International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.6 (2014), pp.367-376
http://dx.doi.org/10.14257/ijmue.2014.9.6.35

New Class Cohesion Metric: An Empirical View

Sandip Mal[1] and Kumar Rajnish[2]

[1]*Research Scholar, Department of Computer Science & Engineering*
[2]*Assistant Professor, Department of Computer Science & Engineering*

Sandip.mal1987@gmail.com, krajnish@bitmesra.ac.in

Abstract

Cohesion is an Object-Oriented (OO) software design property that helps for the measuring of degree of interdependency or connectivity within subsystems of a system. Numerous class cohesion metrics can be found in the literature. Which metric is best suited for a given situation is always a critical question. Few metrics are validated empirically against open source software projects. The purpose of this paper is to validate empirically of the proposed new class cohesion metric (CC) using some open source software projects and find the effected quality factors. Results of this paper conclude that CC continuously gives better correlation with Number Line of Code (NLOC) compare to other existing cohesion metrics. The average value of CC (CohS) of a system also predicts the natures (understandability, modifiability, and maintainability) of a system.

Keywords: *Cohesion Metrics, Reusability, Software quality, Complexity, Modifiability*

1. Introduction

Cohesion measure is the degree of interaction and relationship among classes, methods, and attributes in OO software system. High cohesion is the main goal for designing a good OO system. Since the last decade, OO programming languages, such as C++ and Java, have become widely used in both the software industry and research fields. In an OO paradigm, classes are the basic modules. Therefore, class cohesion refers to the relatedness of class members. A class with high cohesion cannot be easily split into separate classes (Dallal and Briand, 2009). Highly cohesive classes are more understandable, modifiable, and maintainable (Chen *et al.*, 2002; Gui and Scott, 2006).

Cohesion is a measure of the extent to which the various functions performed by an entity are related to one another. Most metrics assess this by considering whether the methods of a class access similar sets of instance variables. Dallal (2012) incorporates a transitive relation between classes to measure cohesion. In cohesion metrics, it should be noted that three of them (LCOM, LCOM3 and LCOM5) are in fact measures of lack of cohesion and proposed by Chidamber and Kemerer (1994). TCC (Badri and Badri, 2004) in contrast to the other three metrics, measures cohesion rather than its absence. In other respects it is similar to LCOM5, being the number of similar method pairs divided by the total number of method pairs. Other cohesion metrics based on software design has given by Chae *et al.* (2000), Hitz and Montazeri (1995), Briand *et al.* (1998), Fernandez and Pena (2006), Bansiya *et al.* (1999). Bieman and Kang (1995) measured reusability of a software system using cohesion metrics. Bonja and Kidanmariam (2006) proposed new cohesion metrics from the similar methods of a class. Counsell *et al.* (2006) provided utility and interpretation of some metrics. Related work in the area of measuring software quality can be found in Counsell *et al.* (2006), Rine and

ISSN: 1975-0080 IJMUE

Nada (2000). Neamtiu *et al.* gave empirical study on some open source software systems. Wieczerzycki (1996) and Li (1997) worked on software reusability measurement. The intuitive notion of cohesion is the extent to which the modules that make up a system are cohesive. The obvious way to assess this is to consider whether the methods of a class access similar sets of instance variables. Table 1 summarizes the characteristics of the cohesion metrics used in the comparative study. We have chosen TCC, LCC, DCD, DCI, Coh as a valid cohesion metrics because these metrics are validated by all the properties given by Briand et al in 1998. Other cohesion metrics does not satisfy all the properties given by Briand.

Table 1. Existing Cohesion Metrics

Name	Definition
Tight Class Cohesion (TCC), Loose Class Cohesion (LCC) (Bieman and Kang, 1995)	TCC considers two methods to be connected if they share the use of at least one attribute directly. A method uses an attribute if the attribute appears in the method's body or the method invokes another method, directly, which has the attribute in its body. LCC considers two methods to be connected if they share the use of at least one attribute directly or transitively.
Degree of Cohesion Direct (DCD) and degree of Cohesion Indirect (DCI)	These are similar to TCC and LCC, respectively, but differ by considering two methods connected also when both of them directly or transitively invoke the same method.
Coh (Briand et al, 1998)	Briand et al. propose a cohesion metric (called Coh) that computes cohesion as the ratio of the number of distinct attributes accessed in methods of a class

The rest of the paper is organized as follows. Section 2 presents proposed cohesion metric. Necessary cohesion properties and theoretical validation of proposed cohesion metrics has been described in Section 3. Section 4 provides case study on five open source software system. Finally, Section 5 deals with conclusion and future scope respectively.

2. Proposed Cohesion Metrics

A class *C* may consists of the set of global variables V = {v_1, v_2, v_3… v_n} and the set of methods M = {m_1, m_2, m_3 ….m_n}. The metric CC proposed for measuring cohesion of a class that satisfies the following two requirements, viz., first, it gives values that can be uniquely interpreted in terms of cohesion, and, second, the values would be within a range of 0 to 1. The value 0 would signify minimum cohesion and 1 the maximum cohesion [22]. To evaluate CC, first calculate Cohesion Value of a global variable i^{th} of a class (CV_i). CV_i value is defined when V and/or M are/is non-empty set, otherwise value of CV_i is zero.

$$CVi = \frac{No\ of\ functions\ share\ the\ global\ variable\ i\ of\ a\ class}{Total\ no\ of\ function\ of\ the\ class}$$

CV_i thus gives the ratio of the number of functions share by i^{th} global variable of a class to the total number of function of a class.
Next define the mean CV_i, Cohesion Count of a class of n global variable (CC) for a class C that is calculated as,

$$CC = \frac{\sum_1^n CVi}{n}$$

Where, *n* represents the number of global variables of a class.

It is obvious that in a class containing all methods access all global variables, the CC value would be 1. This is the case of perfect (or, maximum) cohesion. On the other extreme

- Consider a class with empty set V and/or set M in which case, CC value would be 0.
- Consider a class with no relation between methods and global variables, and then CC value would be 0.

In the above two cases, Cohesion value is minimum. Now cohesion of a system (CohS) of r classes is defined as

$$\text{CohS}=\frac{\sum_1^r CC}{r}$$

Certain criteria have been provided for measuring the quality factors of OO design systems. A system with low *CohS* then *t*he classes of the system are low cohesive in nature and if *CohS* is high then the classes of the system are tightly cohesive in nature. As discussed in [2, 4] the low cohesive system is less reusable, understandable, modifiable, and maintainable.

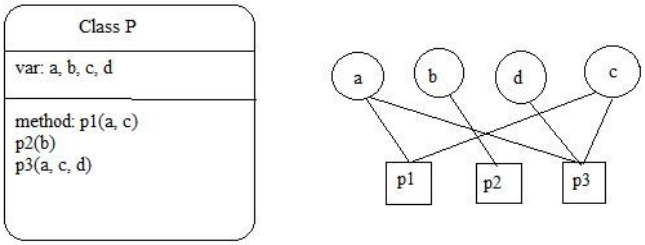

Figure 1. Example of Cohesion Measure of a Class

Figure 1 shows a class with four variables and three methods. P1 access the variable and c. p2 access the variable b. p3 access the variable a, c and d.
So, CV_a= 2/3, CV_b= 1/3, CV_d = 1/3, CV_c = 2/3
CC = (2/3 + 1/3 + 2/3 + 1/3) / 4 = 0.5

2.1 Algorithm for Measuring CC and CohS

Step1: Count the number of function and global variable of a class and assign values M and C respectively.

Step2: if M or C is zero then CC is zero.

Step3: For each global variable

1. Count the number of method (M_i) invoked global variable i.
2. Find ratio of (M_i/ M) and store it in R_i

Step4: Add all R_i to find CC.

To find CohS

Step1: First store number of classes in a variable (P).

Step2: Add all CC of a system.

Step3: Added value of all CC is divided by P.

3. Theoretical Validation of Proposed Cohesion Metrics

Briand *et al.* [11] defined four properties for cohesion metrics. The first property, Property 1, called *non-negativity* and *normalization*, is that the cohesion measure belongs to a specific interval [*0, Max*]. Normalization allows for easy comparison between the cohesion of different classes. The second property, Property 2, called *null value* and *maximum value*, holds that the cohesion of a class equals *0* if the class has no cohesive interactions; the cohesion is equal to *Max* if all possible interactions within the class are present. The third property, Property 3, called *monotonicity*, holds that adding cohesive interactions to the module cannot decrease its cohesion. The fourth property, Property 4, called *cohesive modules*, holds that merging two unrelated modules into one module does not increase the individual modules cohesion. As an example, given two classes, *c1* and *c2*, the cohesion of the merged class *c'* must satisfy the following condition:

$$cohesion\ (c') \leq max\ \{\ cohesion\ (c1),\ cohesion\ (c2)\}$$

CC metric satisfy necessary properties for class cohesion and comparison with other existing metrics given in Table II also described by [22]. The results show that CC metric satisfies all the properties given by Briand *et al.* [11]. The analysis of results shows that 35% of the considered metrics are valid from the theoretical perspective. All other metrics have to be revised to comply with the class cohesion properties. Otherwise, use of these metrics as cohesion indicators is questionable.

Table 2. Theoretical Validation Results of Existing and Proposed Cohesion Metrics

Metrics	**P1**	**P2**	**P3**	**P4**
LCOM1	NO	Yes	Yes	Yes
LCOM2	NO	Yes	Yes	Yes
LCOM3	NO	NO	Yes	Yes
LCOM4	NO	NO	Yes	Yes
Connectivity	NA	Yes	Yes	Yes
LCOM4 + Connectivity	NO	Yes	Yes	Yes
LCOM5	NO	Yes	Yes	NO
TCC	**Yes**	**Yes**	**Yes**	**Yes**
LCC	**Yes**	**Yes**	**Yes**	**Yes**
DCD	**Yes**	**Yes**	**Yes**	**Yes**
DCI	**Yes**	**Yes**	**Yes**	**Yes**
Coh	**Yes**	**Yes**	**Yes**	**Yes**
SCOM	Yes	Yes	NO	Yes
Class Cohesion	Yes	Yes	NO	Yes
CAMC	NO	NA	Yes	Yes
NCAMC	Yes	NA	Yes	Yes
NHD	NO	NA	NO	NO
CBMC	Yes	Yes	NO	Yes
ICBMC	**Yes**	**Yes**	**Yes**	**Yes**
CC	***Yes***	***Yes***	***Yes***	***Yes***

4. Case Study on Open Source Software Systems

Five open source software projects have been chosen from (http://www.sourceforge.net/projects/) for case-study. The basic information about these five projects is given in Table 3. *System1* has*16 classes*, *System2* has *57 classes*, *System3* has *23 classes*, *System4* has *85 classes*, and *System5* has *19 classes*. Total 200 classes have been taken from five systems.

Table 3. Information about Project Taken for Case Study

Software Project	No of Classes
System1	16
System2	57
System3	23
System4	85
System5	19

4.1 Effort Required for Modification

In this section we calculate the number of changes made to the original code and the time required carrying them out. Both of these were recorded. Time would appear to be a better measure of the overall effort entailed. The number of changes was defined to be the number of lines of code (Extended NLOC) that were added, deleted or modified; the time required was simply the time in minutes taken to determine and carry out the requisite changes. Total 200 classes of five systems have been modified with a total 3491 lines in 774 hours by a group of five experienced Java programmers. The results obtained on the 200 classes reveals an almost perfect linear relationship between NLOC and time (Pearson correlation = 0.987). In this paper, we focus on the correlations of NLOC with cohesion metrics. The time and NLOC required by the programmer to extend the systems given in Table 4.

Table 4. Extended Number Line of Code (NLOC) and Time Required Modifying Systems

System	Extended NLOC				TIME			
	Total	**Mean**	**Max**	**Min**	**Total**	**Mean**	**Max**	**Min**
System 1	197	8.95	20	5	47	2.1	4	1
System 2	485	8.6	18	4	142	2.51	5	1
System 3	280	14	30	5	61	3.05	5	1
System 4	2432	28.62	45	3	497	5.85	9	1
System 5	197	8.95	20	5	47	2.1	4	1

4.2 Results

The CC metric has been applied to each class of five software projects. Although CC metric was defined for classes rather than complete systems, the average value (CohS) of CC for all the classes in a system was used in the experiment for whole system. CC is used here to predict reusability of a class in a system. Table V shows the CohS value of each system. Mean, Maximum (max), Minimum (min), and Standard deviation (stdv.) value of CC, TCC, DCD and Coh of five systems is given in Table 6. CohS value of *System3* and *System4* is greater than 0.5, it indicates that maximum number of classes in those system are tightly

cohesive in nature. So, these systems are more reusable, and easy to understand, modify, and maintain than the other systems (System1, System2 and System3).

Table 5. CohS Values of Five Systems

System	CohS	Number of Classes
System1	0.21	16
System2	0.46	57
System3	0.69	23
System4	0.64	85
System5	0.49	19

Table 6. Mean, max, min, and stdv. Values of CC of Five Systems

Metric	Mean	Max	Min	Stdv.
CC	0.65	1	0	0.28
TCC	0.47	12	0	1.05
DCD	0.35	1	0	0.27
Coh	15.8	838	0	77.26

4.3 Empirical Validation

Two approaches were used to evaluate the performance of the various measures in predicting reusability of a class: Pearson correlation and linear regression. Since the objective is to find a metric that will rank components according to the amount of effort likely to be required for modification, we therefore computed the Pearson Correlation coefficients between the NLOC and those produced by the various cohesion measures and set the minimum coefficient value for each system (System1= 0.121, Syatem2= 0.340, System3= 0.501, System4= 0.338, System5= 0.360). Table 7 shows the Pearson Correlation value of each system. This minimum value of each system is taken to validate CC metric in table 8. Table 8 shows the Pearson correlation values of cohesion measures against extended NLOC. The metrics with Pearson correlation values of cohesion measures against extended NLOC is greater than the Pearson correlation values of cohesion measures against original NLOC is taken as valid metrics for reusability. Table 8 shows large Pearson correlation of CC in all system than minimum correlation value. So CC metric is a good predictor of reusability comparatives to other metrics.

Table 7. Pearson Correlations of Cohesion Measures against NLOC. ** Denotes Significant at 1% Level; * Denotes Significant at 5% Level

Measure	TCC	DCD	Coh	CC
System1	**0.121**	-0.341	0.068	-0.486
System2	-0.079	0.048	**0.340**	-0.358
System3	0.008	-0.004	**0.501**	0.264
System4	0.316	0.100	0.109	**0.338**
System5	-0.209	-0.243	0.178	**0.360**

Table 8. Pearson Correlations of Cohesion Measures against extend NLOC. ** Denotes Significant At 1% Level; * Denotes Significant at 5% Level

Measure	TCC	DCD	Coh	CC
System1	**0.423**	-0.198	**0.541***	**0.75****
System2	0.135	0.305*	0.181	**0.415****
System3	0.6**	0.12	0.091	**0.528***
System4	**0.630****	**0.634****	-0.122	**0.746****
System5	0.328	**0.712****	-0.069	**0.856****

It also be noticed that TCC, and CC measures are direct proportion to the effort required to modify a class of a system. Thus the Pearson Correlations of these measurements against extend NLOC are positive values while the Pearson Correlations of other measures (DCD, Coh) are negative values. The proposed new cohesion measures for a class, CC perform better than all the other metrics for all five systems.

It could be the case that they achieve such high rank correlations only because they are better at correctly ordering items that in fact differ very little in the amount of effort they require for modification. If this were the case, then the other metrics could be just as effective in allowing users to choose components that can be readily modified. In order to address this possibility, it is necessary to consider how effective the various metrics are in predicting the amount of modification effort required. Table 8 shows the values of the coefficient of determination (R^2) obtained when NLOC is regressed against each of the four cohesion metrics. R^2 expresses the proportion of the variance in NLOC that is explained by the metric. The results are similar to those obtained in rank correlation. The new metrics, CC is the best predictors of the amount of modification effort required. Most of the R^2 values are highly significant, indicating that the corresponding measure is good linear predictors of modification effort.

Table 9. R2 Correlations of Cohesion Measures against NLOC. ** Denotes Significant At 1% Level; * Denotes Significant At 5% Level

Measure	TCC	DCD	Coh	CC
System1	0.197	0.039	0.292*	0.562**
System2	0.018	0.093	0.033	0.164
System3	0.466*	0.024	0.008	0.562**
System4	0.397	0.402*	0.015	0.557**
System5	0.107	0.507**	0.005	0.732**

These results clearly demonstrate that the proposed cohesion metric is a very good predictor of the effort required to make simple modifications of classes of software projects. It performs better than all of other established metrics used in the study.

5. Conclusion and Future Work

In this paper, an attempt has been made to propose a new cohesion metric which is based on formal definitions, properties of classes. In addition to the proposal, this paper has also presented empirical data on CC and CohS from five open source software projects. All systems have developed in java. From Table 7 and Table 8, it is found that there is a strong correlation between CC and reusability in cases, Pearson correlation and linear correlation. So, this study clearly provided that CC is the valid indicator of external quality attributes of the classes of projects such as reusability. The mean value of CC of a system (CohS) indicates that s*ystem3* and s*ystem4* are more understandable, modifiable, and maintainable than the

system1, *system2* and *system5*. This firmly believes us that this work will encourage other researchers and developers to use the results obtained from this study to predict and measure several other software quality attributes.

The future scope includes some fundamental issues

- To analyze the nature of proposed metric with performance indicators such as design, maintenance, effort and system performance.
- Another interesting study would be together different OO cohesion metric at various intermediate stages of the project. This would provide insight into how application reusability, maintainability, testability evolves and how it can be managed and controlled through the use of metrics.

References

[1] J. A. Dallal and L. Briand, "A precise method-method interaction-based cohesion metric for object-oriented classes", TR, Simula Research Laboratory, ACM Transactions on Software Engineering and Methodology (TOSEM), in press (**2009**).
[2] Z. Chen, Y. Zhou and B. Xu, "A novel approach to measuring class cohesion based on dependence analysis", Proceedings of the International Conference on Software Maintenance, (**2002**), pp. 377-384.
[3] J. A. Dallal, "Mathematical Validation of Object-Oriented Class Cohesion Metrics", International Journal of Computers, vol. 4, Issue 2, (**2010**), pp. 45-52.
[4] G Gui and P. D. Scott, "Coupling and Cohesion Measure for Evaluation of Component Reusability", MSR'06, May 22-23, Shanghai, China, (**2006**), pp. 18-21.
[5] J. A. Dallal, "Incorporating transitive relations in low-level design-based class cohesion measurement", software – practice and experience, vol. 43, Issue 1, published online in Wiley Online Library (wileyonlinelibrary.com), DOI: 10.1002/spe.2127.
[6] S. R. Chidamber and C. F. Kemerer, "A Metrics suite for object Oriented Design", IEEE Transactions on Software Engineering, vol. 20, no. 6, (**1994**), pp. 476-493.
[7] H. S. Chae, Y. R. Kwon and D. H. Bae, "A cohesion measure for object-oriented classes", Software: Practice and Experience, vol. 30, issue 12, (**2000**) October, pp. 1405–1431.
[8] M. Hitz and B. Montazeri, "Measuring coupling and cohesion in object oriented systems", Proceedings of the International Symposium on Applied Corporate Computing, (**1995**), pp. 25-27.
[9] J. M. Bieman and B. Kang, "Cohesion and reuse in an object-oriented system", Proceedings of the 1995 Symposium on Software reusability, Seattle, Washington, United States, (**1995**), pp. 259-262.
[10] L. Badri and M. Badri, "A Proposal of a new class cohesion criterion: an empirical study", Journal of Object Technology, vol. 3, no. 4, (**2004**).
[11] L. C. Briand, J. Daly and J. Wuest, "A unified framework for cohesion measurement in object-oriented systems", Empirical Software Engineering - An International Journal, vol. 3, no. 1, (**1998**), pp. 65- 117.
[12] L. Fernandez, and R. Pena, "A sensitive metric of class cohesion", International Journal of Information Theories and Applications, vol. 13, no. 1, (**2006**), pp. 82-91.
[13] C. Bonja and E. Kidanmariam, "Metrics for class cohesion and similarity between methods", Proceedings of the 44th Annual ACM Southeast Regional Conference, Melbourne, Florida, (**2006**), pp. 91-95.
[14] J. Bansiya, L. Etzkorn, C. Davis and W. Li, "A class cohesion metric for object-oriented designs", Journal of Object-Oriented Program, vol. 11, no. 8, (**1999**), pp. 47-52.
[15] S. Counsell, S. Swift and J. Crampton, "The interpretation and utility of three cohesion metrics for object-oriented design", ACMTransactions on Software Engineering and Methodology (TOSEM), vol. 15, no. 2, (**2006**), pp.123-149.
[16] H. S. Chae, Y. R. Kwon and D. Bae, "A cohesion measure for object oriented classes", Software—Practice & Experience, vol. 30, no. 12, (**2000**), pp.1405-1431.
[17] D. C. Rine and N. Nada, "Three empirical studies of software reuse reference model", Software: Practice and Experience, vol. 30, issue 6, (**2000**) May, pp. 685–722.
[18] I. Neamtiu, G. Xie and J. Chen, "Towards a better understanding of software evolution: an empirical study on open-source software", Journal of Software: Evolution and Process, vol. 25, Issue 3, (**2013**) March, pp. 193–218.
[19] W. Wieczerzycki, "Software Reusability through Versions, Software: Practice and Experience, vol. 26, Issue 8, (**1996**) August, pp. 911–927.

[20] W. Li, "An Empirical Study of Software Reuse in Reconstructive Maintenance", Journal of Software Maintenance: Research and Practice, vol. 9, Issue 2, (**1997**) March, pp. 69–83.
[21] M. Kiewkanya and P. Muenchaisri, "Measuring maintainability in early phase using aesthetic metrics", Proceedings of the 4th WSEAS International Conference on Software Engineering, Parallel & Distributed Systems, (**2005**) February 13-15, Salzburg, Austria, pp. 1-6.
[22] S. Mal and K. Rajnish, "Theoretical validation of New Class Cohesion Metric against Briand Properties", Proceedings of the International Conference on Advanced Computing, Networking, and Informatics, India, (ICACNI-**2013**), Advances in Intelligent and soft Computing, Springer, vol. 243, **(2013)** June, pp. 591-598.
[23] http://www.sourceforge.net/projects/ (Last accessed 20th November, **2013**).
[24] S. Mal and K. Rajnish, "New Quality Inheritance Metrics for Object-Oriented Design" International Journal of Software Engineering and Its Applications, vol. 7, no. 6, (**2013**), pp. 185-200. http://dx.doi.org/10.14257/ijseia.2013.7.6.16.

Authors

Sandip Mal, he completed B.Tech from West Bengal University of Technology, India in the year 2008. He has also completed ME (Software Engineering) from Birla Institute of Technology, Mesra, Ranchi, Jharkhand, India in the year 2012. Currently, he is pursuing Ph. D. on Software Quality Metrics. His Research area is Object-Oriented Metrics, Software Engineering, Database System, and Image Processing.

Kumar Rajnish, he is an Assistant Professor in the Department of Information Technology at Birla Institute of Technology, Mesra, Ranchi, Jharkahnd, India. He received his PhD in Engineering from BIT Mesra, Ranchi, Jharkhand, India in the year of 2009. He has 24 International and National Research Publications. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming Languages, and Database System.