Journal of Computer Science 10 (11): 2330-2338, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.2330.2338 Published Online 10 (11) 2014 (http://www.thescipub.com/jcs.toc)

EMPIRICAL COMPARISON OF TWO METRICS SUITES FOR MAINTAINABILITY PREDICTION IN PACKAGES OF OBJECT-ORIENTED SYSTEMS: A CASE STUDY OF OPEN SOURCE SOFTWARE

K.G., Madhwaraj

Department of MCA, SSN College of Engineering, Kalavakkam, India

Received 2014-03-22; Revised 2014-05-14; Accepted 2014-11-29

Funding : Department of MCA, SSN College of Engineering, Kalavakkam – 603 110

Competing Interests: The authors have declared that no competing interests exist

ABSTRACT

Software maintainability has been an important external quality attribute that concerns both styles of software development, the proprietary model as well as open source. As lots of open source software are predominantly built using the OO paradigm, there exists a need for empirical validation with respect to certain quality aspects especially maintainability. There are a few studies in the past which use code metrics and a few which use design metrics, much earlier in the software development life cycle to predict maintainability. In addition, there are studies which apply both code as well as design metrics to evaluate maintainability. The objective of this research is to perform an empirical comparison of two popular OO metrics suites, the Martin suite and the CK suite on four open source software systems by analysing a few key design metrics such as size, coupling, complexity, inheritance and stability. Two important observations were made with this empirical study. First, between the two OO suite of design metrics, the prediction model developed using Martin metrics scores better than the model developed using the CK suite. Second, the combination of Martin and CK suites is helpful in predicting the maintainability of OO software, with a predictive accuracy of 66.7%, better than that of the models constructed by either Martin metrics or by the CK metrics individually.

Keywords: Software Metrics, Maintainability, Object-Oriented Packages, Prediction Models, Software Quality

1. INTRODUCTION

The IEEE Standard Glossary of Software Engineering Terminology defines maintainability as “The ease with which a software system or component can be modified to correct faults, to improve performance or other attributes, or adapt to a changed environment” (IEEE, 1990). Software maintainability is an important external quality attribute that plays a primary role when the quality of software is evaluated. There have been several proprietary software systems which have evolved using the Object-Oriented paradigm. Plenty of research works exist which have studied the quality attributes of such software. Recently, many open source software systems have also started evolving the Object-Oriented way and hence it becomes essential to investigate such software also from the quality perspective.

At the initial stages of software development, the evaluation of quality parameters was carried out during the later stages of the software development life cycle. At this stage, it becomes rather difficult to make changes in the design. Many empirical studies that have been conducted in the recent past indicate that software design metrics help in better prediction of maintainability when compared to measuring it during the later stages of the software development life cycle. Bansiya (2002) built a hierarchical model using the OO design properties and related those properties to high-level quality attributes. Subsequent to this research, several design metric suites such as Martin metrics (Martin, 2003; Chidamber and Kemerer, 1994) and MOOD metrics (Brito and Abreu, 1996) were extracted from the data sets of commercial software projects and subjected to statistical analysis. Some studies were performed to verify which particular suite of metrics would be able to quantify a specific quality attribute in the best possible way. Subsequently, predictive models gained popularity and thereby researchers started building predictive models using these design metrics to evaluate the quality of many software systems. As lots of open source software is built, predictive models using data sets of open source software systems have gained significance and are particularly focusing on certain quality parameters like maintainability, fault-proneness and understandability.

This research study introduces a new perspective in predicting maintainability using design metrics by making an empirical comparison between two popular OO metric suites, the Martin and CK suites. Further, it also provides indications on which particular metric suite would be better in predicting maintainability of OO software. For this purpose, certain package-level design metrics of the Martin metrics suite were extracted using the jdepend tool (Clark 1999) and the CK suite using the ckjm tool (Greece 2005) from all the versions of popular open source software applications namely *jfreechart*, *javageom*, *freemind* and *treeview*.

The empirical analysis that has been performed, compares the relationships between the package design metrics proposed by the Martin and CK suite across the above four open source software systems. The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 defines both the Martin as well as CK suite of metrics. Section 4 describes the open source software taken for a case study. Section 5 highlights the methodology that was used in predicting maintainability. Section 6 gives the results. Section 7 presents the discussion. Section 8 presents the threats to validity. Section 9 gives conclusions obtained from the empirical study.

2. RELATED WORK

Oman and Hagemeister (1994) quantified the maintainability of a system with an MI (Maintainability Index) which was primarily a combination of different code metrics. The concept of using both the code as well as design metrics in predicting maintainability was proposed by Misra (2005). In this study, it was found that both the metrics were useful in evaluating the maintainability of software. Later, (Zhou and Baowen, 2008) empirically investigated the relationships open source software systems. Based on this investigation between 15 design metrics and maintainability of 148 java software, it was found that size and complexity metrics strongly related to maintainability. Gupta and Chhabra (2012). empirically studied 18 packages from two open source software systems and found strong correlations between package coupling and understand ability of a package (s). This study also suggested that coupling metrics could be used to represent other external quality factors. Elish (2010) explored the relationships between five package-level metrics of the martin suite and the effort required to understand a package. This study studied eighteen packages from two open source systems and found statistically significant correlation between most of the martin metrics and understandability of a package. Elish *et al*. (2011) empirically evaluated three suites of package-level metrics (Martin, MOOD and CK) in predicting the number of pre-release and post-release faults in packages of eclipse software. It was found that models which are based on Martin suite had more predictive power when compared to the MOOD and CK suites across various releases of eclipse. The current study that has been conducted in this study empirically compares the relationships between the Martin suite and the CK suite on maintainability.

3. PACKAGE-LEVEL DESIGN METRICS

3.1. Martin Suite of Metrics

The metrics proposed by Martin (2003) which are used in this empirical study are defined in this section. These design metrics were extracted for all the 52 versions of *jfreechart* using the jdepend tool (**Fig. 1**) since its release. Further, the metrics were extracted at the package level as packages have now become important organizational units for large applications (Niemeyer and Knudsen, 2005).

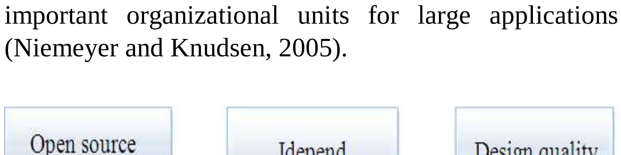

Fig. 1. The extraction process

3.1.1. Concrete Classes (CC)

This captures the number of concrete classes in a package. This metric indicates the size of software.

3.1.2. Abstract Classes (AC)

This captures the number of abstract classes (and interfaces) in the package. This too is a metric that indicates size. Both the above metrics (CC & AC) are indicators of the extensibility of a package.

3.1.3. Afferent Couplings (Ca)

This metric provides a count of the number of other packages that depend upon the classes within a given package. This is an indicator of the responsibility of that package.

3.1.4. Efferent Coupling (Ce)

This metric gives a count of the number of classes in the current package that depends on other packages and classes. This is an indicator of the independence of that package.

3.1.5. Abstractness (A)

This metric for a package is defined as the ratio of the number of abstract classes to the total number of classes in the analysed package. The range of this values for this metric is between 0 and 1, with A = 0 indicating a completely concrete package and A = 1 indicating a completely abstract package.

3.1.6. Instability (I)

This metric is defined as the ratio of efferent coupling (Ce) to total coupling (Ca + Ce). The range of values for this metric is between 0 and 1. When I = 0, it is a completely stable package and when I = 1, it is a completely unstable package. This is an indicator of the package's resilience to change.

3.1.7. Distance from the Main Sequence (D)

This is the perpendicular distance of a package from the idealized line A+I = 1. A package that is squarely on the main sequence is optimally balanced with respect to its abstractness and stability. Ideal packages are either completely abstract and stable or completely concrete and unstable. The range of values for this metric is between 0 to 1, with D = 0 indicating a package coincident with the main sequence and D = 1 indicating a package as far as possible from the main sequence.

3.2. Chidamber and Kemerer (CK) Suite of Metrics

The CK suite consists of six class-level metrics that are defined in this section as follows.

3.2.1. Weighted Methods Per Class (WMC)

WMC is defined as the sum of the complexities of all the methods defined in a particular class.

3.2.2. Coupling between Object Classes (CBO)

This metric gives the number of classes coupled to a given class.

3.2.3. Response for a Class (RFC)

This metric measures the number of different methods that can be executed when an object of that class receives a message.

3.2.4. Depth of Inheritance Tree (DIT)

This metric provides for each class a measure of the inheritance levels from the object hierarchy.

3.2.5. Number of Children (NOC)

This metric measures the number of immediate descendants of the class.

3.2.6. Lack of Cohesion in Methods (LCOM)

This metric counts the sets of methods in a class that are not related in sharing some of the class's data. Since the CK suite captures the metric values at the class level, they have been converted to the package level by taking the average of all classes in a package.

4. CASE STUDY

The software systems that have been taken for a case study are *jfreechart*, *javageom*, *freemind* and *treeview*, all of which are open source. The *jfreechart* software is a very popular charting application that enjoys the maximum downloads (4000 downloads per week). All the 52 versions of *jfreechart*, 15 versions of *freemind* software, 21 versions of *javageom* software and 18 versions of *treeview* software are taken for analysis. All these software systems are popular systems among the user community. The significance of this selection is that all these software were developed using the java language. A dataset of 106 versions of open source software was taken for statistical analysis.

5. METHODOLOGY USED IN PREDICTING MAINTAINABILITY

In our case study, the maintainability of a system is quantified with a Maintainability Index (Oman and Hagemeister, 1994). MI is a combination of different metrics that affect maintainability. It can be defined as follows:

$$MI = 171-5.2 \ln (aveV)-0.23aveV (g')-16.2 \ln (aveLOC) + 50 \sin (\mathrm{sqrt} (2.4perCM))$$

where, aveV is the average Halstead's Volume per module, aveV(g') is the average extended cyclomatic complexity per module, aveLOC is the average count of lines of source code per module and perCM is the average percentage of lines of comments per module. This is a code metric which takes into account several aspects of maintainability like size, complexity and self-descriptiveness of the source code. The range of MI values are given in **Table 1**. The maintainability index for all the versions of the four different open source software was measured and this was taken as the dependent variable for studying the relationships between design metrics and maintainability.

The different package design metrics (AC, CC, Ca, Ce, I, A, D, WMC, RFC, DIT, NOC, CBO and LCOM) were taken as the independent variables. These metrics have already been defined in Section 3. Metric data was collected from the several versions of the four open source software. Notably, as all these metrics were captured at the package level, the mean value of all packages in a particular version was taken as the independent variables for the study.

We know that every software system consists of both the system packages as well as user-defined packages. In this study, only the user-defined packages across all the versions have been considered. This would provide clear indications on how user-defined packages have been designed. Further, it will also provide indications on which metrics need to be taken care while designing the next version of software. The study was conducted in three phases as below:

Table 1. Range of maintainability index

MI value	Maintainability
<65	Poor
65-85	Moderate
>85	Good

- Using the design metrics proposed by the Martin suite as independent variables and MI as a dependent variable
- Using the design metrics proposed by the CK suite as independent variables and MI as a dependent variable
- Using the combination of both Martin and CK metrics as independent variables and MI as a dependent variable

Several statistical tests like multivariate correlation, multivarite regression and factor analysis were performed using the dataset in all the three phases. Further, we tested the OO dataset for multi-collinearity by performing a test for multi-collinearity and a VIF (Variance Inflation Factor) test. The following sub-sections define the different statistical tests that were applied in all the three phases of our case study.

5.1. Multivariate Correlation

The degree of relationship between two or more variables is statistically called as correlation. It can also refer to the co-variation (variation in one variable affecting the variation in the other variable). The degree of correlation between two variables is called as simple correlation or univariate correlation and the degree of correlation between one variable and several other variables can be called as multiple correlation or multivariate correlation. Both uni-variate and multivariate correlation were performed to understand the influence of all the design metrics on maintainability. The following tests were performed to test the levels of correlation.

5.1.1. Test for Multi-Collinearity

Multi-collinearity is a statistical test that is used the test the level of dependence or correlation among design metrics. During correlation, if we find that every variable in correlation is depending on every other variable, chances of multi-collinearity is possible. This can be detected when almost all the inter-correlations between variables have a value greater than 0.9. Statistical evidence has shown that the existence of multi-collinearity within a dataset would never help in providing the right prediction about the correlations between design metrics and if not detected, would result in making biased conclusions.

5.1.2. Variance Inflation Factor (VIF) Test

Multi-collinearity can also be detected by testing the variance inflation factor of all the design metrics. We tested this also and kept the VIF to a minimum by applying another multivariate statistical technique called as Factor Analysis.

5.2. Multivariate Regression

Regression is the determination of statistical relationship between two or more variables. One variable (independent) is the cause of the behavior of another one (dependent). When there are more than two independent variables, the analysis concerning the relationship is known as multiple correlations and the equation describing such relationship is called as the multiple regression equation. Regression analysis is concerned with the derivation of an appropriate mathematical expression which is derived for finding values of a dependent variable on the basis of independent variable(s). It is thus designed to examine the relationship of a variable Y to a set of other variables X_1, X_2, X_3…………X_n. Therefore, multivariate regression analysis was performed to examine the common effectiveness of the metrics. The general form of a multivariate linear regression model can be given by:

$$\overline{y_i} = a_0 + a_1 x_{i_1} + \ldots + a_k x_{i_k}$$
$$y_i = a_0 + a_1 x_{i_1} + \ldots + a_k x_{i_k} + e_i$$

where, $x_{i_1},\ldots,x_{i_k}$ are the independent variables, $a_0,\ldots,a_k$ are the parameters to be estimated, $\hat{y}_i$ is the dependent variable to be predicted, y_i is the actual value of the dependent variable and e_i is the error in the prediction of the i^{th} case. We used stepwise regression to build the model.

5.3. Factor Analysis

This is a multivariate statistical technique that is used if multi-collinearity exists within the data set. If multi-collinearity is left undetected within a data set, biased conclusions can be made while making a few predictions. We performed regression after obtaining the factor scores as a result of factor analysis. Factor scores are a set of values that are generated from the original data set. Regression is later performed with factor scores as the independent variables and MI as the dependent variable. There are two important parameters of factor analysis. The KMO measure of sampling adequacy is used to compare the magnitudes of the observed correlation coefficients in relation to the magnitudes of the partial correlation coefficients. KMO values range between 0 and 1 and it is good to have values closer to one.

Bartlett's test of sphericity is a statistical test that is used to test whether the correlation matrix is an identity matrix i.e., all metric variables are perfectly correlated with themselves (a value of one) and have some level of correlation with the other metric variables. If they are not correlated with the other items, then they can't be a part of the same factor. Researchers always look for significance value less than 0.05.

The communalities are yet another result of factor analysis. The communalities explain the proportion of variance accounted for by the common factors (or 'communality') of a variable. The communality value has a range between 0 to 1. A value of 0 means that the common factors don't explain any variance; 1 means that the common factors explain ALL the variance. Researchers always look for a higher value closer to one.

Therefore, we performed all the tests in each phase that were necessary to make strong conclusions on predicting maintainability.

6. RESULTS

6.1. Predicting Maintainability using Martin Suite

6.1.1. Multivariate Correlation

The inter-correlation values between the design metrics of the Martin suite were not greater than 0.90. Except for the concrete classes, all the other metrics had significant influence on maintainability.

6.1.2. Multivariate Regression

The regression model fetched a multiple correlation coefficient of 0.787. The value of R^2 was 0.620 and also significant at the 99% level. The f values were also high.

Fitted Regression Line:

MI = 55.855-13.678 (D)-2.577 (Ca)-0.154 (CC) + 0.978 (Ce) - (1)

6.1.3. Factor Analysis

Since the efferent coupling of the Martin suite had a VIF of 6.946 (not a desired range), we performed factor analysis on the data set and then later performed regression using the factor scores obtained from factor analysis. The Martin metrics gave a KMO value of 0.592. The Bartlett's test of sphericity value was less than 0.01.

6.2. Predicting Maintainability using CK Suite

6.2.1. Multivariate Correlation

As with the Martin suite, all the inter-correlation values had no multi-collinearity i.e., all the correlation values are not >.9. The variance inflation factor was also checked to detect the presence of multi-collinearity within the dataset. It clearly showed that out of six metrics, two metrics namely WMC and CBO have significant correlation with MI at 99% level. The LCOM metric is also significant at 95% level. WMC was positively contributing towards maintainability. CBO showed a negative correlation.

6.2.2. Multivariate Regression

The NOC metric became a removed variable from the regression analysis as it did not contribute significantly on MI. The coefficient of determination R-square was found to be 0.471. The R square value was also significant at 99% level.

Fitted regression line:

$$MI = 42.007 - 1.181(CBO) + 0.437(RFC) \qquad - (2)$$

6.2.3. Factor Analysis

As done with the Martin metrics suite, factor analysis was performed on the CK metrics data set to remove any levels of multi-collinearity. The results of factor analysis and factor scores regression were as follows:

- The KMO value is just 0.465 which is less than what was obtained with the Martin metrics suite
- The communalities value of the DIT metric was at 0.480 whereas in the Martin metrics suite, all the variables had a very high communality value
- There are two factors that have been formed by factor analysis that explains 82% of the total variance which is less than the Martin metrics suite which obtained 92%
- The regression performed after factor scores obtained through factor analysis yields an R^2 of 0.242 which is very less when compared to the Martin suite which gave an R^2 of 0.463

6.3. Predicting Maintainability using Martin and CK Suite

6.3.1. Multivariate Correlation

The following were the inferences from the analysis. Six metrics out of seven of the Martin suite and three metrics of the CK suite are showing significant correlation with MI. The DIT metric(Martin suite), NOC metric and RFC metric(CK suite) did not show any impact on MI.

6.3.2. Multivariate Regression

Stepwise regression was performed with the combination of the Martin suite and CK suite. We found that the distance metric of the Martin suite as the primary contributor influencing MI. The CBO, WMC and RFC metrics of the CK suite are secondary indicators. The abstractness metric of the Martin suite is significantly influencing MI.

6.3.3. Factor Analysis

The regression with factor scores gave an R^2 of 0.511 i.e., the variables explain 51.1% of the variance in MI.

7. DISCUSSION

7.1. Predicting Maintainability using Martin Suite

7.1.1. Multivariate Correlation

Since the inter-correlation values between the design metrics of the Martin suite were not greater than 0.9, this indicates that there is no big multi-collinearity in the dataset.

7.1.2. Multivariate Regression

As the predicted values were obtained as a linear combination of the distance metric, afferent couplings, concrete classes and efferent couplings, the co-efficient value of 0.787 indicates that the relationship between maintainability and the four independent variables of the Martin suite is quite strong and positive. The coefficient of determination R-square measures the goodness of fit of the estimated Sample Regression Plane (SRP) in terms of the proportion of the variation in the dependent variable explained by the fitted sample regression equation. Thus, the value of R square is 0.620 simply means that about 62% of the variation in maintainability is explained by the estimated SRP that uses distance, afferent coupling, concrete classes and efferent coupling as independent variables.

7.1.3. Factor Analysis

The KMO value of 0.592 is good. The Bartlett’s test is less than 0.01 i.e., i.e., .000 which is very good and is a test which indicates that factor analysis can be continued further. It was also noticed that all the Martin design metrics showed a high communality value which provides us a fact that most of the variance in the dataset have been explained by the factors. This is very positive and good.

7.2. Predicting Maintainability using CK Suite

7.2.1. Multivariate Correlation

The WMC had a positive influence on the maintainability i.e., when the weighted methods for a class increases, maintainability also increases, which is a surprising result. Literature shows that high WMC results in high complexity which in turn reduces maintainability and a low WMC always helps in reusability, testing and more importantly bettering maintainability levels. CBO showed a negative correlation i.e., when CBO decreases the maintainability increases and vice versa. RFC also showed a significant negative influence on maintainability. LCOM shows a positive correlation i.e., when the levels of method cohesion in a class increases, maintainability increases and vice versa. There is past literature which justifies the fact that when higher levels of LCOM exists within a class, it results in a fault or error.

7.2.2. Multivariate Regression

The R^2 value explains about 47.1% of the variation in maintainability that uses CBO and RFC as independent variables. The R square value was also significant at 99% level. The other metrics were removed by the regression model. Though both the metrics CBO and RFC are significant at the 99% level, the F values are not very high.

7.2.3. Factor Analysis

The F-values that were obtained by the CK suite were much lower than the F-values of the Martin suite i.e., The R^2 also seems to be lower in the case of the CK suite when compared with the Martin suite.

7.3. Predicting Maintainability using Martin and CK suite

7.3.1. Multivariate Correlation

There is no multi-collinearity in the OO dataset taken for analysis. Therefore, the conclusions made are valid conclusions.

7.3.2. Multivariate Regression

The following conclusions can be made after performing multivariate regression analysis:

- The distance metric is the balance between abstractness and instability which is giving a negative influence on MI. Previous literature has shown that as and when packages have a high distance value, maintainability becomes difficult. When packages stay within the main sequence, it is good for maintainability purposes (Martin, 2003). Abstractness talks about the number of abstract classes when compared to the concrete classes in a package Instability is the ratio of efferent coupling to total coupling (efferent coupling + afferent coupling). This negative influence indicates that coupling has a negative influence on MI
- The CBO metric of the CK suite is the next important predictor which again indicates that any sort of coupling is detrimental in bringing down the values of MI. To add, it again gives a negative influence on MI
- The WMC metric delivers a negative influence i.e., when the weighted complexity of methods in a class increases, the MI would decrease. It is advised to reduce the complexity of the methods in a class
- The RFC metric and the Abstractness metric are giving positive influences. This gives us another indication that when the count of abstract classes are higher when compared to the concrete classes, this stands as a good sign in increasing the MI. It is advised to use more abstract classes in package design

The model generated is able to give a predictive accuracy of 0.667% i.e., the model is able to explain 66.7% of the variance in MI. The F values are also significant and higher when compared to the F values of Martin and CK suite.

7.3.3. Factor Analysis

Stepwise multiple regression was done with the four factor scores generated after applying factor analysis. The factor scores were taken as the independent variable and the MI was taken as the dependent variable. The comparative study is presented in **Table 2**.

Table 2. Summary of comparative study

Statistic	Martin suite	CK suite	Martin and CK suite
Correlation with MI	Six out of seven metrics are highly influencing MI. All the six metrics were significant at 99% level	Three out of Six metrics are highly influencing MI. Two metrics are significant at 99%level and one metric significant at 95% level.	Six metrics of Martin suite and three metrics of CK suite highly significant with MI.
Goodness of fit (R^2) after regression in %	62	47.1	66.7
F values (ANOVA) after regression	High values	Less when compared to Martin suite	High values
Variance Inflation Factor after regression	Efferent coupling had a VIF of 6.946; Other	Two metrics had a low VIF metrics give a less VIF	Three metrics of CK suite had higher VIF when compared to two metrics of Martin suite
KMO measure of sampling adequacy	0.592	0.465	0.545
Bartlett's test of sphericity (sig value)	0.000	0.000	0.000
Communality values	All metrics have communality values closer to one	DIT metric has a low communality value of 0.480; Other metrics have values closer to one	DIT metric had a communality value of 0.709; All other metrics had communality values
Cumulative % of variance explained after factor analysis	92.715	82.576	88.360
Goodness of Fit (R^2) after factor scores regression in %	45.6	24.2	51.1
F values (ANOVA) after factor scores regression	Higher than CK suite and Martin and CK suite	Less when compared to Martin and CK suite	Not very high

8. THREATS TO VALIDITY

The design metrics of the Martin suite which were used as independent variables in this study were extracted from the source code of different versions of open source software. Therefore, the design information that was extracted is the current design information and not the original design information i.e., significant re factorings or design changes could have been done to the different versions. The various statistical analysis which were done with the different design metrics cannot be taken as final indicators for predicting maintainability. We only took four software applications, the *jfreechart*, *freemind*, *treeview* and *javageom* and their several releases right from their evolution. In order to get meaningful conclusions, more such empirical validations need to be performed in future on different open source data sets. This would further validate the claim that a few metrics in the Martin and CK suites are helpful in predicting maintainability.

9. CONCLUSION

We have made an attempt to investigate two popular OO metric suites on the maintainability of four open source software systems. Zhou and Baowen (2008) found that size and complexity metrics as primary indicators, coupling and cohesion metrics as secondary indicators for predicting maintainability. We found on the contrary, the Abstractness metric of the Martin suite as a primary indicator for predicting maintainability. The secondary indicators are the coupling metrics (both Ca and Ce) which have a negative influence on maintainability and the tertiary indicators being the complexity metrics. Before the next version of open source software is released, the designers are advised to increase the number of abstract classes when compared to concrete classes while deigning user-defined packages.

We also found Martin metrics as a better suite of metrics than the CK suite while predicting the maintainability of four open source software. This conclusion is very much evident where Martin metrics are scoring better than CK metrics (**Table 2**). When both the Martin and CK suite were used to build a model, there are a few important parameters where this model (Martin and CK) seems to predict maintainability better than the Martin and CK suite independently. i.e., The goodness of fit (R^2) after regression is 66.7%, which is better than the Martin suite and the CK suite and the other is the R^2 value with factor scores regression which is 51.1%. Therefore, it is advised to use the Martin and CK suite model in predicting maintainability of open source software. More importantly, the Distance and Abstractness metric of the Martin suite and CBO, WMC and RFC metrics of the CK suite are significantly influencing maintainability either positively or negatively.

As future work, we would like to investigate other popular object oriented-suites and extract evidence on their impact too on predicting the maintainability. Our immediate focus would be on getting the right blend of metrics that would help in predicting the maintainability of object oriented open source software in the best possible way.

10. ACKNOWLEDGMENT

I would like to thank Dr. Chitra Babu, Professor and Head, Department of CSE for giving effective research directions. I would also acknowledge the PG students of my Department who supported me in pursuing this research work".

11. ADDITIONAL INFORMATION

11.1. Funding Information

This project was funded by the Department of Computer Applications, SSN College of Engineering, Kalavakkam.

11.2. Author Contributions

a) A predictive model for OO software maintainability using Martin metric suite

b) Identification of the most influential metrics from both Martin and CK suites useful for predicting OO software maintainability

c) A predictive model for OO software maintainability using a subset of Martin and CK metric suites

11.3. Ethics

I wish to state that this work is done by me wholly and there are no ethical issues that would arise after this article gets published.

12. REFERENCES

Bansiya, J., 2002. A hierarchical model for object-oriented design quality assessment. IEEE Trans. Software Eng., 28: 4-17. DOI: 10.1109/32.979986

Brito, E. and F. Abreu, 1996. Evaluating the impact of object-oriented design on software quality. Proceedings of the 3rd International Software Metrics Symposium, Mar. 25-26, IEEE Xplore Press, Berlin, pp: 90-99. DOI: 10.1109/METRIC.1996.492446

Chidamber, S.R. and C.F. Kemerer, 1994. A metrics suite for object oriented design. IEEE Trans. Software Eng., 20: 476-493. DOI: 10.1109/32.295895

Clark, M., 1999. Pragmatic Project Automation: How to Build, Deploy and Monitor Java Applications. Pragmatic Bookshelf, LLC.

Elish, M.O., 2010. Exploring the relationships between design metrics and package understandability: A case study. Proceedings of the 18th IEEE International Conference on Program Comprehension, Jun-Jul. 30-2, IEEE Xplore Press, Braga, Minho, pp: 144-147. DOI: 10.1109/ICPC.2010.43.

Elish, M.O., A.H.A. Yafei and M.A. Mulhem, 2011. Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems: A case study of eclipse. Adv. Eng. Software, 42: 852-859. DOI: 10.1016/j.advengsoft.2011.06.001.

Greece, A., 2005. Ckjm-a tool for calculating Chidamber and Kemerer Java metrics. Athens University of Economics and Business.

Gupta, V. and J.K. Chhabra, 2012. Package level cohesion measurement in object-oriented software. J. Comput. Sci. Technol., 24: 273-283. DOI: 10.1007/s13173-011-0052-4.

IEEE, 1990. 610.12-1990-IEEE standard glossary of software engineering terminology. Report IEEE Std. DOI: 10.1109/IEEESTD.1990.101064.

Martin, R., 2003. Agile Software Development: Principles, Patterns and Practices. Prentice Hall, NJ. ISBN-10: 0135974445.

Misra, S.C., 2005. Modeling design/coding factors that drive maintainability of software systems, Software Quality J., 13: 297-320. DOI: 10.1007/s11219-005-1754-7.

Niemeyer, P. and J. Knudsen, 2005. Learning Java, 2nd Edn., O'Reilly and Associates, ISBN-10: 0596002858, pp: 826.

Oman, P. and J. Hagemeister, 1994. Construction and testing of polynomials predicting software maintainability. J. Syst. Software, 2: 251-266. DOI: 10.1016/0164-1212(94)90067-1.

Zhou, Y. and X.U. Baowen, 2008. Predicting the maintainability of open source software using design metrics. Wuhan University J. Nat. Sci., 13: 14-20. DOI: 10.1007/s11859-008-0104-6.