IJCTA, 9(10), 2016, pp. 4365-4375

Cognitive Weighted Cohension Complexity Metric

***T. Francis Thamburaj**

Abstract : Cohesion is considered as one of most important object-oriented software attributes to improve the quality of software. It is defined as the degree of the relatedness of the members in a class. Several positive quality attributes like comprehensibility, reliability, reusability, testability, robustness, portability, and maintainability are associated with high cohesion. Among the large number of proposed cohesion metrics, LCOM metrics are widely popular.In the family of LCOM metrics, LCOM5 is highly used to measure the cohesioncomplexity of a class by by the relationship among instance variablesand methods. This paper proposes a new Cognitive Weighted Cohesion (CWC), based on the flawless LCOM5, called Coh metric of Briand. Unlike Coh metric, CWC metric includes both architectural and cognitive complexities on the ground of seven degrees of cohesion measures. The proposed metric is formulated and seven cognitive weights are calibrated based on 63 empirical studies with 120 persons. The experimentation and the case studies proved its applicability. It is also validated with comparative study.The correlation test has substantiated that CWC complexity metric is a valid, better, more comprehensive, more realistic, and more robust indicator of software complexity thanthe existing Coh metric.

Keywords : Cognitive Cohesion Metrics, Cognitive Metrics, Cohesion Metrics, Software Complexity Metrics, Software Engineering.

1. INTRODUCTION

Cohesiveness reduces complexity. One of the primarypurposes behind the Object Oriented analysis and design is to develop a highly cohesive classes and software systems[1]. High cohesion isdesired by the developers, because several positive quality attributes like comprehensibility, reliability, readability, maintainability, modifiability, reusability, testability, robustness, and portability, are associated with it [2][3]. Therefore, there is a constant search for better, more comprehensive, more accurate, more reliable and easy to compute cohesion complexity metric in the software industry.

The term cohesion was first introduced by Larry Constantine in the late 60s as part of structured design[4]. In their definition, the measurement of module cohesion is done by counting the unique number of connections between all attribute pairs and method pairs. It points to the amount of relatedness among the components of the module or class. A highly cohesive class or module is well defined to do a single task in such a way that the class can't be split into more than one class or module [5]. The cohesion metrics can be divided into three categories: structural cohesion, internal or syntactic cohesion, and external or semantic cohesion. This paper deals with the structural cohesion, in the class level.

A. Degrees of Cohesion

Class level cohesion denotesthe amount of relationship present among the attributes and methods of that class.Based on the ordinal scale, the cohesion can be classified into seven categories. They are coincidental cohesion,

* A. Aloysius, Department of Computer Science, St. Joseph's College (Autonomous), Bharathidasan University, Thiruchchiraappalli, Tamil Nadu, francisthamburaj@gmail.com, aloysius1972@gmail.com.

logical cohesion, temporal cohesion, procedural cohesion, communicational cohesion, sequential cohesion, and functional cohesion [6 Ch. 7]. Based on the definition by Myers, they are arranged from weakest to strongest cohesion. In the coincidental cohesion, the elements of a class have nothing in common besides being within the same class. This is the weakest form of cohesion. In the logical cohesion, theelements with similar functionality are collected in one class. In the temporal cohesion,the elements of a class have logical cohesion and are performed at the same time. In the procedural cohesion,the elements of a class are connected by some control-flow. In the communicational cohesion,the elements of a class are connected by some control flow and operate on the same set of data. In the sequential cohesion,the elements of a class have communicational cohesion and are connected by a sequential control flow. In functional cohesion, the elements of a class have sequential cohesion, and all the elements contribute to a well-defined single task [6].

B. Properties of Cohesion Complexity Metric

Briand et al. defined four properties for the good cohesion complexity metric. These properties are commonly used to validate the cohesion metrics [7]. When any one of these properties is not satisfied by a metric, it is not considered to be well-defined. The first property states that the cohesion complexity metric values must be within a specific positive range. This type of normalization helps to compare with other cohesion metrics. The second property asserts that the value of the cohesion complexity metric should be zero, when there is no cohesive interactions in the class. Conversely, the metric value will be maximum, when all possible interactions are present in the class. The third property, called 'monotonicity', holds that the cohesion of a class cannot be decreased by adding more cohesive interactions to thatclass. The fourth property, called 'cohesive modules', states that the cohesion of a class cannot be increased by way of merging the two unrelated classes into a singleclass.

2. SURVEY OF LITERATURE

A. Lack of Cohesion Metrics

Among the large number of proposed cohesion complexity metrics, LCOM (Lack of Cohesion in Methods) metrics are widely popular and highly researched. Chidamber and Kemerer originally proposed LCOM 1 in 1991 [8] and defined it as the number of method pairs that do not share attributes. Since LCOM1 was ill defined, they proposed LCOM2 by modifying the LCOM1 and defined it as $|P| - |Q|$where P is the count of method pairs without common attribute references and Q is the number of similar method pairs. However, if $|P|<|Q|$, LCOM is set to zero [9]. LCOM3 was proposed by Li and Henry in 1993 [10]. It is a graph-oriented method. In this graph, each method is represented as a node and each shared attributeis representedas an edge. Now, the LCOM3 is nothing but the count of the connected components.Hitz and Montazeri [11] proposed LCOM4. It follows the same graphical method asthat of LCOM3. The difference is that the additional edges are used to represent the method invocation. LCOM4 is the number of connected components in a class, where method A and method B is connected if they share an instance variable or either method A invokes method B or vice versa. Henderson-Sellers proposed LCOM5 in 1996. LCOM5 = $(a - kl) / (l - kl)$ Where, 'l' is the number of attributes, 'k' is the number of methods, and 'a' is the sum total of the distinct attributes that are used by each method in thegiven class[12].

B. Other Cohesion Metrics

In 1995 Bieman and Kang introduced the tightly coupled cohesion (TCC) and loosely coupled cohesion (LCC) [5]. In 1998, Briand et al introduced the Coh metric by removing the pitfalls of LCOM5. It computes cohesion as the ratio of the number of distinct attributes accessed in methods of a class [7]. Cohesion Based on Member Connectivity (CBMC) metric was proposed by Chae et al. in 2000. Zhou et al. [13] introduced Improved Cohesion Based on Member Connectivity (ICBMC), an enhanced version of CBMC. Badri et al. [14] proposed connectivity based metrics, namely, Degree of Direct Cohesion (DC_D) and Degree of Indirect Cohesion (DC_I). These are similar to TCC and LCC, respectively.The difference is that they also consider the connected methods, when both of them invoke the same method directly or transitively.

Bonjaformulated the Class Cohesion (CC) metricas the division of the total number of similar method pairs by the total number of all method pairs [15].Fernandez et al. [16] proposed Sensitive Class Cohesion Metric (SCOM) in 2006. According to them,it is the percentage of similar pairs of methods in all classes of the given software system.Here, the similarity between the methods i and j is the division of ($|l_i 3 l_j| |l_i 4 l_j|$) by minimum of ($|l_i|, |l_j|$).In 2011Jehad Al Dallal proposed the Transitive-based object-oriented lack-of-cohesion metric [17].In 2013,Amendeep et al. classified the class cohesion metrics based on the interactions among method-method, method-attribute, and attribute-attribute [18].In 2014, K.P. Srinivasan et al.proposed Lack-of-Cohesion Factor (LCF) [19].Reviews and evaluations of various cohesion metrics can be found in [1-3],[20-22]. The scope of this article is class level cohesion.

All the class level cohesion metrics proposed till date, is based on the architectural complexity only and they do not deal with the cognitive aspect of the cohesion complexity. “Such kind of traditional measurements”, as Wang observes, “cannot truly reflect the real complexity of the software system in any step of the software life cycle, such as design, representation, cognition, comprehension, modification, maintenance andtesting.The ideal measure is the cognitive complexity measure that combines both the structural and operational complexity.”[23]Thiscalls for proposing new cognitive based metric for traditional cohesion metric.

The following section 3 defines and explains the proposed CWC metric. The section 4 portrays the calibration of the cognitive weights. The section 5 deals with the experimentation and the case studies of the new complexity metric. The section 6 does the comparative study of CWC with Coh and the correlation analysis. The section 7 presents the conclusion and the possible future works.

Table 1. Calibration of Cognitive Weights

Category	*Program #*	*CMT (Secs)*	*ACMT (Secs)*	*CW*
CoincidentalCohesion(IC)	P1	530.875	512.161	5
	P2	488.333		
	P3	517.275		
LogicalCohesion(LC)	P4	378.500	387.969	4
	P5	362.625		
	P6	422.782		
Temporal Cohesion(TC)	P7	322.042	324.445	3
	P8	307.500		
	P9	343.792		
Procedural Cohesion(PC)	P10	280.625	282.061	3
	P11	266.125		
	P12	299.432		
Communicational Cohesion(CC)	P13	230.583	235.694	2
	P14	255.500		
	P15	221.000		
Sequential Cohesion(SC)	P16	222.792	211.170	2
	P17	212.051		
	P18	198.666		
Functional Cohesion(FC)	P19	120.875	117.083	1
	P20	98.500		
	P21	131.875		

3. COGNITIVE WEIGHTED COHESION COMPLEXITY METRIC

The proposed Cognitive Weighted Cohesion complexity metric is based on the LCOM5, which is more popular and widely used. The reason behind this, is the simplicity and easiness to compute and get automatized. Further, the necessary data such as the number of attributes, methods, and the interaction among them can be captured early in the software life cycle. This helps for early detection of problems and errors, resulting in less cost, more accurate resource allocation, and better quality monitoring from the early phases of the software life-cycle. Above all, it has high discriminative power [24]. The Henderson formula for LCOM5 is given as follows:

$$\text{LCOM5} = \frac{(a - kl)}{(l - kl)} \tag{1}$$

where, l = total number of attributes; k = total number of methods;and a =sum total of the distinct attributes that are used by each method in thegiven class.

Before formulating the CWC, we have to clear some of the basic and important lacunas of LCOM5 by Henderson-Sellers. The blatant lacuna discovered by Briand et al. is that though Henderson proposed the fixed range of value [0, 1] for the cohesion, it crosses the maximum value for some cases, and rides on up to 2 [7]. This violates the first condition of Briand et al. Also, the fourth condition is not satisfied [25]. Therefore, it is not a valid cohesion metric. Briand et al. rectified these lacunas and provided the new formula for LCOM5, called Coh. The relationship between LCOM5 and Coh is given below:

$$\text{Coh} = 1 - \left(1 - \frac{1}{k}\right)\text{LCOM5} \tag{2}$$

where k is the total number of methods in the given class.

We can simplify the Eq. (2) by using the Eq. (1). This is equal to the following formula,

$$\text{Coh} = \frac{a}{kl} \tag{3}$$

where the variables l, k, and a are same as in Eq. (1).

The proposed Cognitive Weighted Cohesion complexity metric is based on Eq. (3) as given below.

$$\text{CWCoh} = \left(\frac{a}{kl}\right)\text{CW}_{\text{degree}} \tag{4}$$

where the variables l, k, and a are same as in the Eq. (1).

The value of $\text{CW}_{\text{degree}}$ can be any one of the seven values CW_{IC} the coincidental cohesion, CW_{LC} the logical cohesion, CW_{TC} the temporal cohesion, $\text{CW}_{\text{PC}}$the procedural cohesion, CW_{CC} the communicational cohesion, CW_{SC} the sequential cohesion, and CW_{FC} the functional cohesion. These seven values are the cognitive weight values for the classical seven degrees of cohesion [6Ch. 7]. The choice of the value depends upon the nature of cohesiveness present in the program.

4. CALIBRATION OF COGNITIVE WEIGHTS

The Cognitive Weights (CW) for seven different degrees of cohesion are calibrated in this section. To find the cognitive weight factor for each of the seven degrees of cohesion, comprehension tests were conducted for three different groups of students to find out the time and effort taken to understand the complexity of different types of cohesion. The students were given sufficient exposure to Java programming and especially in understanding various degrees of cohesion within a class.

Around 40 students, who have scored 65% and above marks in Semester examination, were selected in each group. Two postgraduate groups and one undergraduate group are called for the comprehension test. They were supplied 21 different programs P1 to P21 with multiple choice questions, three for each degree of cohesion. The time taken by each student to understand the program and to choose the best answer was recorded after the completion of each program. This process was repeated for each group of students.

These program comprehension tests were conducted online in order to be accurate and the comprehension timings were registered automatically by the computer in seconds.The average time taken to comprehend each individual program from P1 to P21 by each group was calculated, so as to get 63 different Comprehension Mean Times (CMT).Since3 different groups of students have done the comprehension test for the same program, their values are averaged to obtain the 21 different CMT values. These values are tabulated in Table I, under the column CMT. The tested programs are grouped into seven degrees of cohesion programs as shown in the category column. The corresponding Average Comprehension Mean Time (ACMT) values are calculated and placed in the ACMT column of the Table I.

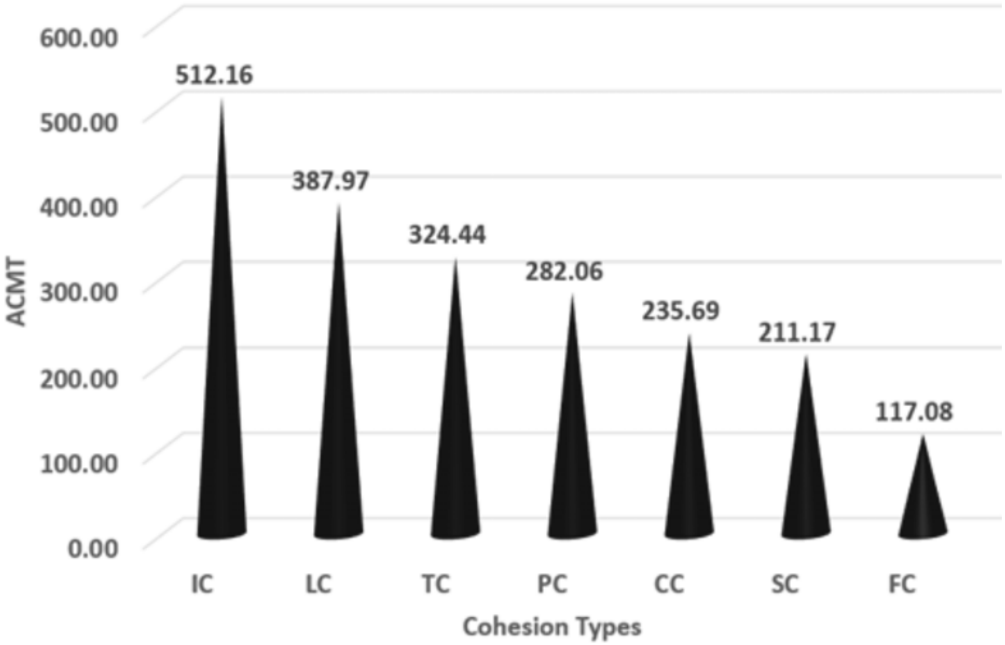

Fig. 1. Average Comprehension Mean Times of seven degrees of cohesion obtained empirically.

The Fig. 1 is the pictorial representation of Table I. The average comprehension mean time for seven different degrees of cohesion is shown in the picture. The CW steadily decreases from weakest to strongest cohesion. The CW refers to the amount of time and effort needed to understand. The CW decreases towards functional cohesion, shows the amount of quality increase in terms of program comprehensibility that helps in various stages of software development life-cycle like designing, testing, modifying, reusing and maintaining. High degree of quality is achieved with the functional cohesion and the lowest quality with the coincidental cohesion. The final cognitive weights are obtained by the MOD 100 function and the CW value for IC is 5, LC is 4, TC is 3, PC is 3, CC is 2, SC is 2 and FC is 1. By normalizing these values, we get IC = 1, LC = 0.75, TC = 0.5, PC = 0.5, CC = 0.25, SC = 0.25 and FC = 0. These values represent the difficulty of understanding various cohesion types. So, the values of understandability are IC = 0, LC = 0.25, TC = 0.5, PC = 0.5, CC = 0.75, SC = 0.75 and FC = 1.

Thus the calibration of difficulty in understanding the different degrees of cohesion in the program has brought out distinct index as the cognitive metric value. The calibration is done by measuring the time and effort needed to comprehend the program as per Wang's methodology [23]. The ratio of these cognitive weights correspond to our natural intuitive understanding of difficulties and hence more meaningful and truthful [26].

5. EXPERIMENTATION AND CASE STUDY

The proposed CWC complexity metric given by Eq. (4) is evaluated with the following four case study programs. These experimentation checks the applicability of the metric. Also, it checks whether the metric value is within the fixed range.

```
/***Coincidental Cohesion – A Case Study ***/
classMyUtil{
publicint factorial(int facto) {
if (facto <= 1) return 1;
elsereturn facto * factorial(facto – 1);
   }
public void CtoF(float fahrenheit){
System.out.print(((fahrenheit - 32)*5)/9);
}
public void MtoK(double miles){
System.out.println(miles * 1.609344);
}
public String isPrime(int prime) {
String msg = ”PN”;
for(inti = 2; i< = prime/2; i++) {
if(prime % i == 0) { msg = “NPN”; }
returnmsg;
}
}
```

The first program given above studies the lower boundary value for cohesion. For this program the Coh and CWC metric values are calculated. The number of attributes l = 0. The number of methods $k = 4$. The value for a = 0, since no method is accessing any attribute. The CW_{degree} is the coincidental cohrsion, $CW_{IC} = 0$. Therefore, Coh = $a/kl = 0/4*0 = 0$ and CWC = 0*0 = 0.

```
/*** Logical Cohesion – A Case Study ***/
class Area{
private double length, breadth, depth;
doubleareaSquare(double d1){
this.length = d1;
return length*length;
}
doubleareaRect(double d1, double d2){
this.length = d1;  this.breadth=d2;
return length*breadth;
}
doubleareaSolid(double d1, double d2, double d3){
this.length = d1;  this.breadth=d2;  this.depth=d3;
return length*breadth*depth;
}
doubleareaCube(double d1){
this.length = d1;
return length*length*length;
}
}
```

The second program given above tests the logical cohesion. In this program the values for l = 3 and k =4. The value of a = 1+2+3+1 = 7.Coh = a/kl = 7 / 3*4 = 0.583 and CWCoh = Coh*CW_{LC} = 0.583 * 0.25 = 0.146.

```
/*** Sequential Cohesion – A Case Study ***/
classMindReader{
privateintnum=0, result=0;
voidthink_A_Number(inti){
num = i;
doubleNum(num);
}
voiddoubleNum(inti){
result= i*2;
if(result>num)    addNine(result);
}
voidaddNine(inti){
result = i+9;
if(result>num)    subThree(result);
}
voidsubThree(inti){
result = i-3;
if(result>num)    divideByTwo(result);
}
voiddivideByTwo(inti){
result = i/2;
if(result>num)    subNum(result);
}
voidsubNum(inti){
result = i - num;
System.out.print(result+"");
}
}
```

The sequential cohesion is experimented in the third case study program given above. In this program the value of l = 2, k = 6, and a = 11. The value of Coh = a / kl = 11/12 = 0.917 and the value of CWC = 0.917 * 0.75 = 0.68775.

```
/*** Functional Cohesion – A Case Study ***/
class Interests {
private float P, N, R;
floatintIOB(float p, float n){
this.P=p;   this.N=n; this.R=5;
return (P*N*R)/100;
}
floatint BOI(float p, float n){
```

```
this.P = p;  this.N = n; this.R = 6;
return (P*N*R)/100;
}
voidintSBI(float p, float n){
this.P = p;  this.N = n; this.R = 7;
System.out.print((P*N*R)/100+"");
}
}
```

The fourth case study program given above, experiments the upper boundary with the functional cohesion program. In this program, the number of attributes l = 3. The number of methods k = 3. The value for a = 3 + 3 + 3 = 9, since each method is accessing all the three attributes.The CW_{degree} is CW_{FC} = 1, since the program has functional cohesion.

Therefore, Coh = a/kl = 9/3*3 = 1 and CWC = 1*l = 1.

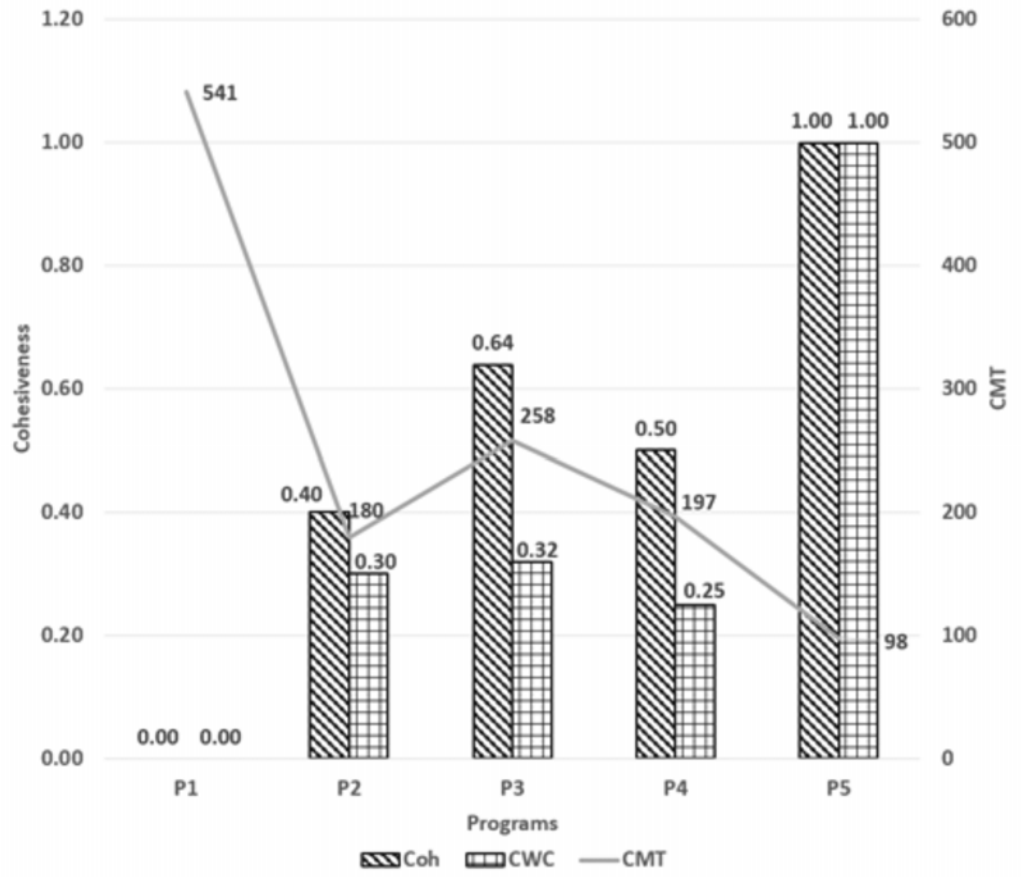

Fig. 2. CWC, Cohmetric values Vs Comprehension Mean Time.

Thus the four case studies prove the applicability of the newlyproposed and defined CWC metric. Here the complexity value of CWC is less than the complexity value of Coh, because CWC pulls down the cohesion value, due to the combined complexity of both structural and cognitive aspects of the program. Though CWC complexity has decreased, the range of complexity value is fixed as that of Coh. That is, from zero to one and hence it is dimensionless which is one of the seven criteria of Abreu et al for robust object-oriented metric [27].Also, this satisfies the first condition of Briand et al [7]. and the other three conditions are also satisfied since Coh metric satisfies them [25]. Therefore, it is a valid cohesion metric.

6. COMPARATIVE STUDY& CORRELATION ANALYSIS

Table 2. Comparative Study of Coh and CWC

Program #	*Coh*	*CWC*	*CMT*
P1	0.0	0.0	541
P2	0.40	0.30	180
P3	0.64	0.32	258
P4	0.50	0.25	197
P5	1.0	1.0	98

Coh and CWC are the calculated complexity metric values. CMT is the Comprehension Mean Times obtained empirically.

In this section, the comparative study and correlation analysis are done to validate the CWC complexity metric, as it is done in other cases of newly proposed complexity metric [28]. The comparative study is performed against the Cohcomplexity metric from which the metric is derived. For this study, a comprehension test was conducted to a group of students who are doing their master's degree. There were forty students, secured more than 65 marks, in the group. The students were given five different programs, P1 to P5, in Java. The time taken to complete the test in seconds is captured in the online style, in order to maintain the accuracy. The average time taken to comprehend each test program by all the students is calculated and placed in Table II under the column head CMT. The Coh and CWCcomplexity values are calculated manually for each of the five programs as demonstrated in the case study section of this article. Their values are also tabulated under the column Coh and CWC.All the values of CWC is less than the Coh values, since CWC is augmented with cognitive factors that pull down the values of complexity metric.

Based on the values of Table II, Pearson Correlation test was conducted between the Cohcomplexity metric values and CMT values. The correlation value r(Coh, CMT) is – 0.8640. The Pearson Correlation with CWC complexity metric values and CMT values was calculated and the value *r*(CWC, CMT) is – 0.7608. Here, we observe that both the correlation values are strongly negative, which means that high cohesion scores low CMT values, low cohesion scores high CMT values and vice versa. The high correlation values imply that both Coh and CWC complexity metrics correlate well with CMT values captured in the empirical test conducted. This shows that the CMT values are truthful and meaningful. The smaller correlation value for CWC complexity metric than the Cohcomplexity metric concludes that CWC complexity metric is a better indicator of complexity of the classes.

This fact is further clarified clearly in the correlation chart given in Fig. 2. Due to the addition of cognitive weight factor, the CWCcomplexity values are less than the Cohcomplexity values in all cases except the boundaries, satisfying the first axiom of valid metric of Briand et al. [7] The captured CMT values are also generally tend towards the CWC complexity metric values than to the Coh complexity metric values. Thus, theproposed CWC complexity metric, which considers not only the architectural complexity but also the cognitive complexity, is proved to be comparatively better, statistically robust and more realistic complexity metric than Coh metric.

7. CONCLUSION

In this article, a new complexity metric called Cognitive Weighted Cohesion has been proposed and mathematically defined for measuring the class level complexity. Unlike the Coh metric, the CWC complexity metric captures not only the structural complexity but also the cognitive complexity that arises due to time and effort needed to comprehend the classes in the software system. The cognitive weights are calibrated using series of comprehension tests and found that the cognitive load for different degrees of cohesion differ. The proposed CWC complexity metric is more comprehensive in nature, truer to reality, and valid by satisfying Briand's all conditions for cohesion metric. It is proved empirically by conducting a set of comprehension tests. Further, the applicability

of the complexity metric is verified by experimentation and case studies. Added to that, it is confirmed statistically, by performing Pearson correlation analysis that concluded saying that CWC complexity metric is a better and more accurate indicator of class cohesion complexity than the existing Coh complexity metric.

Regarding the future works, the CWC complexity metric can be applied and studied for the other object-oriented languages such as C++, C#, ADA etc. Further, the empirical studies can be done with software industry groups, instead of the student groups. A tool can be developed for calculating the CWC complexity metric values to compare it with other related cohesion complexity metrics.Also, large scale empirical analysis with the open source software can be done.

REFERENCES

1. AnkitGarg, "A Review on Coupling and Cohesion Metrics," Special Issue on Int. Journal of Recent Advances in Engineering & Technology (IJRAET), For National Conference on Recent Innovations in Science, Technology & Management (NCRISTM) ISSN (Online): vol. 4, no. I-1, pp. 2347-2812,Feb. 2016.
2. A. Aloysius, "A Cognitive Complexity Metrics Suite for Object Oriented Design," PhD Thesis, Bharathidasan University, Trichy, India, 2012.
3. S.P. Sreeja, M. Binoj, S.N. Kavitha, "A Review on Determining Cohesion and Coupling Based Object Oriented Metrics," International Journal of Engineering Research and Development e-ISSN: 2278-067X, p-ISSN: 2278-800X, vol. 12, no. 1, pp. 74-79, January 2016.
4. W.P. Stevens, G.J. Myers, and L. L. Constantine, "Structured design," IBM Systems Journal, vol. 13, no. 2, pp. 115-139, 1974.
5. J.M. Bieman, B.K. Kang, "Cohesion and Reuse in an Object-Oriented System," Proc. Symposium on Software Reusability, Seattle, Washington, United States, pp.259–262, 1995.
6. E. Yourdon, Larry Constantine, "Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design," Prentice-Hall, Yourdon Press, ISBN 0-13-854471-9, 1979.
7. L.C Briand, J. Daly, and Wusr J., "A Unified Framework for Cohesion Measurement in Object-oriented Systems," Empirical Software Engineering, vol.3, no.1, pp.67-117, 1998.
8. S.R Chidamber, C.F Kemerer, "Towards a Metrics Suite for Object-Oriented Design," Object-Oriented Programming Systems, Languages and Applications (OOPSLA), vol. 26, pp.197–211, 1991.
9. S.R Chidamber, C.F Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Tr. on Software Engineering, vol.20, pp.476–493, 1994.
10. W. Li., S.M. Henry, "Maintenance Metrics for the Object Oriented Paradigm," Proc. 1st Int. Software Metrics Symposium, pp.52–60, 1993.
11. M. Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object Oriented Systems," Proc. Int. Symposium on Applied Corporate Computing, pp. 25–27, 1995.
12. B. Henderon-Sellers,"Object-Oriented Metrics: Measures of Complexity," Upper Saddle River, New Jersey, Prentice Hall, 1996.
13. Y. Zhou, B. Xu, J. Zhao, H. Yang, "ICBMC: An Improved Cohesion Measure for Classes," In Proceedings of International Conference on Software Maintenance, Montreal, Canada, pp. 44-53, 2002.
14. M. Badri, L. Badri, "A Proposal of a New Class Cohesion Criterion: An Empirical Study," Jr. of Object Technology, vol. 3, pp.145-159, 2004.
15. C. Bonja, et al., "Metrics for Class Cohesion and Similarity between Methods," Proc. 44th Annual ACM Southeast Regional Conference, Melbourne, pp.91–95, 2006.
16. L. Fernández, R. Pena, "A Sensitive Metric of Class Cohesion," Int. Jr. of Information Theories and Applications, vol.13, pp.82–91, 2006.
17. J.A. Dallal, "Transitive-based Object Oriented Lack-of-cohesion Metric,"ProcediaComputer Science,vol. 3, pp. 1581-1587, 2011.

18. Kaur Amandeep, and Kaur Puneet Jai, "Class Cohesion Metrics in Object Oriented Systems," International Journal of Software and Web Sciences (IJSWS), vol. 3, no. 2, pp.78-82, 2013.

19. K.P. Srinivasan, T. Devi, "A Complete and Comprehensive Metrics Suite for Object-Oriented Design Quality Assessment," Int.Jr of Software Engineering and Its Applications, vol. 8, no. 2, pp. 173-188, 2014.

20. S. A. Ebad, Moataz A. Ahmed, "Review and Evaluation of Cohesion and Coupling Metrics at Package and Subsystem Level," Journal of Software vol. 11, no. 6, pp. 598-605, 2016.

21. N.Gehlot and RituSindhu, "A Class Cohesion Measure for Evaluation of Resuablity," World Engineering & Applied Sciences Journal, ISSN 2079-2204, vol. 6, no. 2, pp. 104-108, 2015.

22. T. Kaur and Rupinder Kaur, "Comparison of Various Lacks of Cohesion Metrics," International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958, vol. 2, no. 3, February 2013.

23. Y. Wang, and J. Shao, "Measurement of the Cognitive Functional Complexity of Software," Proc. Second IEEE Int. Conf. Cognitive Informatics (ICCI'03), pp. 1-6, 2003.

24. Al DallalJehad, "Measuring the discriminative power of object-oriented class cohesion metrics," Software Engineering, IEEE Transactions on, vol. 37, no. 6, pp. 788-804, 2011.

25. S. Mal and K. Rajnish, "New Class Cohesion Metric: An Empirical View," International Journal of Multimedia and Ubiquitous Engineering, vol. 9, no. 6, pp. 367-376, 2014.

26. N.E. Fenton, and J. Bieman, "Software Metrics: A Rigorous and Practical Approach," 3rd edition, CRC Press, ISBN: 9781439838228, Nov. 2014.

27. F. B. Abreu et al., "Object-oriented software engineering: Measuring and controlling the development process," Proc. 4th int. Conf. software quality, vol. 186, pp. 1-8, 1994.

28. FrancisThamburaj, A. Aloysius, "Cognitive Weighted Polymorphism Factor: A Comprehension Augmented Complexity Metric," World Academy of Science, Engineering and Technology, Indexed in ISI, SCOPUS Computer and Information Engineering, ISSN 1307-6892, vol. 2, no. 11, pp. 1604-1609, 2015.