Machine Learning Algorithm for Assessing Reusability in Component Based Software Development

Pooja Negi and Umesh Kumar Tiwari

September 7, 2020

MACHINE LEARNING ALGORITHM FOR ASSESING REUSABILITY IN COMPONENT BASED SOFTWARE DEVELOPMENT

[1]Pooja Negi, [2]Umesh Kumar Tiwari

[1,2] Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun

[1]poojanegi669@gmail.com, [2] umeshtiwari22@gmail.com

Abstract- Software reusability has been present for several decades. Software reusability is defined as making new software from existing one. Objects that can be reused: design, code, software framework. We reviewed several approaches in this dissertation, i.e. object-oriented metrics, coupling factor, etc., by which the software's reusability increases. Therefore this thesis analysis on how to classify and reuse the program using those metrics and apply the algorithm of machine learning. In this thesis we test open source software and generate a ck metric of that source code then a machine learning algorithm will process the data using weka tool to give the result. We test coefficient of correlation, mean absolute error, root mean square error, relative absolute error and root relative square error less the program would be better from this we get 98.64 accuracy on online examination system software.

Keywords- Reusability, Machine Learning Algorithm, Random Forest, ck-metric.

1. Introduction

In today's world every sector of service or industry is dependent on computer based application. Industry which develops and outsources the software service is major and growing rapidly in the world. The process of software development is described as the term software engineering. Software engineering is analyzing the need of the user; develop the software in a systematic method which will satisfy the user needs. According to IEEE Standard "The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software". Standard set of goals for developing software in systematic procedure are analysis, designing, and implementation, testing and maintainable. Software must be reliable, useable, modifiable, testable, portable, accurate and maintainable.

Reusability defines as developing a new product from existing one; it works on inheritance [1]. Features for software reuse includes: adaptable, brief, consistence, flexible, simple, tested several times, error chance will less. A substance's reusability is the creation of new material from existing ones. Reusability of software engineering involves the use of existing assets in any form within the phase of software product creation these assets are products and the software development lifecycle products include code, software modules, test suits, design and documentation. The ability to reuse relies on the ability to create bigger items out of smaller parts in an integral way. It also needed application software development characteristics that need not be considered in cases where reusability is not necessary. Reusability is the ability to make use of anything more than once and reuse is the practice of using something more than once. Component is any portion where something has already been made. It in my sense means program[3]. In software system engineering, this allows a software system to have language instruction, sub-routine, process, object, classes, etc. as assembly components. In system architecture components communicate with each other to achieve their goals.
Component Based Software Engineering is a form of Software Engineering that offers the feature of reusability. CBSE stands on the philosophy of "to buy, don't build". Component based software development (CBSD) aim was assembling large software system from existing ones. Component-based architecture focuses on the fragmentation

of the design into logical component or individual function, represent the well-defined communication. Component is the core object of component based software engineering [1]. The objective of component based software engineering is to develop large and complex software system through integration to reduce the cost development rate, effort effectively.

2. Literature Survey

Singh et al.[4] suggested a model-based approach for reusability determination in the current system. Logistic model tree blends Liner regression with Tree induction. In this method, we first produce meta information by evaluating the desired metric of object-oriented software reusability and then calcite the metric value. We use LMT based framework for reusability prediction. Precise classification is the success criterion. Singh et al. [5] proposed metric for reusability estimation of black-box software component along with metrics for interface, understanding, complexity, understanding, reliability and customizability. The identified attribute of measurement model have to be quantified through metrics then these metric are aggregated to estimate black-box component. The component reusability of black-box component estimated as

$$CR= W_1*(1-IC) + W_2* UC+W_3*RCC+W_4*R \ldots\ldots\ldots\ldots..1$$

W_1, W2, W3, W_4 are the weights and other are the metric for identified attribute for reusability. Weight can be influenced by the domain constraints. M.H. et al.[6] proposed a structure for developing a relationship between the standard software metric design variable and the reusability of the code. GQM paradigm is used to estimate the reusability of code from design metric and weighted factor from it. Code reusability model further processed by a system that calculates different CK-metric component forms. Data related to empirical analysis, more analysis to generate the effect of individual metric on the reusability of code, linear coefficient combination with respect to individual design metric. The proposed program performs basic mathematical modeling in the form of code reusability.

$$Cr = (\beta * 100)/\alpha \ldots\ldots\ldots 2$$

Where α is the total amount of class available in each CK-metric, β is a CK-metric variable that matches the amount of newly developed classes through reusability of corporate code. Hudaib et.al.[7] present a paper in which they succeed in classifying groups according to their level of reusability. To define the extent of reuse they use CK-metric package. Self-organizing Map algorithm organizes three separate java-based machine datasets with CK-metric value. In this experiment we came to know how SOM with different size of SOM grid can be applied on software metric. The systems chosen are Hibernate, JDT Eclipse, and PDE Eclipse. Padhy et.al.[8] proposed a metric which will measure the reusable codes in the multi paradigm language. The proposed metric is combination of one of the six metric developed by Chidambaram and Kemmerer. Reusability also increases with increasing the depth of inheritance. Mahajan[9] suggested an effort estimation model based on a dynamic neural network technique to improve the back-propagation algorithm used for testing. The four steps of the model are pre-processing, preparation, estimation, and analysis. In this model project is divided into module and module reusability is analyzed using fuzzy logic to derive reusability matrix. This model deviated from the standard model by around 8 per cent. An average error of less than 1.25 percent over the 39 projects. Selvarani et al.[10] have suggested a similar algorithm to multi-layer perceptron. In order to get the best fit, they implement his proposed algorithm in a prior model of dynamic reusability. DyRM model is used to measure the reusability of the template with the addition of four new input parameters (i.e., cost metric, quantity metric, schedule of work and percentage error). Consistency and complexity factor are determined from the parameter of four inputs. Weight is assigned to the model to provide data. The result is found to fit in best with the training data as well as improved uniformity in the error result. Byun et.al.[11] proposes a dynamic reusability metric to improve the reusability of static and dynamic analysis and to apply visualization of code to determine the level of reusability. For static analysis, XCodeParser is used, and HPROF for dynamic analysis. To evaluate the modularity metric of dynamic reusability which is based on cohesion and coupling is defined. Visualization tool is designed to define the object which needs to be reused. Zozas et.al.[12] present the reusability index is determined by conducting backward linear regression as a function of a series of metrics each being weighted with a particular value. The hierarchical model of reusability is implemented at three stages. Case study is also performed on 80 reusable assets accessible from open source. Provided a result with respect to the REI correlation, accuracy, discriminative and predictive power. Vyas et.al.[13] have introduced a model that can be used to forecast function model usability. We use five machine learning algorithms based on the best performing model to predict usability of function models. Online tools on the software product line are used to pick the product line feature for the software. WEKA was used to develop predictive models. Java code is used to pass validation of the algorithm 's results. Papamichali et.al.[14] suggested a static analysis metric model to evaluate the reusability from five different properties. Projects are evaluated using Code Search Engine AGORA to measure reuse rate. Each metric is associated with some code to measure a reusability, and followed a correlation-based approach.

3. Tools Used

a) **Visual Studio:** Visual studio is Microsoft Software. It is an integrated surrounding for growth. This is used to create blogs, desktop applications , mobile devices, and online services, as well as programs. It is also used to produce data with code metrics that calculate complexity, maintenance etc.

b) **WEKA 3.8.4**: It's tech which is open source. It is used for pre-processing, classification, regression, clustering, association rule and visualization of data, also well suited for developing new algorithm for machine learning.

4. Software Used

We have prepared the data sets using chidamber and kemerer tool for object oriented programming by using online exam software from open source project from github. It is a online platform where universities, school, institutions conducting exams to evaluate student. In this no more computer printouts, pen and paper is required. An online exam is now easier and faster than ever. It is most convenient, secure and user friendly online exam software available.

5. Machine Learning Algorithm

We use machine learning algorithm i.e.; random forest algorithm. It enhanced computing power, now we can select algorithms that perform very intensive calculations. Random forest is a supervised algorithm for the classification. It is the learning method for classification, regression and other tasks that works by creating a multitude of decision tree at the time of training and outputting the class which is the class mode or mean prediction of the individual tree.

We prepared the data sets by downloading open source projects from github using Chidamber and Kemerer tool for object oriented programming.

Selection of Software Metric

We also defined process and object-oriented programming language software metrics which are used to assess the quality of software or system components. We can use machine learning algorithm to predict, evaluate software component quality. The following parameters are used as input attributes for the software component.

The following metrics are used as input attributes of the software components of open source project

a) Maintainability Index: Calculates an index value between 0 and 100, which reflects the relative ease with which to maintain the code. A high value stands for better maintenance. Color coded scores can be used to classify trouble spots in your code very easily. A green rating is between 20 and 100 and implies strong sustainability for the language. A yellow ranking is between 10 and 19, which shows that the code can be maintained moderately. A red rating is between 0 and 9, and indicates low maintenance.

b) Cyclomatic Complexity: Measures the code's structural complicity. It is created by calculating the number of various code paths within the program flow. To achieve better code coverage, a system that has complicated control flow needs more checks and is less maintainable.

c) Depth of Inheritance: Specifies the number of different classes inheriting from each other, all the way back to the base class. Depth of inheritance is similar to class coupling, as each of the inherited classes may be influenced by a shift in a base class. The higher this number, the deeper the inheritance and the higher the potential for modifications of the base class to result in a change that breaks. A low value is good for Depth of Inheritance, and a high value is poor.

d) Class Coupling: Measures the coupling by parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields specified by external types, and decoration attributes. Good software design requires a high degree of compatibility and low coupling between types and methods. High coupling refers to a design which is difficult to reuse and maintain due to its many interdependencies with other types .

e) Lines of Source Code: Software metric used to measure the size of a computer program by counting the number of lines in the source code.

f) Line of Executable Code: Indicates the total number of lines or operations with executable code. This is a count of the number of executable code operations.

Open source software source code is processed to obtain CK-metric result in the visual studio. Using this software metric, we can use data mining techniques to evaluate software components, predict the quality. The following metrics will be used as input attributes of open source project software components.

	Cyclomatic Complexity	Maintainability Index	Depth of Inheritance	Class Coupling	Lines of Source Code	Line of Executable Code
Min	0	71	1	0	1	0
Max	12	100	9	18	204	35
Mean	2.208	88.917	5.286	4.375	29.208	5.958
Std. Dev	2.718	11.271	3.546	4.689	47.11	8.094

TABLE 1: Result of ck-metric

Data Cleaning and Data Transformation

Described software metrics for object-oriented programming language, which is used for software product quality measurement. By using these metrics we can use machine learning algorithm to assess, predict software component quality. The components of software are calculated using software metrics.

Applying Machine Learning Techniques

Using Random forest, by setting the test mode to 3-fold validation we have applied the classification for input attributes and the results are shown in figure 3.2.

In this result we are finding:

a) Correlation Coefficient: A numerical measure of some type of correlation i.e.; a statical relationship between two variables.
b) Mean absolute error: Is a measure of errors between predicted and actual value; it is the average prediction error.
c) Root Mean Square Error: Is a standard deviation of the residual(prediction error)
d) Relative Absolute Error: Approx. error data is the discrepancy between an exact value & some approx. value.
e) Root Mean Square Error: Is a relative error to what it would have been if a simple predictor had been used.

6. Result

In this dissertation we use open source tools with random forest algorithms. We note that recall, precision, TP rate and FP rate are determined by random forest algorithm with fold cross validation, coupling between classes(cluster 0) and coupling afferent class(cluster 1) Precise given by 98.64 random forest

Correlation Coefficient	0.8548
Mean Absolute Error	1.8679
Root Mean Square Error	4.5619
Relative absolute error	33.1288%
Root Relative Squared error	55.6782%

Table 2: Result Table

Acronyms used in performance measurement

Within this dissertation we developed the rule of performance measurement and a matrix of uncertainty that analyzes the efficacy of the classification model. The conditional classifier has two classes, positive or negative.

Performance Measurement Rules

a) Precision: The number of correctly predicted cases as positive for the number of predicted cases as positive , high precision leads to low false-positive rates.

$$TP/(TP+FP)\ldots\ldots\ldots\ldots\ldots..1$$

b) Recall: The number of cases correctly calculated as positive for the number of cases which can be counted as positive.

$$TP/(TP+FN)\ldots\ldots\ldots\ldots.2$$

c) F-score: Precise or recall average is either false positive or false negative.

2*(Recall* Precision)/(Recall+Precision)………..3

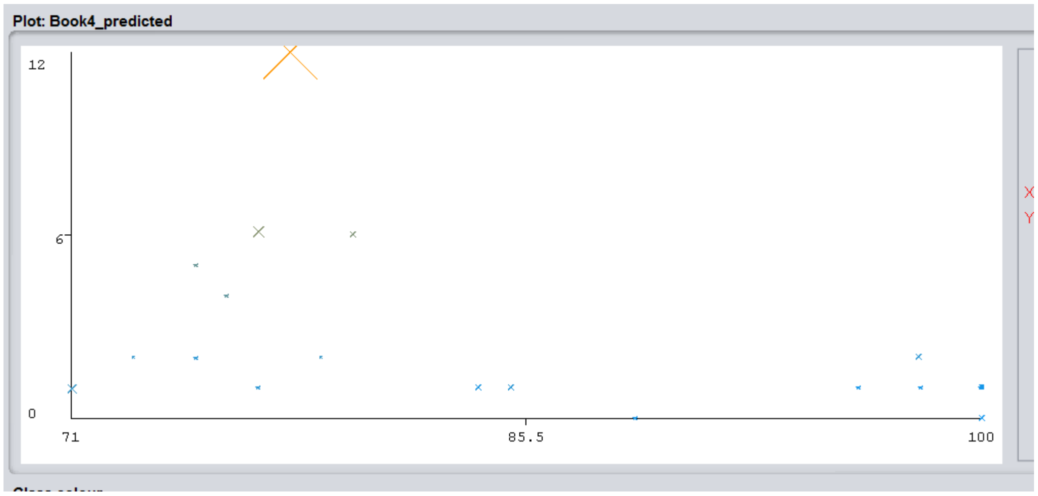
d) Accuracy:- The number of correctly reported cases compared to total number.

(TP+TN)/ (TP+TN+FP+FN)……….4

Visulaization of metrics attributes

1. Maintanability Index and Cyclomatic Index

MI

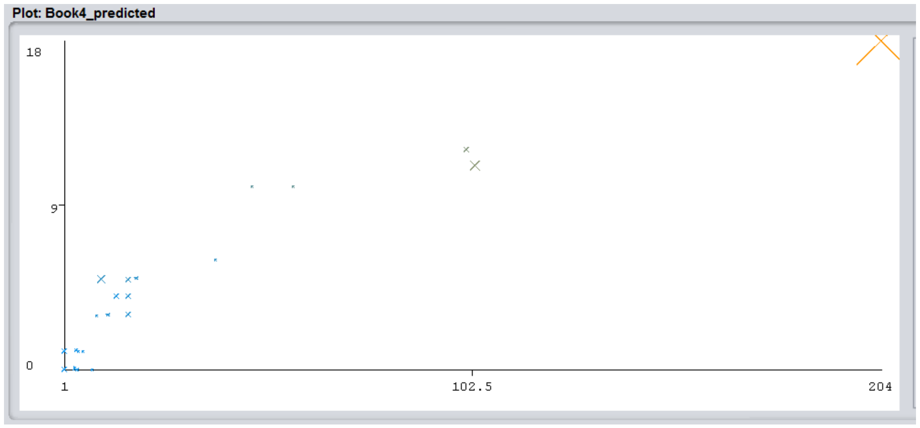


CI

Figure 1 Visualizer between Maintanability Index and Cyclomatic Complexity

1. Class Coupling and Line of Source Code

Line of Source Code



Class Coupling

Figure 2: Visualize between class coupling and line of code

7. CONCLUSION

This dissertation focuses on the reusability of Software. We studied several metrics to identify the reusable component. The metrics which are used to determine the reusability are CK- metric, Object Oriented metrics and coupling factor. The software reusability is of two types reuse by design and reuse for design. There will be high benefit of software reusability that is reliability, quality, efficiency, low time consumption of the software which is going to develop.

In this dissertation we noticed that there is some field in which reusability is not done so our focuses are on to assess reusability by using machine learning algorithm i.e.; random forest algorithm. In this we are first assessing metric by using visual studio, the result is further processed in weka tool to get the accuracy. In this we can achieve 98.64 accuracy.

In future we can propose some neural network algorithm and methodologies to assess the trade-off between reusability and development cost, through which we can determine what will be suitable for the software development.

REFERENCES

[1] Basha. N. Md Jubair, Dr. Moi Salman, " Component Base Software Development; A State of Art", IEEE-International Conference On Advances In Engineering, Science An Management, 2014.

[2] L. Nautiyal, U. Kumar Tiwari, S. Chandra DimriI and S. G. Koolagudi, "Component based software development: New Era with new innovation in software development", International Journal of Computer Applications, vol. 51, no. 19,2012.

[3]G. Beydoun, A. Hoffmann, R. Garcia, J. Shen and A. Gill, "Towards an assessment framework of reuse: a knowledge-level analysis approach", *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 87-95, 2019. Available: 10.1007/s40747-019-0116-1 [Accessed 6 August 2020].

[3] M.-E. Paschali, A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, and I. Stamelos, "Reusability of open source software across domains: A case study," *Journal of Systems and Software*, vol. 134, pp. 211–227, 2017.

[4] Singh. Sarabpreet, Singh. Pushpinder, Mohan. Neeraj and Parvinder S. Sandhu, " Logistic Model Trees based of Object Oriented Software Components", International Journal Of Research in Engineering and Technology Vol. 1, No. 3, pp. 181-185, 2012.

[5] Pratap. Aditiya, Singh,Pradp Tomar,"Estimation of Component Reusability Through Reusability Metrics", International Scholarly and scientific Research & Innovation, pp.1892-1899, 2014.

[6] M. H.m and D. Nandakumar, "Constructing Relationship Between Software Metrics and Code Reusability in Object Oriented Design," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, 2016.

[7] N. Padhy, S. Satapathy, and R. P. Singh, "Utility of an Object Oriented Reusability Metrics and Estimation Complexity," *Indian Journal of Science and Technology*, vol. 10, no. 3, 2017.

[8] "Best Computer Science Journal, International Journal of Computer Science and Mobile Computing - Home." *IJCSMC*, www.ijcsmc.com/.

[9] R. Selvarani and P. Mangayarkarasi, "oDyRM: Optimized Dynamic Reusability Model for Enhanced Software Consistency", *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 3, 2017. Available: 10.14569/ijacsa.2017.080322.

[10] Byun, Eun Young, et al. "Reusability Strategy Based on Dynamic Reusability Object Oriented Metrics." *Journal of Engineering Technology*, vol. 6, no. 1, Jan. 2018, pp. 365–377.

[11] N. Padhy, R. Panigrahi, and K. Neeraja, "Threshold estimation from software metrics by using evolutionary techniques and its proposed algorithms, models," *Evolutionary Intelligence*, Jul. 2019.

[12] Zozas, A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, P. Avgeriou, and I. Stamelos, “REI: An integrated measure for software reusability,” *Journal of Software: Evolution and Process*, vol. 31, no. 8, 2019.
[13] Vyas, Geetika, et al. “Prediction Algorithms and Consecutive Estimation of Software Product Line Feature Model Usability.” *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, doi:10.1109/aicai.2019.8701400.
[14] Papamichail, Michail D., et al. “Measuring the Reusability of Software Components Using Static Analysis Metrics and Reuse Rate Information.” *Journal of Systems and Software*, vol. 158, 2019, p. 110423., doi:10.1016/j.jss.2019.110423.
[15] B. Prakash, D. Ashoka and V. Aradhya, "Application of Data Mining Techniques for Software Reuse Process", *Procedia Technology*, vol. 4, pp. 384-389, 2012. Available: 10.1016/j.protcy.2012.05.059 [Accessed 6 August 2020].
[16] U. Tiwari and S. Kumar, "Cyclomatic complexity metric for component based software", *ACM SIGSOFT Software Engineering Notes*, vol. 39, no. 1, pp. 1-6, 2014. Available: 10.1145/2557833.2557853.
[17] Ghofrani, E. Kozegar, A. bozorgmehr and M. Divband Soorati, "Reusability in artificial neural network: An emperical study", *ACM*, 2019. Available: 10.1145/3307630.3342419 [Accessed 6 August 2020].