International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Design and Implementation of Component based Metric for Software Complexity Measurement

Sonal Gahlot, Rajender Singh Chhillar

Abstract*: This paper designs a single component based metric to measure the complexity of any software in any phase of software development life cycle. The metric is designed on the basis of existing coupling and cohesion metrics like normalized hamming code (NHD), lack of cohesion in methods (LCOM), conceptual coupling(CoCC), structural and semantic coupling metric(SSCM). The designed metric also covers the coupling between parent and its inherited class, static import, anonymous class contribution and the coupling between inner and outer class to analyze the complexity of software precisely. The analysis of the metric has been done on seven industrial and academic projects against existing state of art coupling and cohesion metric i.e. NHD, COCC, SSCM, LCOM5 and method attribute cohesion metric. The result and analysis shows the significance of the designed metric.

***Keywords* : About MMAC, LCOM5, NHD, Complexity, CoCC, SSCM, Coupling, Cohesion.**

I. INTRODUCTION

In the Current Era, the dependency on the software's has been increased tremendously in the day to day life. Currently, including business growth, everything directly or indirectly depends upon the software[1]. Due to this amount of development in the software field, the need for efficient software development has been increased[2]. The cost and Quality are the two major constraints in any software development. Both factors are highly affected by the time and effort required to develop the software. This directly depends upon the complexity of the software as less complex software needs less maintenance as well as testing. One more solution to reduce the cost of the software is to increase the reusability i.e. instead of developing the software from scratch using the existing software components to develop new software. The concept of using an existing software component to develop the new software is known as the component-based software engineering (CBSE)[3]. The component-based software engineering reduces the development time as well as the cost. The quality of the software developed by CBSE depends upon the complexity of the software which in turn depends upon the component selected. It means proper selection of software module is necessary for the development of high-quality software in less time by using CBSE[4]. Moreover, the complexity of the software can be reduced by selecting the component which can be loosely coupled with high cohesion to the new software system[5].

Coupling is the strength by which component of different modules are interconnected while the cohesion is interconnection between components of the same module. Basically, the cohesion shows the strength of the module due to its component while the coupling shows the bond between two modules as shown in fig 1.

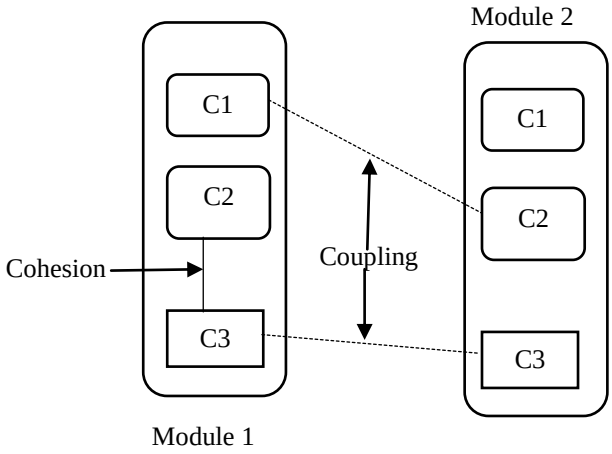

Fig 1: Coupling and Cohesion Between Software Modules

Fig. 1 also shows the concept of coupling and cohesion. There is a trade-off between coupling and cohesion for the complexity as well as the quality of a software[2]. A high-quality software should produce less complexity, coupling and high cohesion as low coupling and high cohesion reduces the testing and maintainability cost[6]. Coupling, cohesion as well as the complexity of software can be measured by using the software metric[7]. Software metric plays a significant role to control the quality of software this is due to the fact that improvement needs measurement which is done through the metric. To control the software quality various software metric already has been by different authors by measuring the coupling, cohesion and the complexity of software.

LCOM is the lack of cohesion in methods, is the one the CK suite metric used to measure the lack of cohesion by dividing the different methods with the total number of methods[8]. LCOM metric is modified by various authors, LCOM5 is the latest metric to define the lack of cohesion which is given by (1).

Manuscript published on 30 September 2019
* Correspondence Author
Sonal Gahlot*, Department of Computer Science and Application, M.D.U Rohtak, India. Email: sonalght@gmail.com
Rajender Singh Chhillar, Department of Computer Science and Application, M.D.U Rohtak, India. Email: chhillar02@gmail.com

$$LCOM5 = \frac{m - nm * na}{nm - nm * na} \quad (1)$$

Here, nm, na are the number of method and number of attributes for any particular class while m gives the number of attributes used by each method of class commonly.
CohV [9] is the metric used to measure the cohesion between the variables by dividing the frequency of variable usage by the total number of variables. It is given by (2)

$$CohV = \frac{\sum Variables\ Used}{Total\ Variables} \quad (2)$$

Equation (1) is computed by focusing on a single task and the cohV gives the results for a particular task only. In a similar way, cohM[9] gives the cohesion between the methods by calculating the number of methods that uses the same type of variables. It is given by (3)

$$CohM = \frac{\sum Methods\ uses\ common\ Variables}{Total\ Methods} \quad (3)$$

Equation (3) gives the cohesion between the methods for a particular task only. CAMC[10] metric is an acronym for the cohesion among methods in a class by computing the average of parameter occurrence metric. The CAMC metric is totally dependent on the parameter list of a method which generates the parameter occurrence metric by the (4).

$$\partial = \sum_{a=1}^{m} \sum_{b=1}^{n} V_{ab} \quad (4)$$

Here ∂ gives the parameter occurrence metric and V_{ab} is the value entry for the a^{th} row and b^{th} column, which is 1 only if the parameter of a row a occurs the parameter list of the method of b column. The total parameter and method assumed in above are m, n respectively. This parameter occurrence metric is used to calculate the CAMC metric by (5).

$$CAMC = average(\partial) \quad (5)$$

The CAMC calculated by (5) is for a class only and it the average values as denoted by the equation itself. The metric NHD [11]stands for normalized hamming distance is the alternate method to measure the cohesion. It uses the count of methods that uses the common occurrence of the parameters. It is computed on the basis of the parameter agreement metric which is a lower triangular metric of the parameter occurrence metric. The NHD is computed by (6):

$$NHD = 1 - \frac{2}{mn(n-1)} \sum_{a=1}^{m} c_a (n - c_a) \quad (6)$$

Here, NHD is derived for any class and c_a is the number 1's in the ath column while m, n are total parameter and methods as already defined in the previous equation. Here, $\frac{2}{mn(n-1)} \sum_{a=1}^{m} c_a (n - c_a)$ calculates the parameter disagreement value when subtracted from 1 gives the NHD. The high cohesion gives the higher value of the NHD and NHD is more significant as compared to the CAMC metric[8].
MMAC stands for the method attribute cohesion [10]. It basically gives the cohesion between the methods through the attributes. MMAC can be computed by the method invocation metric. Method invocation (MI) metric has rows and columns equal to the number of methods and entry to any cell is one only if the corresponding function call the other function directly or indirectly. In other words, if the Mab is 1 then function b^{th} is called by the a^{th} function either directly or indirectly. The MMAC is given by (7).

$$MMAC(c) = \begin{cases} 0 & nm = 0\ or\ na = 0 \\ 1 & nm = 1 \\ \frac{2}{nm(nm-1)} \sum_{a=1}^{nm-1} \sum_{b=a+1}^{nm} MI(a,b) & else \end{cases} \quad (7)$$

Where, nm, na are the number of methods and attributes in class c respectively. MI is the methods invocation matric as defined above. On using the method invocation details the (7) can be rewritten to (8).

$$MMAC(c) = \begin{cases} 0 & nm = 0\ or\ na = 0 \\ 1 & nm = 1 \\ \frac{2}{nm * na(na-1)} \sum_{a=1}^{nm} y_a (y_a - 1) & else \end{cases} \quad (8)$$

Here, y_a is the number of 1's in the a^{th} column of MI matric. These the cohesion metric defined by various authors, few coupling metrics to calculate the coupling among the components are as follow.
CoCC [12] is the conceptual coupling metric which is computed on the basis of conceptual similarity between two classes (CSBC). CSBC depends upon the conceptual similarity between the class and method which is computed on the basis similarity between the methods of a class. It means the conceptual similarity can be calculated by evaluating the conceptual similarity between the methods. The CoCC is given by (9).

$$CoCC(c) = \frac{\left(\sum_{a=1}^{n} CSBC(c, b_a) \,|c \neq b_a\right)}{n-1} \quad (9)$$

Here, CSBC is the similarity between the classes and n is the number of classes.
SSCM [13] is the structural and semantic coupling metric which includes the semantic as well as the structural relation between the methods and classes. The structural relation is the dependency of other entities on evaluated method and dependency of evaluated method on the other entities. The semantic relation uses the semantic data like identifiers and comments to find the relationship between two entities. SSCM is computed by the hybrid coupling between two classes that is evaluated on the basis of method pair coupling, direct dependency and the cosine relationship. It is given by (10).

$$SSCM = \frac{\left(\sum_{a=1}^{n} HCBC(c, b_a) \,|c \neq b_a\right)}{n-1} \quad (10)$$

Where, HCBC is the hybrid coupling between classes and n is the number of classes. These the existing metric to compute the coupling and cohesion effectively but no metric has been defined to compute the complexity directly on the basis of coupling and cohesion metric. This paper designs a metric to compute the complexity of any interface discussed in next section.

II. PROPOSED WORK

The existing metric either measure the cohesion or the coupling efficiently. The complexity of any project computed through an existing metric can be improved by focusing on the coupling as well as the cohesion. Moreover, the coupling of the attributes and methods in the inherited classes and parent classes along with static import affects the complexity of the project. The complexity of the project is also affected by the inner classes along with anonymous classes. The proposed metric considers all the above factors and computes the complexity of the project efficiently. The proposed metric is given by (11).

$$CI = \frac{\sum CC_o + \sum CC_i + \sum CC_a}{n1+n2+n3} \quad (11)$$

equation (11) gives the complexity of interface metric which is calculated by computing the class complexity of each outer class (CC_o), class complexity of each inner class (CC_i) and class complexity of each anonymous class (CC_a) where the CC_o, CC_i, CC_a is given by equation (12), (14) and (15) respectively. The sum of the CC_o, CC_i, CC_a is divided by the sum of n1, n2, and n3 where n1, n2, and n3 are number of outer classes, number of inner classes and number of anonymous classes respectively. The value of CI varies from 0 to 1 and the value close to 0 exhibits low complexity projects and vice-versa.

$$CC_o = \vartheta_1 * NHD + \vartheta_2 * LCOM5 - \vartheta_3 * SSCM - \vartheta_4 * CoCC + \vartheta_5 * IC \quad (12)$$

Such that $\vartheta_1 + \vartheta_2 + \vartheta_3 + \vartheta_4 + \vartheta_5 = 1$.

NHD, LCOM5, SSCM, CoCC metric are given by the (6), (1), (10), (9) respectively. The metric IC is computed by the coupling and cohesion between the components of parent and the child class. It is given by the (13).

$$IC = \frac{O_m + a_{co}}{m_p + a_p + m_c + a_c} \quad (13)$$

Here, O_m is the overridden methods while a_{co} is the common attributes. The m_p, a_p, m_c, a_c are the methods and attributes of parent and child class respectively. The CC_i is given by (14).

$$CC_i = \vartheta_1 * NHD_i + \vartheta_2 * LCOM5_i - \vartheta_3 * SSCM_o - \vartheta_4 * CoCC_o \quad (14)$$

Such that $\vartheta_1 + \vartheta_2 + \vartheta_3 + \vartheta_4 = 1$

In the inner class SSCM, CoCC is calculated only for the outer class while NHD and LCOM is computed for inner class only as shown in (14). In the anonymous class, there is no inheritance and the coupling so only cohesion is computed as shown in (15).

$$CC_a = \vartheta_1 * NHD_a + \vartheta_2 * LCOM5_a \quad (15)$$

Such that $\vartheta_1 + \vartheta_2 = 1$

The NHD and LCOM5, both are computed only for the anonymous class only as given by the (15). This proposed metric computes the complexity of any software. The implementation and analysis of the metric is done in next section.

III. IMPLEMENTATION AND ANALYSIS

The implementation of the proposed metric has been done by using the MATLAB along with the JPeek and Code MR tool. The Jpeek tool is used to compute the cohesion metric value i.e. NHD, LCOM5. The SSCM and CoCC is computed by MATLAB program with takes the input from Code MR tool. The implementation has been done on different academic and industrial projects. The details of the projects are given in the table 1.

Table 1: Projects used for Analysis

Sr. No.	Project Name	Description	Application
1	AutoCAD-D	Academic Project	Mechanical Drawing
2	Facebook-D	Academic Project	Social Media
3	Friends Tracker-D	Academic Project	Social Media
4	Web Intranet Manager	Academic Project	ERP
5	jFree Chart	Industrial Project	Representation and Charting
6	Finding Bugs	Industrial Project	Code Analyzer
7	HTML Parser	Industrial Project	Code Parser

Table 1 shows the name of projects for analysis. The description represents that whether the project is academic or industrial. All the academic projects have been designed by the students of DPG institute of technology and Management, Gurugram. Different academic project covers the project of different application areas' like AutoCAD-D is a replica of the original AutoCAD with limited functions used to design the different machine drawings. Similarly, The Facebook-D is the replica of original Facebook with less functionality covers the social media application. The Friends Tracker-D are the applications for Social media and face recognition as shown in table 1. The jFree Chart is the industrial project available over internet to draw the charts on given data. Web intranet manager is a ERP project design by a startup company to provide the intranet interaction and sharing between any company employees. Finding Bugs ia code analyzer project to find the bugs in a code. This project is also availed through internet.

These project has been evaluated using the tools specified above to determine the NHD, LCOM5, MMAC SSCM, CoCC metric. The details of the projects are mentioned in the table 2.

Table 2: Detail of Projects used for Analysis

Sr. No.	Project Name	Number of Classes	Number of Packages	Number of External Packages	Number of External Classes	Line of Code
1	AutoCAD-D	19	9	21	149	1629
2	Facebook-D	6	1	11	45	446
3	Friends Tracker-D	47	8	26	112	2063
4	Web Intranet Manager	15	1	5	17	1327
5	jFree Chart	650	40	38	269	71692
6	Finding Bugs	1426	55	60	714	90152
7	HTML Parser	14	1	4	16	648

Table 2 shows the number of classes along with the number of packages in the project being evaluated. The classes which are imported through the external packages are also given in the table. The line of code to estimate the size of project is also shown in the table. The overall analysis shows that evaluation project varies in size from 446 line of code to 90152 line of code. This is done to show the scaling factor that evaluation can be done easily on any project. Scaling also has been shown through the fig 2, 3 that reflect the package and modularity view of the web intranet manager and friends tracker-D project respectively. The fig 2 shows each class if linked through a same package while the fig 3 shows that different modules of the friends tracker-D projects exhibits coupling as well as the cohesion.

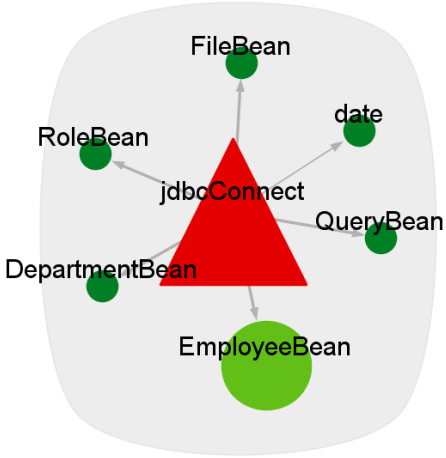

Fig 2: Package Dependency View for Web Intranet Manager Project

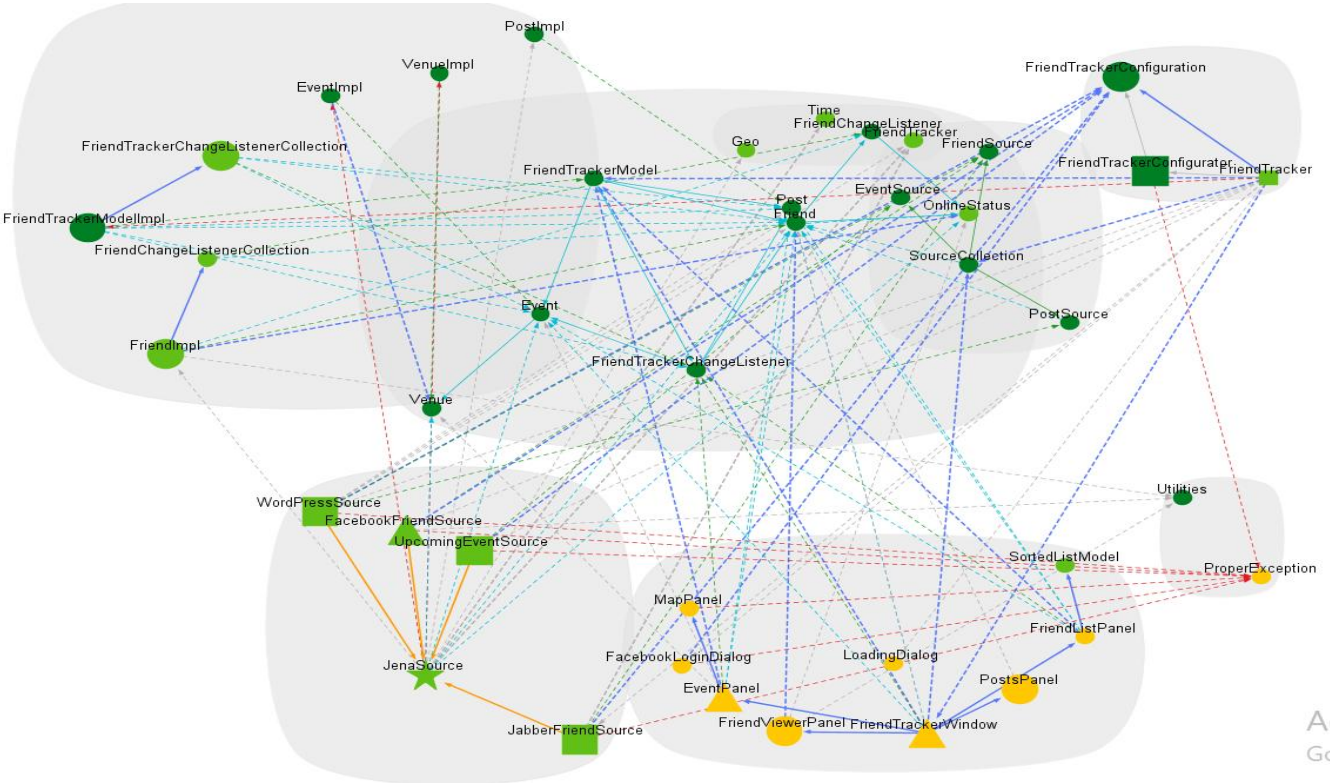

Fig 3: Modularity View of Friends Tracker Project

Fig 2 and 3 are generated through the CodeMR tool to shows the existing coupling and cohesion amoung the analyzed projects. The green color of the class shows that it is not increasing the complexity of the project, While the yellow denotes it imcreases the complexity of project In the similar way, if the color is red means that this is a problemastic class that increases the overall project complexity. Table 3 shows the value of proposed metric for the projects shown in the table 1.

Table 3: Proposed Metric for Projects

Sr. No.	Project Name	CC_o	CC_i	CC_a	CI
1	AutoCAD-D	0.315	0.080	0.047	0.147
2	Facebook-D	0.764	0.193	0.115	0.357
3	Friends Tracker-D	0.949	0.240	0.142	0.444
4	Web Intranet Manager	0.568	0.144	0.085	0.266
5	jFree Chart	0.062	0.016	0.009	0.029
6	Finding Bugs	0.137	0.035	0.021	0.064
7	HTML Parser	0.230	0.058	0.034	0.108

Table 3 shows the value of proposed metric i.e. CI calculated by using the CC_o, CC_i, CC_a. The complexity of the project friends tracker –D is highest as it is close to 1 compared to other projects. It is due to the high coupling exhibited by the project with low cohesion as shown in the fig 3. While the complexity of jfree Chart project is lowest among all the projects . It can also be seen that complexity of industrial projects is less as compared to the complexity of the academic projects. The proposed metric CI can be compared with the other metric to show the significance of the metric.

Table 4 compares the CI metric with the existing state of art metric i.e. NHD, LCOM5, MMAC, SSCM, COCC. All the metric already has been satetd in the inroduction part of the paper.

Table 4: Comparative Analysis of Designed Metric

Sr. No.	Project Name	NHD	LCOM5	MMAC	SSCM	CoCC	CI
1	AutoCAD-D	0.00	0.00	0.00	0.00	0.10	0.147
2	Facebook-D	0.00	0.50	0.00	0.50	0.12	0.357
3	Friends Tracker-D	7.12	0.79	0.50	2.85	4.45	0.444
4	Web Intranet Manager	2.50	0.50	0.50	0.50	0.85	0.266
5	jFree Chart	7.06	0.50	0.54	0.00	4.02	0.029
6	Finding Bugs	5.39	1.28	0.79	5.25	9.5	0.064
7	HTML Parser	1.17	0.72	0.50	4.12	3.87	0.108

Table 4 shows that high hamming distance metric value in the jfreechart project with 0 structral and sematic coupling while high coupling is exhibited by the friends tracker project. The analysis shows that doesn't compute the complexity of project precisely which is being calculated by the Ci metric effectively.

IV. CONCLUSION

This paper designs CI metric to compute the complexity of interface at any stage of project by computing the coupling and cohesion among the components of project. The analysis has been done on the seven academic and industrial projects which exhibits high complexity of the academic projects as the industrial projects. The comparative analysis of the proposed metric against the existing state of art metric i.e. NHD, MMAC, SSCM, CoCC, LCOM5 shows that the proposed metric computes the complexity of any project effectively. In future other metric can also be combined to improve product based metrics.

REFERENCES

1. I. G. Czibula, G. Czibula, D. Miholca, and Z. Onet-marian, "The Journal of Systems and Software An aggregated coupling measure for the analysis of object-oriented software systems," vol. 148, pp. 1–20, 2019.
2. P. Gandhi, K. Vashisht, K. Dawra, S. Jaitly, and P. Banerjee, "Component Based Software Development – ' An Efficient Approach ,'" International Journal of Scientific Engineering and Science, vol. 2, no. 1, pp. 26–32, 2018.
3. A. Khan, K. Khan, M. Amir, and M. N. A. Khan, "A component-based framework for software reusability," International Journal of Software Engineering and its Applications, vol. 8, no. 10, pp. 13–24, 2014.
4. J. Kaur and P. Tomar, "Clustering based Architecture for Software Component Selection," International Journal of Modern Education and Computer Science, vol. 10, no. 8, pp. 33–40, 2018.
5. S. Tiwari and S. S. Rathore, "Coupling and Cohesion Metrics for Object-Oriented Software," Proceedings of the 11th Innovations in Software Engineering Conference on - ISEC '18, pp. 1–11, 2018.
6. I. Chowdhury and M. Zulkernine, "Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities," Journal of Systems Architecture, vol. 57, no. 3, pp. 294–313, 2011.
7. A. Sellami, A. Majdi, E. Mohamed Ahmed, and W. Abubaker, "Complexity Metrics for Component-based Software Systems: Developer Perspective," Indian Journal of Science and Technology, vol. 11, no. 32, pp. 1–7, 2018.
8. H. Izadkhah and M. Hooshyar, "Class Cohesion Metrics for Software Engineering : A Critical Review ∗," Computer Science Journal of Moldova, vol. 25, no. 1, pp. 44–74, 2017.
9. P. Rana and R. Singh, "Comparative Analysis of Cohesion Metrics for," Journal of Theoretical and Applied Information Technology, vol. 96, no. 14, pp. 4369–4378, 2018.

10. J. Al and L. C. Briand, "An object-oriented high-level design-based class cohesion metric," Information and Software Technology, vol. 52, no. 12, pp. 1346–1361, 2010.
11. S. Counsell, S. Swift, and J. Crampton, "The interpretation and utility of three cohesion metrics for object-oriented design," ACM Transactions on Software Engineering and Methodology, vol. 15, no. 2, pp. 123–149, 2006.
12. D. Poshyvanyk and A. Marcus, "The Conceptual Coupling Metrics for Object-Oriented Systems."
13. "Software product quality assessment using scoped class cohesion metric (sccm) raphael ngigi wanjiku Master of Science (Software Engineering) Jomo Kenyatta University Of Agriculture And Technology" 2017.