Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EMPIRICAL VALIDATION OF COUPLING METRICS FOR OBJECT-ORIENTED SYSTEM

MUKESH BANSAL[1], CHAITANYA PURUSHOTTAM AGRAWAL[2]

[1]Research Scholar, Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal, INDIA
[2]Professor, Dept of Computer Science & Applications, Makhanlal Chaturvedi National University of Journalism and Communication, Bhopal, INDIA
* Corresponding author's Email: mukeshbansal76@gmail.com

ABSTRACT

The Object oriented design metrics can be used to make quality management decisions. The objective of this study is the examination of the connection among object-oriented design metrics. We made a survey and analyzed various object oriented metrics available in literature. We have proposed three new object oriented metrics viz. Attribute Interface Coupling, Method Interface Coupling (MIC) and Design Complexity to measure coupling. The new metrics shall help in measuring complexity of design at early stage based on coupling, designing object-oriented code as well as improve its quality by removing the anomalies and redundancy from code. These metrics have been validated by using six java based projects of different application areas. The empirical validation proves the significance of the proposed metrics.

Keywords: *Coupling Metrics, Object-Oriented System, MOOD, CK, Complexity, Interface Coupling*

1. INTRODUCTION

The software metrics has significant role to determine the software quality and the same is accepted by the community of software engineers [1][2], while the software quality engineers underlined the usage of metrics to determine the software quality [3]. Due to the growth of the object oriented technology in today's era of software development makes object oriented metrics highly useful. Object oriented metrics are used to determine the software quality in terms of complexity, reusability, maintainability, testability and understand ability [4]. The software metrics are generally applied at the early stage of software to generate quality software [5]. The priority software quality parameter is decided on the basis of application area of the software [6]. This priority parameter maps to particular software metrics for efficient results. Different software metrics are designed to analyze the software quality are Chidamber and Kemerer (CK), Lorenz and Kidd and MOOD. These metrics use different parameters to determine the software quality. CK metrics suite involves following metrics [7].

1.1 Weighted Methods per Class (WMC):

This is a type of CK metrics which is used to measure the complexity of any particular class. In this metric the weight of each method in a class is evaluated on the basis of complexity of the method. If all the methods in the class are equally complex then the number of methods in each class gives the WMC value. The effectiveness of any software is inversely proportional to the WMC i.e. lower the value of WMC results in higher effectiveness. This concept doesn't involve friends operator as these operators are used to evaluate the usability, quality and complexity of software being monitored [8]. Mathematically it can be given by equation (1)

$$WMC = \sum_{c=1}^{n} M_c \quad (1)$$

Here, n is the number of methods with $M_1, M_2, \ldots M_n$ as the complexity of the method.

1.2 Number of Children (NOC)

It represents the number of classes inherited from any particular class. Higher number of children

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

enhances the reusability of code as well as the testing efforts.

1.3 Depth of Inheritance Tree (DIT)

It demonstrates the length of inheritance tree in terms of number of classes from root to the leaf node [8][9].

1.4 Coupling between Objects (CBO)

It gives the relation (coupling) of any particular class with other number of classes. The increase in CBO results in decrease in the reusability of the code. Moreover, this metric is used to compute the complexity, reusability and the quality[9].

1.5 Lack of Cohesion on Methods (LCOM)

It denotes the number of methods present in the class without any common instance variable minus number of methods available with common instance variable**.** The increase in LCOM value denotes the lower cohesion [10].

1.6 Response for a class (RFC)

It denotes the number of methods to be executed in response to the message received by the object. It is directly proportional to the complexity of the class i.e. Higher the number of methods to be executed, greater is the complexity of the software [9][10].

These metrics completes the CK metric suite, similarly other metric suite like MOOD and Lorenz and Kidd involves different metrics. Different authors have worked on these metrics to analyze the software quality. The author of [11] presented a set of eleven well established object oriented metrics that can be used to rank programs on their complexity values, to assess testability and maintainability of the programs. While the author of [12] gave approach to assess the design quality of internal and external structure of a system at the class level which is the most fundamental level of a system. In the case study conducted by author of [13] design measures to evaluate the software quality are measured. The author computes the quality of six different java based projects by using the CK metric suite. The author of [14] reviewed MOOD and QMOOD set of metrics. The author demonstrated that these metrics are very useful to analyze the software quality. The authors of [15] defined cohesion and coupling metrics that works on dependency graphs between software modules and dependencies. In [16] a prediction model consisting of ten OO metrics using statistical analysis technique in order to derive relationship between maintenance and metrics has been proposed. NN based estimation of software quality has been done in [17]. They compared parametric model and ANN model to estimate accuracy. The author of [18] maintains relationship between static metrics and software fault proneness by computing static metrics (Cyclomatic complexity) and dynamic metrics (dataflow coverage). In [19], a model is devised to predict faulty classes in java application. The author of [20] studied on Comparing Complexity in Accordance with Object Oriented Metrics. The study highlighted the object-oriented software metrics proposed in 90s' by Chidamber, Kemerer and several studies were conducted to validate the metrics and discovered several deficiencies. A study on Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction has been done in [21]. This work uses the code of the Mozilla web and email suite. The study also used these modified metrics and added one more object-oriented metric i.e. Lack of cohesion on methods (LCOM) and the well-known lines of code metric (LOC). The study used logistic regression and machine learning methods to predict the fault proneness of the code. This study clearly shows that the existing metric suite can be used to determine the software quality. While the literature doesn't cover any metric that determines the coupling of the attributes as well as the methods to determine the complexity and maintainability of the software. This paper defines a new set of metric to determine the coupling [22][23] between the attributes and the methods which calculates the complexity as well as maintainability of software. The rest paper is organized four more sections. The next Section i.e. section 2 gives the new metrics which are proposed. In section 3 we take a case study to calculate the values of proposed metrics. Then in the section 4 these metric are used to evaluate the software complexity on different projects. Then conclusions and future research directions are given in Section 5.

2. PROPOSED OBJECT ORIENTED METRICS

This section proposes object oriented metric for the analysis of an object oriented software. This suite has included 3 set of metric described below.

2.1 Attribute Interface Coupling (AIC)

AIC may be used as a measure of coupling between two classes. High value of AIC indicates tight coupling and vice versa. This metric can be

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

defined as the sum of ratio of data as well as control attribute parameters of all the classes

$$AIC=\sum_{i=0}^{n}\frac{Id+Od}{Ic+Oc} \quad (2)$$

Id = total number of input data parameters
Ic = total number of input control parameters
Od= total number of output data parameters
Oc= total number of output control parameters
n= total number of classes

2.2 Method Interface Coupling (MIC)

MIC may be used as a measure of coupling between methods within a class or different classes. High value of MIC indicates higher coupling and vice versa. This metric can be defined as the sum of ratio of data as well as control parameters of all the methods of the class.

$$MIC=\sum_{i=0}^{m}\frac{Id+Od}{Ic+Oc} \quad (3)$$

Id = total number of input data parameters to a method
Ic = total number of input control parameters to a method
Od= total number of output data parameters to a method
Oc= total number of output control parameters to a method
m= total number of methods in a class

2.3 Design Complexity

Design Complexity helps in measuring coupling of overall design. Higher value of DC indicates high coupling and Lower value of DC indicates low coupling.
It can be defined as the sum of Attribute Interface Coupling (AIC) and Method Interface Coupling (MIC) of all the classes.

$$DC= AIC+\sum_{1}^{c} MIC \quad (4)$$

C= Total no of classes
The design complexity metric covers the AIC as well as MIC metric. The behavior of the design complexity is the result of the AIC and MIC that's why DC can be used to analyze the software quality. These metric can be understood by the case study done in the next section.

3. CASE STUDY

This section explains the proposed metric given in previous section by using an example.

3.1 Attribute Interface Coupling (AIC)

We assume that there are two classes, sample and experiment. Class name sample having three input data parameters as id1, id2, id3 and three input control parameters as ic1, ic2, ic3 and there are two output data parameters od1, od2 and one output control parameter oc1.
Similarly class experiment having input data parameters as id1, id2 and three input control parameters as ic1, ic2, ic3 and there are no control parameters in it. So AIC can be calculated as:
AIC= (3+2)/ (3+1) + (2+0)/ (3+0) =1.95

3.2 Method Interface Coupling (MIC)

We assume that there is one class **sample** having two methods M1, M2.
M1 is having three parameters as input in which two are input data parameter id1, id2 and one is input control parameter ic1 and it is returning only one control parameter oc1
M2 is having three parameters as input in which two are input data parameter id1, id2 and one is input control parameter ic1 and it is returning only two control parameter oc1, oc2 and one data parameter od1. So MIC can be calculated as:
MIC= (2+0)/ (1+1) + (2+1)/ (1+2) =2
We assume that there is one class **experiment** having two methods M1, M2.
M1 is having three parameters as input in which two are input data parameter id1,id2 and 1 is input control parameter ic1 and it is returning only 1 control parameter oc1
M2 is having three parameters as input in which two are input data parameter id1, id2 and 1 is input control parameter ic1 and it is returning only 2 control parameter oc1, oc2 and one data parameter od1.So MIC can be calculated as:
MIC= (2+0)/ (1+1) + (2+1)/ (1+2) =2

3.3 Design Complexity

$$DC= AIC+\sum_{1}^{c} MIC$$

DC can be calculated as: DC=1.95+ (2+2) =5.95. The value 5.95 denotes the design complexity. Higher value of Design complexity shows the higher coupling means high complex project resulting high maintainability and testability cost.

4. RESULTS AND DISCUSSION

The analysis has been done on six java projects downloaded from the internet. The six packages used for analysis are

1. classifier package of Weka
2. Cluster Package of Weka
3. LibSvm
4. Minicopier
5. DependencyFinder
6. MYSQL Connector for Java-5.1.8.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The selected projects are from application field to justify the application area of specified metric in different fields. The classifier package of the WEKA library covers different algorithms of classification like C4.5 a decision tree based classifier, RBF neural network based classifier. This package is used for in the machine learning for the classification purpose. While the cluster package covers the clustering algorithms like K-mean etc. These algorithms are used to cluster the similar type of items and to separate the dissimilar items. The libSVM is the SVM classifier used for the classification purpose. The libsvm covers different kernels used for the classification purpose. These three packages are useful in the machine learning. The minicopier is used to copy the items from one location to another. This project is downloaded from the internet. This project is an example of general purpose projects used in any type of application. The dependency finder library is used to generate the dependency among the different modules of a project. This also shows the attributes and the method of a class used by another class. This package is used in the software metric evaluation. The MYSQL connector is a driver to connect the java with the mysql database. This covers the application based connectivity among two packages. The analysis on these projects covers the machine learning, software metric evaluation, database connectivity driver and the general application. All the packages have been downloaded from their respective website on internet and analysis has been done only on the classes available directly in the package.

These projects have been analyzed by using the statistics to describe the data, descriptive statistics that gives the CK metric as well as the proposed metric statistic. This statistic covers the minimum, maximum values as well as the mean and standard deviation of the corresponding metric for each project.

Table 1: CK Metric Statistic for LibSVM project

Metric	Min	Max	Mean	Standard Deviation
WMC	0	34	4.9474	7.6990
DIT	0	1	0.7368	0.4524
NOC	0	3	0.2632	0.7335
CBO	0	15	2.4737	3.5335
RFC	0	89	11.0526	20.0207
LCOM	0	543	30.2632	124.2693
Ca	1	7	2.4737	1.8669
NPM	0	18	1.3158	4.0832

Table 1 shows the statistics of total 8 metric including the 6 CK metric and Coupling (Ca) and number of public method per class (NPM) metric for the libSVM project. This has been calculated by evaluating the metric value using the CK metric evaluation tool.

Table 2: Proposed Metric Statistic for LibSVM Project

Metric	Min	Max	Mean	Standard Deviation
CPP	0	24	0.987	3.338
MPC	0	206	13.187	34.863
AIC	0	822	13.006	69.130
MIC	0	11860	203.500	1243.256
DC	0	12682	216.506	1306.971

Table 2 describes the values for proposed metric suite for the libSVM project. The values have been calculated by using 'dependency finder' tool. A large variation can be found in the design complexity of libSVM project due to large variation in method interface coupling.

Table 3: 95% confidence interval of CK metric mean for LibSVM project

Metric	Lower Limit	Upper Limit
WMC	30.289	37.711
DIT	0.782	1.218
NOC	2.646	3.354
CBO	13.297	16.703
RFC	79.350	98.650
LCOM	483.104	602.896
Ca	6.100	7.900
NPM	16.032	19.968

Table 4: 95% Confidence Interval of proposed Metric for LibSVM project

Metric	Lower Limit	Upper Limit
CPP	23.479	24.521
MPC	200.557	211.443
AIC	811.206	832.794
MIC	11665.881	12054.119
DC	12477.933	12886.067

Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The table 3 and 4 describe the 95% confidence interval value i.e. the range under which the 95% of the total values falls. It can be analyzed that the in table 3 the values of RFC and WMC is 79 and 30 respectively which signifies the reduced testability and understandability. The high value of LCOM denotes lower productivity i.e. high design efforts required for the project. In the table 4 the design complexity is large i.e. 12477 which denotes the complex design which also requires high efforts.

Table 5: CK Statistics for Classification package WEKA project

Metric	Min	Max	Mean	Standard Deviation
WMC	1	99	13.7241	18.8961
DIT	0	1	0.5862	0.5012
NOC	0	3	0.3793	0.8200
CBO	0	36	6.9655	7.2431
RFC	1	269	44.6207	54.9834
LCOM	0	2987	161.6207	554.7815
Ca	0	15	1.6552	3.1879
NPM	1	84	11.7241	16.1221

Table 5 denotes the metric statistic for the classifier package of the WEKA project. While the table 6 covers the proposed metric statistics of same i.e. classifier package of WEKA project. The table 5 shows that range of LCOM values have more deviation as compared to the LCOM value of the libSVM project.

Table 6: Proposed Metric Statistic for classification package WEKA Project

Metric	Min	Max	Mean	Standard Deviation
CPP	0	10	0.963	2.124
MPC	0	99	12.049	20.273
AIC	0	412	11.741	47.685
MIC	0	4291	172.531	709.442
DC	0	4692	184.272	749.986

Table 6 describes the values for proposed metric suite for the classification package of WEKA project. The variation in the design complexity of classifier package of WEKA project is less as compared to the design complexity of the LibSVM project. This is due to the less variation in the method interface coupling. Moreover, this clearly denotes that the classifier package of WEKA project is less complex as compared to the libSVM project. The identified ranges in the table 5 and 6 may have outliers so to get accurate range of value 95% confidence interval values has been calculated shown in table 7 and 8 for CK metric and proposed metric respectively. These tables provides the actual range of the CK metric and the proposed metric values.

Table 7: 95% confidence interval of CK metric mean for classification package WEKA project

Metric	Lower Limit	Upper Limit
WMC	91.812	106.188
DIT	0.809	1.191
NOC	2.688	3.312
CBO	33.245	38.755
RFC	248.085	289.915
LCOM	2775.972	3198.028
Ca	13.787	16.213
NPM	77.867	90.133

The 95% confidence interval value presents that the lower values of LCOM are the outliers while actual value lies at the upper range i.e. around 2775. It means the LCOM value of the project is very large. The high WMC and NPM values are also identified in the project. This shows large of public methods are available in the class which can be used any other class in the project. These values identify a complex design of classifier package as compared to libSVM project.

Table 8: 95% Confidence Interval of proposed Metric for classification package WEKA project

Metric	Lower Limit	Upper Limit
CPP	9.530	10.470
MPC	94.517	103.483
AIC	401.456	422.544
MIC	4134.129	4447.871
DC	4526.164	4857.836

The table 8 denotes the range of design complexity values is large but less than the design complexity value of the libSVM project due to the similar variation in the MIC value. This clearly denotes that the design of project is less complex as compared to the design of libSVM project. The table 9 is used to determine the CK metric value of the clustering package of WEKA

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

project. In the similar fashion the table 10 denotes the proposed metric value of the clustering package of the WEKA project. The CK tool and dependency finder tools are used to get the values of corresponding metrics. The minimum, maximum, mean and the standard deviation values of the clustering package of WEKA project for the CK metric suite is given in the table 5.

Table 9: Clustering-WEKA project

Metric	Min	Max	Mean	Standard Deviation
WMC	1	89	20.4194	19.0801
DIT	0	1	0.4194	0.5016
NOC	0	8	0.5484	1.6899
CBO	0	24	10.7742	7.7705
RFC	1	192	64.5161	57.5615
LCOM	0	3546	290.6774	656.4888
Ca	0	17	2.1935	3.5536
NPM	1	58	14.6129	14.1508

Table 9 shows the statistics of total 8 metric including the 6 CK metric and Coupling (Ca) and number of public method per class (NPM) metric for the clustering package of WEKA project. The result shown in table includes the minimum, maximum, mean and standard deviation value of each metric.

Table 10: Proposed Metric Statistic for clustering package WEKA Project

Metric	Min	Max	Mean	Standard Deviation
CPP	0	12	0.954	2.225
MPC	0	107	12.161	21.095
AIC	0	435	11.828	48.661
MIC	0	4763	176.391	752.200
DC	0	5079	188.218	793.132

Table 10 describes the values for proposed metric suite for the clustering package of WEKA project. A variation similar to the variation found in classification package of the WEKA project is found in this project.

Table 11: 95% confidence interval of CK metric mean for clustering package WEKA project

Metric	Lower Limit	Upper Limit
WMC	82.001	95.999
DIT	0.816	1.184
NOC	7.380	8.620
CBO	21.150	26.850
RFC	170.886	213.114
LCOM	3305.198	3786.802
Ca	15.697	18.303
NPM	52.809	63.191

Table 12: 95% Confidence Interval of proposed Metric for clustering package WEKA project

Metric	Lower Limit	Upper Limit
CPP	11.526	12.474
MPC	102.504	111.496
AIC	424.629	445.371
MIC	4602.684	4923.316
DC	4909.961	5248.039

The table 11 and 12 describe the 95% confidence interval value i.e. the range under which the 95% of the total values falls. The range of values doesn't show any major difference between the values obtained in the classification package and clustering package of the WEKA project. It means the clustering packages exhibits same complexity as of the classification package of the WEKA project.

Table 13: CK Metric of Minicopier project

Metric	Min	Max	Mean	Standard Deviation
WMC	0	40	9.0000	10.6344
DIT	1	3	1.5833	0.7930
NOC	0	0	0.0000	0.0000
CBO	0	11	2.4167	3.2602
RFC	0	116	25.4167	31.2190
LCOM	0	574	50.9167	164.8897
Ca	1	4	1.8333	1.1146
NPM	0	38	8.3333	10.1115

Table 13 denotes the metric statistic for the minicopier project and the table 14 denotes the proposed metric statistics of same i.e. minicopier project. The table 13 shows that range of LCOM values is less as compared to the WEKA projects.

Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 14: Proposed Metric Statistic for Minicopier Project

Metric	Min	Max	Mean	Standard Deviation
CPP	0	12	0.968	2.277
MPC	0	113	12.645	23.381
AIC	0	480	12.301	51.983
MIC	0	5125	174.462	768.349
DC	0	5239	186.763	811.415

Table 14 presents the values for proposed metric suite for the minicopier project. The variation in the design complexity of minicopier project is same as of the WEKA projects while less than the variation in design complexity of the LIBSVM projects. This is due to the less variation in the method interface coupling. Moreover, this clearly denotes that minicopier project is less complex as compared to the LIBSVM project. The identified ranges in the table 13 and 14 may have outliers so to get accurate range of value 95% confidence interval values has been calculated shown in table 15 and 16 for CK metric and proposed metric respectively.

Table 15: 95% confidence interval of CK metric mean for Minicopier project

Metric	Lower Limit	Upper Limit
WMC	33.243	46.757
DIT	2.496	3.504
NOC	0.000	0.000
CBO	8.929	13.071
RFC	96.164	135.836
LCOM	469.234	678.766
Ca	3.292	4.708
NPM	31.575	44.425

The 95% confidence interval value presents that the lower values of LCOM are the outliers while actual value lies at the upper range i.e. around 469. The less values of metric LCOM as well as CBO and RFC as compared to LIBSVM project shows the less complex project.

Table 16: 95% Confidence Interval of proposed Metric for Minicopier project

Metric	Lower Limit	Upper Limit
CPP	11.531	12.469
MPC	108.185	117.815
AIC	469.294	490.706
MIC	4966.760	5283.240
DC	5071.891	5406.109

The table 16 denotes the range of design complexity values is same as of the design complexity range of the classifier and clustering package of the WEKA project but less than the design complexity value of the libSVM project due to the similar variation in the MIC value. This clearly denotes that the design of project is less complex as compared to the design of libSVM project while the minicopier project has same complexity as of the classifier and clustering package of WEKA project.

The table 17 is used to determine the CK metric value of the MYSQL connector project. In the similar fashion the table 10 denotes the proposed metric value of the MYSQL connector project. The CK tool and dependency finder tools are used to get the values of corresponding metrics.

Table 17: Statisitcs of MYSQL Connector project for CK Metric Suite

Metric	Min	Max	Mean	Standard Deviation
WMC	1	536	26.9181	74.9505
DIT	0	6	0.9123	0.9570
NOC	0	17	0.3041	1.4104
CBO	0	59	5.5205	7.6238
RFC	1	1069	58.4561	134.7769
LCOM	0	143374	2487.5205	14042.3461
Ca	0	66	5.2339	9.7909
NPM	0	535	21.6608	71.1266

Table 17 shows the statistics of total 8 metric including the 6 CK metric and Coupling (Ca) and number of public method per class (NPM) metric for the MYSQL connector project. The range of almost LCOM metric is highest in this project as compared to all other projects being analyzed till now. It means the complexity of the project is high as compared other projects analyzed till now.

Table 18: Proposed Metric Statistic for MYSQL connector Project

Metric	Min	Max	Mean	Standard Deviation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CPP	0	22	0.985	3.066
MPC	0	148	11.843	28.900
AIC	0	612	11.687	56.691
MIC	0	9805	196.627	1105.508
DC	0	10417	208.313	1158.602

Table 18 describes the values for proposed metric suite for the MYSQL connector project. A large variation can be found in the design of MYSQL connector project due to large variation in method interface coupling. The distinguished ranges in the table 17 and 18 may have exceptions so to get exact scope of significant worth 95% certainty interim qualities has been ascertained appeared in table 19 and 20 for CK metric and proposed metric individually.

Table 19: 95% confidence interval of CK metric mean for MYSQL Connector project

Metric	Lower Limit	Upper Limit
WMC	524.686	547.314
DIT	5.856	6.144
NOC	16.787	17.213
CBO	57.849	60.151
RFC	1048.655	1089.345
LCOM	141254.212	145493.788
Ca	64.522	67.478
NPM	524.263	545.737

Table 20: 95% Confidence Interval of proposed Metric for MYSQL Connector project

Metric	Lower Limit	Upper Limit
CPP	21.476	22.524
MPC	143.062	152.938
AIC	602.313	621.687
MIC	9616.102	9993.898
DC	10219.030	10614.970

The table 19 and 20 describe the 95% confidence interval value i.e. the range under which the 95% of the total values falls. The range of values shows that the design complexity of the project is higher than the minicopier and classification and clustering package of WEKA project but somewhat lower than the libSVM project. It means the project exhibits high complexity as of the classification, clustering package of the WEKA project and minicopier project.

Table 21: CK Metric Statistic for Dependency Finder Project

Metric	Min	Max	Mean	Standard Deviation
WMC	0	69	6.6412	11.6004
DIT	0	2	0.8015	0.4710
NOC	0	7	0.2290	0.8732
CBO	0	65	4.1221	9.4836
RFC	0	274	13.9771	30.1770
LCOM	0	2340	76.7099	334.5154
Ca	0	29	3.9389	4.5989
NPM	0	61	5.7023	10.6379

Table 22: Proposed Metric Statistic for Dependency Finder Project

Metric	Min	Max	Mean	Standard Deviation
CPP	0	24	0.988	3.338
MPC	0	206	13.188	34.863
AIC	0	822	13.006	69.131
MIC	0	11860	203.500	1243.257
DC	0	12682	216.506	1306.971

Table 22 depicts the qualities for proposed metric suite for the dependency finder venture. The variety in the design complexity of dependency finder venture is in the range of the values given by the LibSVM venture. This is because of the similar variation in the method interface coupling. The recognized ranges in the table 21 and 22 may have exceptions so to get exact scope of significant worth 95% certainty interim qualities has been ascertained appeared in table 23 and 24 for CK metric and proposed metric individually

Table 23: 95% confidence interval of CK metric mean for Dependency Finder project

Metric	Lower Limit	Upper Limit
WMC	66.995	71.005
DIT	1.919	2.081
NOC	6.849	7.151
CBO	63.361	66.639
RFC	268.784	279.216
LCOM	2282.178	2397.822
Ca	28.205	29.795
NPM	59.161	62.839

Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The table 23 is used to determine the 95% confidence interval value of CK metric value for the dependency finder project. In the similar fashion the table 24 denotes the 95% confidence interval value of proposed metric value of the dependency finder project.

Table 24: 95% Confidence Interval of proposed Metric for Dependency Finder project

Metric	Lower Limit	Upper Limit
CPP	23.479	24.521
MPC	200.557	211.443
AIC	811.206	832.794
MIC	11665.881	12054.119
DC	12477.933	12886.067

The table 24 denotes the range of design complexity values is large and has same range as the complexity value range of the libSVM project due to the similar variation in the MIC value. This clearly denotes that the design of project is complex and complexity is almost same as complexity of the libSVM project. It means the project has more complexity as compared to the minicopier project and clustering, classification package of the WEKA project.

Table 25: Correlation of CK metric With Proposed Metric

Proposed/CK Metric	WMC	DIT	NOC	CBO	RFC	LCOM	Ca	NPM
CPP	-0.191	-0.029	0.301	-0.165	-0.214	-0.173	0.040	-0.196
MPC	-0.110	0.016	0.152	-0.116	-0.158	-0.035	0.380	-0.109
AIC	-0.099	0.159	-0.089	-0.105	-0.142	-0.054	0.805	-0.092
MIC	0.764	0.145	-0.056	0.710	0.733	0.791	-0.107	0.755
DC	0.768	0.164	-0.032	0.710	0.729	0.808	-0.084	0.759

In the table 25 the correlation between the CK metric and the proposed metric is found. The design complexity correlation with the CK metric determines the significance of the proposed metric. The design complexity is highly correlated i.e. 0.768, 0.710, 0.729, 0.808, 0.759 with the WMC, CBO, RFC, LCOM and NPM respectively. It means the high design complexity shows the high complex model which is also determined by the WMC, DIT, LCOM, CBO and the NPM factors. This is already seen in the analysis of the six projects. It means the design complexity metric can be used to find the complexity of any project.

5. CONCLUSIONS

This paper designs a coupling metric to determine the complexity of software. These different proposed metrics can be used to check the complexity of design at an early stage to remove the anomalies as well as redundancy of the code and hence will be helpful in better design of object-oriented system. The metric uses the method and the attribute coupling to determine the complexity of the project. The metric is understood with the help of a case study. The validation of the metric is done by determining the correlation of the metric with the CK metric. Moreover, the analysis is done on six java projects. The high correlation of the design complexity metric with the CK metric and accurate results of proposed metric on six java projects proves the significance of the metric. The future research work aims at reviewing as to how methodically tool applied on these metrics to escort the designing of difficult systems.

REFERENCES

[1] Singh S, Kaur S. A systematic literature review: Refactoring for disclosing code smells in object oriented software. Ain Shams Engineering Journal. 2017 Mar 22.

[2] Roger S. Pressman, “Software Engineering: A Practitioner’s Approach”, 6th ed., McGraw Hill International, 2005.

[3] N. Fenton and S. Lawrence Pfleeger, “Software Metrics: A Rigorous Approach”, 2nd ed., International Thomson Press, London, 1996.

[4] Preeti Gulia et al.”Design based Object-Oriented Metrics to Measure Coupling and Cohesion”, International Journal of Engineering Science and Technology

(IJEST), ISSN: 0975-5462 Vol. 3 No. 11 November 2011.

[5] Alenezi M, Zarour M. Modularity measurement and evolution in object-oriented open-source projects. InProceedings of the The International Conference on Engineering & MIS 2015 2015 Sep 24 (p. 16). ACM.

[6]. Basili, V., Briand, L., & Melo, W. (1996) A Validation of Object-Oriented Design Metrics as Quality Indicators. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, VOL. 22, NO. 10, OCTOBER 1996

[7]. Chidamber, S. R. and Kemerer, C. F (1994), A Metrics Suite for Object-Oriented Design, *IEEE Transactions on Software Engineering*, vol. 20 no. 6, pp. 476–493

[8]. K.P. Srinivasan, Dr. T.Devi (2014). A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment. *International Journal of Software Engineering and Its Applications 8(2)*, 2014, 173-188.

[9]. Vanitha N, "A Report on the Analysis of Metrics and Measures on Software Quality Factors – A Literature Study", IJCSIT, Vol. 5, 2014

[10]. Bansal Mukesh, Agarwal CP, Sasikala P (2012). Predict Software Fault Proneness Using Object Oriented Metrics. *International journal of computing, intelligent an communication technology*, ISSN 2319-748X

[11] Chiller R S, Chhikara Arti (2012). Analyzing the complexity of java programs using Object Oriented Software Metrics. IJSCI, ISSN (online):1694-0814, Vol.9, No. 3, January 2012.

[12] Gupta Deepali,Kumar Rakesh(2012) . Heuristics Based on Object Oriented (OO) Metrics. International Journal of Emerging Technology and Advanced Engineering,ISSN 2250-2459, Volume 2,Issue 5, May 2012

[13] Kulkarni et.al (2010) .Validation of CK metrics for object oriented design environment. Third International Journal of Emerging trends in Engineering and Technology, 978-0-7695-4246-1/10,2010 IEEE.

[14] Chawala Sonia (2013) .Review of MOOD and QMOOD metric set. IJARCSSE, ISSN-2277-128x, Volume 3, Issue 3, March 2013

[15]. J. Zhao and B. Xu (2004). Measuring aspect cohesion, Proceedings of 7th International Conference on Fundamental Approaches to Software Engineering (FASE'04), Lecture Notes in Computer Science, Volume 2984, Springer-Verlag, pp. 54– 68, 2004.

[16]. Li and Henry (1993), Object-Oriented Metrics that Predict Maintainability‖, Journal of Systems and Software, vol 23, no.2, pp.111-122, 1993.

[17]. Khoshgaftaar T.M, Allen, Hudepohl J and S.J. Aud (1997). Application of neural networks to software quality modeling of a very large telecommunications system." IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.

[18]. Giovanni (2000). Estimating Software Fault-Proneness for Tuning Testing Activities‖ Proceedings of the 22nd International Conference on Software Engineering (ICSE2000), Limerick, Ireland, Jun.2000

[19]. E.L. Emam, W. Melo and C.M. Javam (2001) ―The Prediction of Faulty Classes Using Object-Oriented Design Metrics‖, Journal of Systems and Software, Elsevier Science, pp. 63-75, 2001

[20]. Kumar Rakesh and Kaur Gurvinder (2011). Comparing Complexity in Accordance with Object Oriented Metrics. *International Journal of Computer Applications*, Published by Foundation of Computer Science. BibTeX 15(8):42–45, February 2011

[21]. Gyimothy, T., Ferenc, R., & Siket, I. (2005) Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, VOL. 31, NO. 10, OCTOBER 2005

[22] Farooq A, Braungarten R, Dumke RR. An empirical analysis of object-oriented metrics for java technologies. In9th International Multitopic Conference, IEEE INMIC 2005 2005 Dec 24 (pp. 1-6). IEEE.

[23] Singh G, Ahmed MD. Effect of coupling on change in open source Java systems. InProceedings of the Australasian Computer Science Week Multiconference 2017 Jan 30 (p. 22). ACM.