total lines of documentation
Coupling	CBO: coupling between the classes object CBOI: inverse coupling between the classes object NII: number of incoming invocations NOI: number of outgoing invocations RFC: response set for class
Inheritance	DIT: the inheritance tree depth
Complexity	NL: maximum level of nesting NLE: maximum level of else- if nesting WMC: weighted methods for each class
Reuse Information	Reuse rate: extent which the developers select the component for reuse

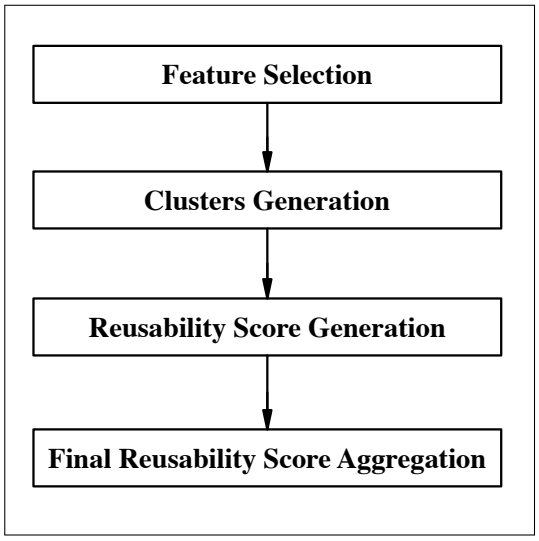

Figure 1. Model Design Flow Chart

of the PCC could vary between -1 and 1. The sign of the PCC represents the direction of the correlation. Negative PCC indicates a negative correlation, while the positive indicates a positive correlation. Moreover, the PCC value describes the correlation strength. The closer the PCC value to 1 or -1, the stronger the correlation. Meanwhile, the correlation gets weaker when the PCC value is closer to 0 [28],[29].

4. Reusability Estimation Modeling

This section discusses the design of the proposed reusability estimation model. This model has two foundations elements which are the static metrics and reuse rate. Also, the model consists of four steps. These steps are feature selection, clusters generation, reusability score generation, reusability estimation model construction, as shown in Figure 1. The following subsection presents the details of each step.

A. Feature Selection

In this work, we selected the software metrics using the GQM paradigm. Figure 2 presents the GQM model that was used to select the static metrics. The goal was to assess the reusability of java classes by analyzing the classes' static metrics. The GQM questions focus on the components' variability, size, dependability, and complexity, as shown in Figure 2. A similar model was used in [19]; however, we could not use the same due to the difference in the dataset attributes. Software component documentation influences its reusability. Well-structured and clear documentation helps the software developers while analyzing and understanding the software component. As a result, proper component documentation could increase its reusability. Also, a high cohesive class indicates that it has good modularity. Hence cohesive promotes simplicity; therefore, it has a positive impact on the class reusability. Meanwhile, high coupling enables class complexity which has a negative influence on the components' reusability. As a result, complex classes are more likely to be non-reusable [30]. Furthermore, software components with a deep inheritance tree are more complicated since the high involvement of methods and classes. However, a high Depth of Inheritance Tree (DIT) produces more reusable software components [31].

Goal: Measure java classes reusability from the viewpoint of the software developer.
- **Question:** How cohesive are the class elements?
  - **Metrics:** LCOM5
- **Question:** How complex is the java class?
  - **Metric:** NL, WMC
- **Question:** Is the component tightly or loosely coupled?
  - **Metric:** CBO
- **Question:** Is the class well documented?
  - **Metrics:** TCLOC
- **Question:** How independent is the class?
  - **Metric:** DIT
- **Question:** What is the size of the class?
  - **Metric:** TLOC, TNOS
- **Question:** What is the variability of the class?
  - **Metric:** TNM, TNA, TNPM
- **Question:** To what extent the class is imported?
  - **Metric:** Reuse Rate

Figure 2. Selected Features Using GQM Model

TABLE II. Reuse Rate Statistical Information

Reuse Rate Interval	Number of instances	Percentage
0	17932	72.320%
[1,10)	5468	22.054%
[10,20)	546	2.202%
[20,30)	223	0.899%
[30,40)	145	0.585%
[40,50)	92	0.371%
[50,60)	60	0.242%
[60,70)	46	0.186%
[70,80)	37	0.149%
[80,90)	29	0.117%
[90,100)	21	0.085%
[100,1000)	186	0.750%
[1000,2000)	8	0.032%
>=2000	1	0.004%

B. Clusters Generation

The used dataset consists of independent attributes. Therefore, an exploratory data analysis helps to understand how these attributes and instances are related. The primary aim of the exploratory data analysis is to identify how reuse rate is related to the remaining dataset's attributes. Furthermore, the dataset has a wide range of reuse rates that are between 0 and 3964. Table II depicts a statistical summary of the reuse rates. According to Table II, about 72% of the dataset instances have a zero reuse rate. Meanwhile, the number of instances with the highest reuse rate is 1, about 0.36% of the dataset.

Moreover, there are similarities between static metrics regardless the reuse rates are high or low. Table III illus-

TABLE III. Comparison of Two Dataset Instances

Metric	Metric Value	Metric Value
LCOM5	1	1
NL	3	3
WMC	62	68
CBO	9	5
TCLOC	104	93
DIT	0	1
TLOC	372	343
TNA	26	11
TNM	23	43
TNOS	129	132
TNPM	3	36
Reuse Rate	**3964**	**0**

trates the static metrics of two instances, one with 0 reuse rate while the other instance has the highest reuse rate of 3964. The two instances almost have the same static metrics values. However, their reuse rate is different. Therefore, the is a need to group the dataset instances based on reuse rate to avoid the effect of similarity between the static metrics and the gap between the reuse rates while generating the reusability scores.

Data clustering was utilized to recognize the similarity patterns among the dataset instances. Also, data clustering would help to prevent any negative interference of unrelated instances. In this work, a density-based clustering algorithm was used. Many clustering algorithms assume that a particular type of probability distribution was used to generate the data. As a result, these algorithms produce spherical shape clusters. Also, density-based clustering can identify and eliminate outliers and noises [32].

The used dataset was clustered using the WEKA python. The algorithm was set to act on the reuse rate. The algorithm generated ten clusters with a Sum of Squared Error (SSE) of 0.0433. Table IV presents the dataset clusters and the number of instances their percentage. Cluster 1 has about 86% of the dataset instances; meanwhile, cluster 5 has only one instance shown in Table IV. Moreover, Cluster 2 and Cluster 5 were merged into one cluster while establishing the reuse score since Cluster 5 has one instance.

C. Reusability Score Generation

The second step is to translate the metric values of each cluster into a reusability score. The estimation of the reusability score is based on the distribution reuse rate over the full range of the static metrics. The following steps were applied for each metric in a cluster to generate the reusability score.

First, we assigned a reusability score which is the sum of the reuse rate of the java classes. Figure 3 depicts the reuse score of the LCOM5 metric in Cluster 8. According to Figure 3, LCOM5 of value 1 has the highest reuse

TABLE IV. Dataset Instances Distributed Among Clusters

Cluster #	Number of Instances	Percentage
1	21244	85.682%
2	12	0.048%
3	278	1.121%
4	51	0.205%
5	1	0.004%
6	141	0.056%
7	2497	10.070%
8	532	2.145%
9	15	0.060%
10	23	0.092%

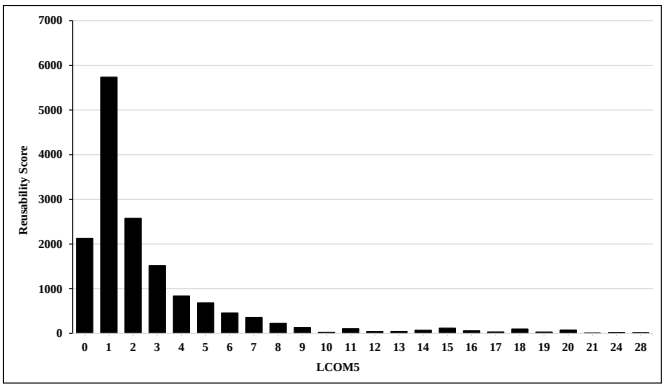

Figure 3. Reusability Score Distribution of LCOM5 in Cluster 8

score while LCOM5 of 21 has a reusability score of 5. Theoretically, low LCOM5 indicates good cohesion, which promotes the component's reusability. Figure 5 showed that low LCOM5 values have higher reusability scores, which aligns with the theoretical interpretation of LCOM5. However, the reason that LCOM5 with 0 value has a lower reusability score is due to the number of dataset instances.

Then, we rescaled the reusability scores after assigning the reuse score for the metric by min-max normalization. The vales of the normalized reusability scores all fall within the interval [0,1]. The normalization would unify the value of the reusability scores for all metrics of all clusters. The normalized reusability score was calculated using the following equation:

$$NRS = \frac{(RS - RS_{min})}{(RS_{max} - RS_{min})} \quad (4)$$

where RS is the reuse score, RS_{min} is the minimum reuse score, RS_{max} is the maximum reuse score.

Lastly, Support Vector Regression (SVR) was applied to the obtained dataset with the static metric value and the reusability score. The SVR utilizes the Support Vector Machine (SVM) to generalize the optimal tube shape training instances with the best continuous values outputs while reducing the model's error rates and complexity [33]. Tables

TABLE V. SVR Results of Cluster 1, Cluster 2 & 5, and Cluster 3

Metric	Cluster 1			Cluster 2 & Cluster 5			Cluster 3		
	MAE	RMSE	PCC	MAE	RMSE	PCC	MAE	RMSE	PCC
LCOM5	0.043	0.140	0.79	0.219	0.313	0.56	0.044	0.141	0.62
NL	0.046	0.087	0.98	0.244	0.324	0.61	0.044	0.051	1.00
WMC	0.038	0.092	0.90	0.127	0.238	0.67	0.095	0.143	0.62
CBO	0.040	0.087	0.95	0.264	0.424	0.36	0.087	0.147	0.80
TCLOC	0.021	0.077	0.82	0.120	0.237	0.37	0.122	0.177	0.44
DIT	0.049	0.057	0.99	0.138	0.270	0.94	0.090	0.189	0.90
TLOC	0.038	0.079	0.89	0.121	0.253	0.01	0.078	0.129	0.42
TNA	0.039	0.090	0.93	0.182	0.249	0.82	0.050	0.102	0.81
TNM	0.034	0.092	0.87	0.131	0.248	0.29	0.082	0.148	0.59
TNOS	0.024	0.070	0.90	0.118	0.228	0.74	0.076	0.127	0.56
TNPM	0.035	0.101	0.88	0.122	0.265	0.09	0.079	0.131	0.71

TABLE VI. SVR Results of Cluster 4,6,7

Metric	Cluster 4			Cluster 6			Cluste 7		
	MAE	RMSE	PCC	MAE	RMSE	PCC	MAE	RMSE	PCC
LCOM5	0.091	0.221	0.50	0.072	0.169	0.81	0.045	0.138	0.81
NL	0.093	0.144	0.95	0.062	0.091	0.98	0.051	0.109	0.96
WMC	0.109	0.188	0.60	0.097	0.172	0.66	0.039	0.091	0.89
CBO	0.159	0.262	0.35	0.090	0.134	0.90	0.034	0.090	0.92
TCLOC	0.111	0.188	0.02	0.106	0.171	0.72	0.049	0.086	0.80
DIT	0.120	0.220	0.93	0.101	0.222	0.91	0.058	0.112	0.97
TLOC	0.118	0.194	0.11	0.130	0.189	0.51	0.021	0.057	0.80
TNA	0.146	0.224	0.61	0.073	0.132	0.78	0.047	0.115	0.91
TNM	0.126	0.201	0.21	0.098	0.144	0.57	0.032	0.075	0.88
TNOS	0.114	0.173	0.15	0.058	0.108	0.65	0.024	0.071	0.83
TNPM	0.153	0.238	0.09	0.115	0.162	0.58	0.034	0.085	0.90

V-VII presents regression results of all static metrics in all generated clusters. Overall, error rates of the regression models are very minimal Cluster 9 scores the highest error rates where MAE rates vary from 0.195 to 0.104 and RMSE rates from 0.146 to 0.295. Meanwhile, Cluster 1 had the best error rates. Also, the Pearson coefficient results show a wide range of correlations. The grey cells present the metrics with low correlation with reuse scores. Furthermore, DIT and reuse scores had high correlations in all clusters, as shown in Tables V - VII.

D. Final Reusability Score Aggregation

The final process of the proposed reuse estimation model is to compute the java classes reusability score by aggregating the static metrics' reusability scores, which were obtained in the previous steps. We chose to calculate the final score is using the weighted average. The reason for selecting the weighted average rather than the arithmetic average is that the static metrics have a different influence on the java class reusability. Therefore, we incorporated the PCC as the weight to ensure that the static metrics' influence on reusability was considered. The Final Reusability Score (FRS) of the java class is calculated using the following formula:

$$FRS = \sum_{s=1}^{j} (w(s) * ReuseScore_{metric(s))} / \sum_{s=1}^{j} w(s) \quad (5)$$

where j is the number of the static metrics, w(s) is the obtained cluster's PCC between the reuse score and the static metric.

TABLE VII. SVR Results of Cluster 8-10

Metric	Cluster 8			Cluster 9			Cluster10		
	MAE	RMSE	PCC	MAE	RMSE	PCC	MAE	RMSE	PCC
LCOM5	0.048	0.143	0.85	0.193	0.250	0.72	0.122	0.257	0.53
NL	0.037	0.048	0.99	0.195	0.246	0.82	0.113	0.181	0.92
WMC	0.053	0.101	0.79	0.189	0.266	0.56	0.133	0.249	0.32
CBO	0.066	0.136	0.89	0.122	0.193	0.72	0.158	0.223	0.63
TCLOC	0.083	0.133	0.51	0.141	0.192	0.80	0.132	0.250	0.36
DIT	0.092	0.178	0.92	0.116	0.146	0.97	0.142	0.201	0.97
TLOC	0.044	0.082	0.59	0.175	0.259	0.45	0.109	0.199	0.24
TNA	0.069	0.175	0.77	0.145	0.220	0.74	0.110	0.218	0.54
TNM	0.071	0.115	0.86	0.104	0.198	0.77	0.106	0.198	0.19
TNOS	0.054	0.117	0.67	0.111	0.200	0.74	0.111	0.194	0.25
TNPM	0.079	0.129	0.85	0.125	0.214	0.69	0.145	0.281	0.28

5. Reusability Estimation Model Evaluation

In this section, we evaluated the effectiveness and accuracy of our proposed model. The evaluation phase consists of two phases. In the first phase, we manually evaluate the reuse score using the proposed model on the static metrics of selected java classes. Meanwhile, the second phase aims to assess the accuracy of our methodology by comparing it with a different reusability model. The following subsection depicts the evaluation phases in detail.

A. Effectiveness Evaluation

The effectiveness evaluation aims to confirm that the obtained reusability score is reasonable and interpretable. As mentioned earlier, this evaluation was performed manually on selected java classes. The first step is to identify the java class cluster membership. Since cluster membership helps determine the metrics' reusability scores and PCC, which are required to aggregate the java class reusability score. Table VIII presents the static metrics of the selected java classes with their reusability scores. The cohesive java classes, which have low LCOM5, have scored higher reusability scores than non-cohesive classes. Moreover, complexity could negatively affect the component's reusability. NL and WMC describe the class's complexity. The obtained reusability score has shown the expected. The classes that have high NL and WMC have lower reusability scores lower.

From a coupling perspective, the components that appear to have low CBO values had higher reusability scores. In addition to documented components tend to be more reusable. However, the quality of the documentation depends on the TCLOC and the component's size. Therefore, a high value of TCLOC does not necessarily mean that the component is well documented since the sufficiency of the documentation does not depend on the component's size. Some classes have TCLOC values; however, their reusability score is relatively low due to their large size, as shown in Table VIII.

B. Accuracy Evaluation

To evaluate the accuracy of our proposed estimation model. We compared the performance of our regression models with a different reusability estimation model proposed by Papamichail et al. [20]. The reason we chose this model is that we both used the same dataset. However, we applied the GQM model as a feature selection strategy. Meanwhile, the work in [20] used the entire dataset. In

TABLE VIII. Overview of Java Classes Reusability Scores Computed Using the Proposed Model

Cluster 1											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
1	0	1	1	0	1	2	5	1	1	1	0.70 or 70%
9	1	17	11	35	8	120	2	43	28	34	0.20 or 20%
Cluster 2 & 5											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
4	1	23	2	271	0	373	12	22	14	13	0.44 or 44%
39	3	59	3	263	1	434	1	90	73	85	0.33 or 33%
Cluster 3											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
1	1	30	5	112	0	189	4	16	48	11	0.57 or 57%
2	5	104	12	40	0	785	13	440	393	426	0.24 or 24%
Cluster 4											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
0	0	4	0	21	0	37	2	4	4	4	0.62 or62%
79	1	130	59	978	2	1526	4	170	116	159	0.25 or 25%
Cluster 6											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
1	0	11	2	69	0	139	5	10	32	8	0.61 or 61%
1	8	589	15	190	0	2936	17	117	1591	11	0.26 or 26%
Cluster 7											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
1	1	17	1	15	0	65	4	8	26	8	0.62 or 62%
13	2	99	16	13	1	560	18	141	146	133	0.16 or 16%
Cluster 8											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
0	0	4	1	256	0	334	0	35	1	35	0.63 or 63%
18	1	40	18	200	1	346	5	28	38	26	0.31 or 31%
Cluster 9											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
2	0	4	1	15	1	48	1	29	6	26	0.99 or 99%
21	3	107	31	657	1	1933	53	1742	386	1456	0.27 or 27%
Cluster 10											
LCOM5	NL	WMC	CBO	TCLOC	DIT	TLOC	TNA	TNM	TNOS	TNPM	Reusability Score
1	0	4	0	12	0	24	1	3	2	2	0.57 or 57%
14	3	161	20	1205	1	1796	12	107	221	101	0.12 or 12%

addition to that, we both used univariate analysis to predict the reusability scores. Howsoever, our regression model was trained on each static metric separately within a cluster.

Figure 4 presents a comparison of the MAE values between our model and the model in [20]. Since our regression model was trained on each cluster separately, we selected the best, average, and worst regression model results to compare them with the MAE results of the regression model used in [20]. In general, our regression model has lower error rates than the Papamichail et al. approach [20]. However, their regression model has lower error rates for some static metrics. Our proposed model provided more accurate results on most of the cases, as shown in Figure 4.

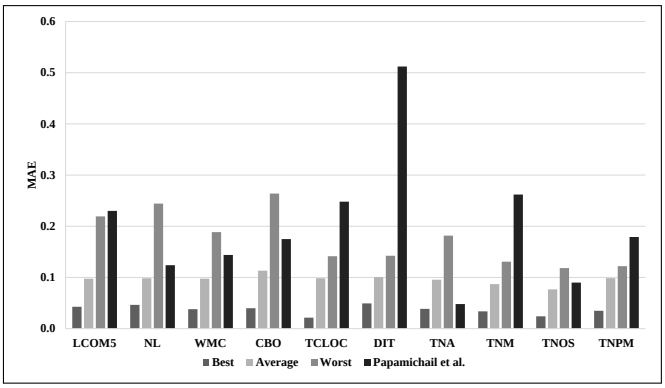

Figure 4. Comparison of Error Rates

6. Threats to Validity

There are some factors that may threaten the validity of our proposed model. First, the dataset was a result of benchmarking the most popular projects in the Maven repository. The reuse rate was calculated based on these projects only. Therefore, projects' popularity is the primary influencer of the reuse rate that could change over time. These changes may affect the effectiveness of our proposed model. Meanwhile, there might be other factors that could influence the reusability of the components, such as the popularity of the project provider. Furthermore, our proposed model depends on the analysis methodology and machine learning model we decided to use while building our model. Adapting different analysis methodologies and machine learning algorithms may produce different results.

7. Conclusion

Software development methodologies have drastically changed, especially with the rapid increase of OSS availability. Also, the adaptation of agile development helped increase the software project productivity by reusing OSS components rather than developing the projects' components from scratch. However, the decision to reuse components is not blindly taken. A software component has thoroughly to be assessed to prior any reuse decision. Measuring the component's reusability by analyzing its static metric is one method that can be used in the assessment phase. In this work, we proposed a reusability score model that statistically analyzed the static metrics of java classes. These metrics were collected by [24] and published as a public dataset. The reuse rate of is dataset is measured by counting the number of the import lines. Furthermore, we explored the effect of feature selection using GQM, data clustering, and regression in aggregating the reusability score. The error rates of the used machine learning algorithms were very minimal. Also, the findings of the effectiveness and accuracy evaluation showed that the proposed model could effectively estimate the java classes' reusability. We intend to explore the effectiveness of our estimation methodology in different components' levels. In future work, we will explore the effectiveness of our methodology in estimating the reusability of the java methods and packages.

References

[1] C. Okoli and K. Carillo, "The best of adaptive and predictive methodologies: open source software development, a balance between agility and discipline," *International Journal of Information Technology and Management*, vol. 11, no. 1/2, p. 153, 2012.

[2] P. J. Kaur and S. Kaushal, "A fuzzy approach for estimating quality of aspect oriented systems," *International Journal of Parallel Programming*, vol. 48, no. 5, pp. 850–869, 2020.

[3] R. Panigrahi, N. Padhy, and S. C. Satapathy, "Software reusability metrics estimation from the social media by using evolutionary algorithms: refactoring prospective," *International Journal of Open Source Software and Processes (IJOSSP)*, vol. 10, no. 2, pp. 21–36, 2019.

[4] Manju and P. K. Bhatia, "Dynamic Reusability Measurement Using Machine Learning Algorithms in Object-Oriented Environment," in *Intelligent Computing and Communication Systems*. Springer Singapore, 2021, pp. 393–399, series Title: Algorithms for Intelligent Systems.

[5] P. Mangayarkarasi, "Automated Software Design Reusability using a Unique Machine Learning Technique," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 5, pp. 1825–1828, 2020.

[6] M. Amin and M. Hammad, "Predicting successful software reuse using machine learning," 2021.

[7] M. Amin and M. Hammad, "Improving software reuse prediction using feature selection algorithms," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. IEEE, 2020, pp. 1–6.

[8] D. P. Wangoo and A. Singh, "A Classification based Predictive Cost Model for Measuring Reusability Level of Open Source Software," *International Journal of Information Technology and Computer Science*, vol. 5, no. 1, p. 5, 2018.

[9] S. Maggo and C. Gupta, "A machine learning based efficient software reusability prediction model for java based object oriented software," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 6, no. 1, pp. 1–12, 2014.

[10] P. K. Singh, O. P. Sangwan, A. P. Singh, and A. Pratap, "A framework for assessing the software reusability using fuzzy logic approach for aspect oriented software," *IJ Information Technology and Computer Science*, vol. 7, no. 2, pp. 12–20, 2015.

[11] M. B. S. Deshpande, D. B. Kumar, and D. A. Kumar, "Assessment of Software Reliability by Object Oriented Metrics using Machine Learning Techniques," *International Journal of Grid and Distributed Computing*, vol. 14, no. 1, p. 10, 2021.

[12] N. Padhy, S. C. Satapathy, and R. Singh, "Estimation of complexity by using an object oriented metrics approach and its proposed algorithm and models," *International Journal of Networking and Virtual Organisations*, vol. 21, no. 1, p. 43, 2019.

[13] R. Kaur and S. Sharma, "An ANN Based Approach for Software Fault Prediction Using Object Oriented Metrics," in *Advanced Informatics for Computing Research*, A. K. Luhach, D. Singh, P.-A. Hsiung, K. B. G. Hawari, P. Lingras, and P. K. Singh, Eds. Singapore: Springer Singapore, 2019, vol. 955, pp. 341–354.

[14] R. Malhotra and M. Khanna, "Prediction of change prone classes using evolution-based and object-oriented metrics," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1755–1766, 2018.

[15] N. Padhy, R. Singh, and S. C. Satapathy, "Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation," *Cluster Computing*, vol. 22, no. 4, pp. 9787–9804, 2019.

[16] A. Shri, P. S. Sandhu, V. Gupta, and S. Anand, "Prediction of reusability of object oriented software systems using clustering approach," *World academy of science, Engineering and Technology*, vol. 43, pp. 853–856, 2010.

[17] S. I. Zahara, M. Ilyas, and T. Zia, "A study of comparative analysis of regression algorithms for reusability evaluation of object oriented

based software components," in *2013 international conference on open source systems and technologies*. IEEE, 2013, pp. 75–80.

[18] P. Negi and U. K. Tiwari, "Machine learning algorithm for assessing reusability in component based software development," *EasyChair Preprint*, vol. 4142, pp. 1–8, 2020.

[19] A. K. M. Fazal-e Amin and A. Oxley, "Reusability assessment of open source components for software product lines," *International Journal on New Computer Architectures and Their Applications (IJNCAA)*, vol. 1, no. 3, pp. 519–533, 2011.

[20] M. D. Papamichail, T. Diamantopoulos, and A. L. Symeonidis, "Measuring the reusability of software components using static analysis metrics and reuse rate information," *Journal of Systems and Software*, vol. 158, p. 110423, 2019.

[21] S. Ergasheva, A. Kruglov, and I. Shulhan, "Development and evaluation of gqm method to improve adaptive systems," *computing*, vol. 4, p. 9, 2019.

[22] S. Ito, S. Hayashi, and M. Saeki, "How Can You Improve Your As-Is Models? Requirements Analysis Methods Meet GQM," in *Requirements Engineering: Foundation for Software Quality*. Springer International Publishing, 2017, vol. 10153, pp. 95–111, series Title: Lecture Notes in Computer Science.

[23] Z. Song, Y. Wang, W. Wang, and J. Zhang, "An Empirical study of Exploring Relevant Metrics to Assess Software Product Quality," in *2020 7th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2020, pp. 114–124.

[24] M. D. Papamichail, T. Diamantopoulos, and A. L. Symeonidis, "Software reusability dataset based on static analysis metrics and reuse rate information," *Data in Brief*, vol. 27, p. 104687, Dec. 2019.

[25] B. Liu, *Web Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[26] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. Springer Science & Business Media, 2011.

[27] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005.

[28] J. Benesty, Jingdong Chen, and Yiteng Huang, "On the Importance of the Pearson Correlation Coefficient in Noise Reduction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 4, pp. 757–765, May 2008.

[29] K. Pearson, "VII. Mathematical Contributions to the Theory of Evolution.—III. Heredity, and Panmixia..pdf," *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, no. 187, pp. 253–318, 1896.

[30] B. MohanGoel and P. Kumar Bhatia, "Analysis of Reusability of Object-Oriented System using CK Metrics," *International Journal of Computer Applications*, vol. 60, no. 10, pp. 32–36, Dec. 2012.

[31] H. Manoj and A. Nandakumar, "Constructing relationship between software metrics and code reusability in object oriented design," *APTIKOM Journal on Computer Science and Information Technologies*, vol. 1, no. 2, pp. 63–76, 2016.

[32] C. Aggarwal and C. Reddy, *DATA CLUSTERING Algorithms and Applications*, ser. Data Mining and Knowledge Discovery Series. CRC Press/Taylor & Francis Group, May 2013.

[33] J. Hawkins, "Support Vector Regression," p. 14.

Mariam Amin is currently studying M.Sc. Software engineering at the University of Bahrain. Also, she works as a Computer Programmer at Bahrain Defense Force, Royal Medical Services. She received her B.Sc. in Computer Science from the University of Bahrain in 2005. Her current research focus is in software analysis and evolution and machine learning.

Mustafa Hammad is an Associate Professor in the Department of Computer Science at the University of Bahrain. He received his Ph.D. in Computer Science from New Mexico State University, the USA, in 2010. He received his Master's Degree in Computer Science from Al-Balqa Applied University, Jordan, in 2005 and his B.Sc. in Computer Science from The Hashemite University, Jordan, in 2002. His research interests include wireless sensor network, machine learning, software engineering with a focus on software analysis and evolution.