Sensitivity and Monotonicity in Class Cohesion Metrics

Deepika Kansal, Tejashree Aher, Rushikesh K. Joshi
Department of Computer Science & Engineering, IIT Bombay
Mumbai, India
deepika@indianoil.in,tejashree.aher@iitb.ac.in,rkj@iitb.ac.in

ABSTRACT

Cohesion metrics have been explored widely in the literature for procedural programs and subsequently for object oriented programs. However, as it is commonly found with metrics based approaches, most attempts have been shown to have suffered from some or the other flaws. In this paper, in light of two problems associated with cohesion metrics namely *sensitivity* and *monotonicity*, an analysis of a few recent refined metrics is provided. A new metric called ISCOM (Improved Sensitive Cohesion Metric), is proposed. ISCOM addresses these problems capturing the strength of cohesion with a finer granularity and a higher sensitivity. A comparison is also made with the help of cohesion lattice based models to highlight the contributions of the new approach based on sensitivity and monotonicity. We also apply cohesion lattice theory to validate the approach.

CCS CONCEPTS

• **General and reference** → **Measurement**; **Metrics**; • **Software and its engineering** → *Abstraction, modeling and modularity*; *Software maintenance tools.*

KEYWORDS

Object Oriented Design, Cohesion, Classes, Metrics, Sensitivity, Monotonicity

ACM Reference Format:
Deepika Kansal, Tejashree Aher, Rushikesh K. Joshi. 2019. Sensitivity and Monotonicity in Class Cohesion Metrics. In *12th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC'19), February 14–16, 2019, Pune, India.* https://doi.org/10.1145/3299771.3299794

1 INTRODUCTION

Software measurement is an essential approach to quantify the quality of programs. Measurement can further be used as a driver to improve quality [1]. Class Cohesion measures the togetherness of a module's components [2]. More the cohesiveness in a class, lesser is the need of applying refactoring to the class. Less cohesive classes attract various refactoring opportunities to improve the design of the class [3]. The research on metrics is thus not only contributing to the evolution of the metrics themselves, but it has also contributed to refining the methods of measurement in software engineering [4]. In recent years, new proposals on effective cohesion metrics have been sporadic since the research has gradually converged after inception in the approach of the CK suite [5].

A variety of metric oriented approaches for analyzing class cohesion can be found in the literature. Many cohesion measures such as LCOM1 [5], LCOM2 [6], LCOM3 [7], LCOM4 [8], RLCOM [10], CR [11], TCC [12] and LCC [12] formulate their metric values in terms of the cardinality of the set of method pairs sharing attributes. LCOM* [9] takes into account the total number of attributes accessed by each method to compute lack of cohesion in a class. A known drawback of these measures is that the number of attributes shared by method-pairs is not taken into consideration. As a result, in many cases the conclusions do not consider the effect of actual contribution of a method pair towards cohesion. Instead, their focus mainly stays on obtaining a yes/no value for each method-pair reflecting whether there is any sharing within the pair.

Table 1: The Metrics

Terminology:: Let a and m be the number of attributes and methods respectively in a given class where P be the number of method pairs which do not share any attribute and Q be the number of method pairs which share some attribute. Given I_i be the attribute set accessed by the method M_i and A_j is the number of methods that access the attribute j.

Metric	Definition
LCOM1	No. of disconnected method pairs (P)
LCOM2	if(P>Q) {P-Q} else 0
LCOM4	No. of connected components in a given class
RLCOM	if(Q>0) {P/Q} else 0
LCOM*	$\frac{\left[\frac{1}{a}\sum_{j=1}^{a} A_j\right]-m}{1-m}$
TCC	Q/(P+Q)
CC	$\frac{2}{m(m-1)}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\left[\frac{\lvert I_i \cap I_j\rvert}{\lvert I_i \cup I_j\rvert}\right]$
LSCC	$\frac{2}{m(m-1)}\left[\frac{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\lvert I_i \cap I_j\rvert}{a}\right]$
SCOM	$\frac{2}{m(m-1)}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\left[\frac{\lvert I_i \cap I_j\rvert}{min(\lvert I_i\rvert,\lvert I_j\rvert)}\times \frac{\lvert I_i \cup I_j\rvert}{a}\right]$

*1. Deepika Kansal carried out her Masters thesis work at IIT Bombay where this work was done. She is presently with IOCL, Delhi. 2. Tejashree Aher was a Dual Degree student at IIT Bombay where this work was done. She is currently a Big-Data Engineer in IT Industry. 3. Rushikesh K. Joshi is Professor of Computer Science and Engineering at IIT Bombay.

ISEC'19, February 14–16, 2019, Pune, India

ACM ISBN 978-1-4503-6215-3/19/02…$15.00
https://doi.org/10.1145/3299771.3299794

CC [13] computes total sharing of attributes between each pair of methods w.r.t. the size of accesses of that pair, and normalizes it. This improves its discrimination power as compared to earlier metrics. However, we observe that if both methods in a method pair share the same set of attributes, then it considers the contribution of this method pair towards the metric value as fullest regardless of the number of attributes (size) that they share. Due to this reason, it violates the property of monotonicity (*adding an edge does not reduce cohesion*) [16] [21]. (Examples of violation of monotonicity in CC are illustrated in latter discussions through cases iv, vii and x in Table 3) Also, in many other cases, it suffers from insensitivity. We define insensitivity as the *need for addition or removal of a considerable number of edges to change the class cohesion value* (Examples of insensitivity of CC appear later in Table 3 through case pairs ii-iii, v-vi and viii-ix).

LSCC [19] computes the metric value in terms of relative sharing among pairs with respect to maximum possible sharing among pairs, in contrast to CC′s computation wrt to the union of access between the methods in the pair. This improves the monotonicity and accuracy as compared to CC, since it considers the attribute sharing in a class as whole. However, it does not provide any special emphasis on method pairs with more attribute sharing. This results in a loss of sensitivity which still remains an issue as it can be observed from case pairs iii-iv, vi-vii, and ix-x in Table 3, and case pairs C-D and G-H of Table 6.

SCOM [14] on the other hand overcomes this issue by defining a term called *'weight factor* to inject more sensitivity into the metric. It improves the effect of the actual contribution of a method pair, since it consider the weightage of a method pair in accordance with the combined total number of attributes accessed by both the methods. However, another problem arises in SCOM leading to unexpected results, especially when a proper subset-superset relationship exists between the attribute-set of a particular method pair. Dallal [15] discuss this problem with SCOM through several cases in which various other metrics including SCOM fail. Dallal and Briand [19] show through example cases that CC and SCOM both violate the property of monotonicity.

The proposed approach that culminates into a new metric ISCOM rectifies these problems found in the SCOM and other metrics by offering better sensitivity. Programming tools can benefit from the new metric to provide a better view on class cohesion for software designers and developers during both development and refactoring activities. With the proposed metric, we aim to measure class quality more accurately in terms of cohesion, without losing the goodness properties of some of the earlier metrics, such as ease of computation, accuracy, automation. We show through examples that the proposed metric is more sensitive to the small changes made to method-attribute accesses. ISCOM falls under the category of Connection Magnitude-based Metrics [20]. As compared to existing well-known metrics, we show that it uses the range [0...1] more effectively.

SCOM is also compared with a cohesion lattice based approach of Joshi and Joshi [22]. Their Cohesion lattices classify object oriented classes into seven types of cohesion lattice patterns instead of measuring cohesion in terms of numbers. The effectiveness of the proposed metric is illustrated using comparisons with both metric based and cohesion lattice based approaches. We demonstrate the effectiveness of proposed metric with the help of example cases from [20], [14] and [15].

The rest of the paper is organized as follows. The next section discusses sensitivity issues in SCOM. Section 3 formulates the proposed metric. Section 4 discusses our observation on the improvements achieved with the help of example cases.

2 ON THE SENSITIVITY OF SCOM

Fernandez [14] formulated the SCOM metric to bring more sensitivity in the practice of cohesion metrics. They also assess SCOM with respect to various desirable qualitative properties such as simplicity, independence, accuracy, automation, value of implementation, sensitivity, monotonicity, measurement scale, boundary marks and prescription, and then they finally claim that SCOM agrees on all of these properties. But subsequently the deficiencies of SCOM w.r.t. sensitivity and monotonicity were shown by Dallal [15] [19].

In a recent research on the quality analysis of the class cohesion metrics [20], it was shown that SCOM captures the variations in the cohesiveness of a class with a finer granularity as compared to other existing ones. Dallal [15] also concluded in his research work that SCOM attains a stronger discriminative power among other class cohesion metrics selected by them. In this section, a brief background to SCOM is provided highlighting the issue of sensitivity. SCOM introduces a *Weight factor* w.r.t a particular method-pair, which is the ratio of the cardinality of attributes accessed by the method pair and the total number of attributes in the class.

Connection intensity computes the amount of sharing with respect to the cardinality of the smallest method in the pair in terms of attribute coverage. The connection intensity $C_{i,j}$ for a method pair (M_i, M_j) is computed as follows.

$$C_{i,j} = \begin{cases} 0 & \text{if } (I_i \cap I_j) = \phi \\ \frac{card(I_i \cap I_j)}{min(card(I_i), card(I_j))} & \text{otherwise} \end{cases}$$

where I_k is the attribute set accessed by method M_k.

In order to improve the effect of the contribution of a method pair on the overall metric value SCOM, the weight factor $W_{i,j}$ is multiplied by the connection intensity of that pair.

$$SCOM = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{i,j} \times W_{i,j}$$

$$where\ W_{i,j} = \frac{card(I_i \cup I_j)}{a}$$

A major problem in the formulation of $C_{i,j}$ is that when there exists a proper subset-superset relationship between the attribute sets of a given method-pair, i.e. $I_i \subset I_j$, the value of connection intensity $C_{i,j}$ remains fixed at 1 regardless of the increase in the number of shared attributes between them. Due to this problem in $C_{i,j}$ formulation, the conclusions of SCOM lose sensitivity in certain cases. This behaviour is brought out through cases depicted in Table 2. The table shows 12 cases of access matrices with rows as methods, columns as attributes, symbol 'x' showing an attribute access, and symbol 'o' showing no access.

It can be observed that in Table 2, while moving from case I to case XII, attribute accesses are removed one by one and method

Table 2: Insensitivity Issues: Cases and their SCOM Values

I	II	III	IV	V	VI
1	1	1	1	0.33	0.33
VII	VIII	IX	X	XI	XII
0.33	0.33	0	0	0	0

per method. So, there should be a gradual decrease in the cohesion metric value. However, the results of SCOM do not change as per this intuition. SCOM distinguishes these 12 cases with only three different values, which shows its problem of insensitivity clearly. A sudden drop in the SCOM metric value can be found when a proper subset-superset relationship between any method-pair vanishes. While a change is held for sometime, one can observe a sudden drop in the SCOM value, while stepping from case IV to case V and from case VIII to IX in Table 2.

It can be seen that step-wise values are found in this behavior, which is the cause of loss of sensitivity of the object oriented metric. If a method accesses only one attribute and one other method accesses all the attributes in a class then such a method pair makes a significant contribution in overestimating the SCOM value. Sudden change and overestimation of the metric value are the crucial adverse effects for insensitivity.

3 AN IMPROVED SENSITIVE CLASS COHESION METRIC (ISCOM)

In this section, we formulate a new metric called Improved Sensitive Class Cohesion Metric (ISCOM). Let a and m be respectively the cardinalities of the sets of attributes and methods of the class under measurement. Now, we redefine connection intensity of a pair of methods by dividing the number of shared attributes between a given method-pair by the average of the cardinalities of the two attribute sets accessed by the two methods in that pair.

So, the connection intensity $C'_{i,j}$ of a method pair give I_i and I_j to be the individual attribute sets accessed by the two methods m_i and m_j, can be expressed as follows:

$$C'_{i,j} = \begin{cases} 0 & \text{if } (I_i \cap I_j) = \phi \\ \frac{2*card(I_i \cap I_j)}{card(I_i)+card(I_j)} & \text{otherwise} \end{cases}$$

The new metric ISCOM with the above change in SCOM is computed as follows:

$$ISCOM = \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C'_{i,j} \times W_{i,j}$$

$$where\ W_{i,j} = \frac{card(I_i \cup I_j)}{a}$$

Table 3: Cohesion values: accesses removed one by one

	i	ii	iii	iv	v	vi
ISCOM	1	0.9	0.82	0.75	0.65	0.56
SCOM	1	1	0.91	0.75	0.75	0.67
LSCC	1	0.83	0.75	0.75	0.58	0.5
CC	1	0.83	0.83	1	0.78	0.78
	vii	viii	ix	x	xi	xii
ISCOM	0.5	0.39	0.31	0.25	0.083	0
SCOM	0.5	0.5	0.41	0.25	0.083	0
LSCC	0.5	0.33	0.25	0.25	0.083	0
CC	1	0.67	0.67	1	0.33	0

The change made to pairwise connection intensity makes a considerable difference in estimating a class's cohesion value, overcoming the sensitivity issues associated with SCOM. Let us consider the example cases shown in Table 3. In these cases, starting from case i onwards, accesses are removed one by one column-wise (i.e. attribute by attribute) rather than horizontally (function by function) that was done earlier in Table 2.

Even with such an attribute by attribute incremental drop in the cohesion of the class, the new connection intensity designed above is able to detect the drop in the cohesiveness of the class such as in cases iv & v and in cases vii & viii, which SCOM does not detect. However, except for these cases in the table, we can observe that SCOM through its weight factor distinguishes the cases correctly on a decreasing scale of cohesion starting from case i down to case xii. The same weight factor term was used earlier by SCOM.

It can be observed from Table 3 that LSCC and CC both suffer from the issue of sensitivity. Secondly, with the removal of method-attribute access, the cohesion metric value is expected to decrease, indicating monotonicity, which is violated by CC. From Table 4, it can be observed that ISCOM while it fixes the sensitivity issue of SCOM, it also performs comparable to CC and LSCC. In Table 4, the values of ISCOM, CC, SCOM and LSCC are shown for all the cases given in Table 2.

Comparing SCOM and ISCOM over cases shown in Table 2, we see that ISCOM addresses the sensitivity issue effectively. With a systematic drop in connectivity and thereby a drop in cohesion starting from case I to case XII in Table 2, a gradual drop in the ISCOM value due to a drop in the total method-attribute sharing is observed as shown in Figure 1. Clearly, ISCOM overcomes the issues of insensitivity, sudden change and overestimation that was observed with SCOM. In Table 4, ISCOM gives 0 for cases 9 to 12 as there is no method pair that shares an attribute. It can be noted that SCOM and other metrics also detect it as full lack of cohesion.

.

Table 4: Diversity in cohesion measures for cases in Table 2

Case	I	II	III	IV	V	VI	VII	VIII	IX-XII
ISCOM	1	0.90	0.78	0.60	0.33	0.28	0.22	0.13	0
SCOM	1	1	1	1	0.33	0.33	0.33	0.33	0
LSCC	1	0.83	0.67	0.5	0.33	0.25	0.17	0.083	0
CC	1	0.83	0.67	0.5	0.33	0.25	0.17	0.083	0

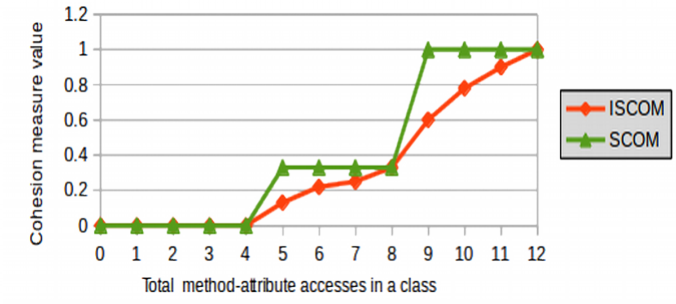

Figure 1: Capturing variations in the cohesion measures for cases shown in Table 2

Thus, ISCOM provides better results during quantitative analysis and deals with all the sensitivity issues effectively as demonstrated through Table 3 and Table 4.

ISCOM holds the various qualitative properties as indicated in [16] [21]. ISCOM is easy to understand and automate as compared to earlier other metrics such as CBMC[17] and ICBMC[18], and primarily, ISCOM is more sensitive to variations in cohesion of the class as compared to SCOM which was shown to be improving upon the earlier existing cohesion metrics. It also gives more accurate result as compared to SCOM and other metrics. We provide a comparison with a few metrics in the following section. ISCOM provides a value in range [0, 1]. Newly added cohesive interactions to the class do not reduce its cohesion value. Value of ISCOM does not change with renaming of any of class, method or attribute. ISCOM is independent of semantics of program, like most other metrics, as its definition depends only on the interaction information between the methods and attributes of a class, and this can be a drawback in certain cases where designer's knowledge is not captured by access matrix, such as in a case when a disconnected method is colocated due to similarity in abstraction but not due to attribute sharing. However, the focus of metrics has been to develop simpler and more accurate way of measuring cohesion, which are also reliable. To further validate the approach, we also used the cohesion lattice approach[22] to show parallels with cohesion lattice structures.

4 ANALYZING REPRESENTATIVE CASES

In this section, we further compare representative diverse cases from the literature as shown in Figure 2. The cases are from [20] (examples A, F, H), from [14] (examples C, D, E) and from [15] (examples I, J), in addition to two new example cases B and G. The cases are used to compare earlier class cohesion metrics, RLCOM, LCOM*, TCC, CC, LSCC and SCOM with ISCOM. Cohesion lattice

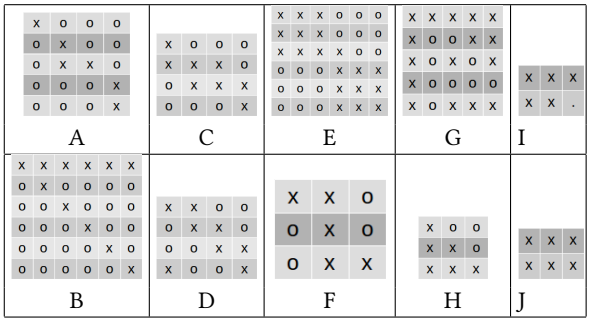

Figure 2: Method/Attribute Access Matrices

patterns for all of cases have been identified for highlighting the variations captured by these cases. The comparison brings out the differences among the discrimination capabilities of the metrics.

As pointed out in [22], Umbrella, Inverted Umbrella and Mountain Range possess moderate cohesion strength, Partition has low cohesion, whereas, Diamond and Chain are high cohesion patterns, Dot being the most cohesive pattern with maximum value of cohesion. The effect of internal structure is also briefly touched upon through these cases. It is shown that the new metric ISCOM does capture the cohesiveness of internal structure of the cohesion lattices effectively and hence a deeper comparison can be made through it. For example, a partition may be seen to have a higher cohesion when the partitions themselves are significant in sizes and are independently cohesive. The cases cover all the seven cohesion lattice patterns described in [22]. Table 5 enlists the patterns for the 10 cases chosen for comparison. Table 6 brings out the metric values for these cases. The cases have been arranged on an increasing scale of ISCOM value. The observations made from these comparisons are brought out below.

.

Case A is a Partition pattern, the least cohesive among all. As it can be seen from Table 6, all metrics except CC capture the partition lattice as the least cohesive case. This is due to the reason that since in case A, method pair (4th and 5th row) shares the same set of attributes (though there is only one member in that set), and CC rates it above B. An interesting observation to note is that, all metrics except SCOM agree on the second least cohesive case for B, an Umbrella pattern. The reason is that many method pairs share proper subset-superset relationship in case B, and SCOM considers these pairs as fully cohesive components. Mountain range is considered to be less cohesive than Diamond and Dot patterns due to absence of a single fully cohesive internal components. However,

Table 5: Lattice Patterns

Cases	Lattice Patterns	Cases	Lattice Patterns
A	Partition	F	Inverted Umbrella
B	Umbrella	G	Diamond
C	Mountain Range	H	Chain
D	Mountain Range	I	Chain
E	Mountain Range	J	Dot

Table 6: Variations in discrimination capability

Case	RLCOM	LCOM*	TCC	CC	LSCC	SCOM	ISCOM
A	4	0.875	0.2	0.15	0.05	0.075	0.058
B	2	0.83	0.33	0.056	0.056	0.33	0.095
C	1	0.67	0.5	0.19	0.17	0.36	0.236
D	0.5	0.67	0.67	0.22	0.17	0.25	0.25
E	0.67	0.57	0.6	0.4	0.23	0.29	0.27
F	0	0.67	1	0.44	0.33	0.61	0.46
G	0	0.45	1	0.5	0.44	0.81	0.55
H	0	0.5	1	0.5	0.44	0.89	0.58
I	0	0.33	1	0.67	0.67	1	0.8
J	0	0	1	1	1	1	1

the actual cohesiveness depends on the deeper internal structure of the class, and the pattern itself is not sufficient to indicate the finer gradations among classes. Cases C, D and E exhibit this structure. No metric except SCOM declares D to be less cohesive as compared to C. Due to overestimation, SCOM assesses case D to be less cohesive. For cases D and E, all metrics except RLCOM and TCC consider case E as more cohesive. Case F exhibits Inverted Umbrella pattern. Interestingly, all metrics except LCOM* agree on case F to be more cohesive as compare to case E, which is a Mountain Range class. In an Inverted umbrella there is at least one attribute which is accessed by all member functions, but there is no function accessing all attributes. We can observe this from the access graph shown in Figure 2. Case F possesses this access pattern, whereas, in case E, no such common attribute or common method exists. LCOM* fails to capture this difference in these two cases.

Case G exhibits the Diamond pattern. ISCOM, SCOM, LSCC, CC and LCOM* capture the higher cohesion of case G as compared to other cases A-F. Cases H-J are more cohesive than case G. Cases H and I exhibit the Chain pattern. ISCOM, LSCC, CC and LCOM* capture the gradation in the cohesiveness for these cases. TCC, RLCOM overestimates these cases to be fully cohesive, while SCOM overestimates case I among these two. Case J is the fully cohesive example case of Dot pattern. ISCOM, LSCC, CC and LCOM* correctly consider only this case as fully cohesive, while other metrics also consider other example cases as fully cohesive, which is an overestimation.

5 CONCLUSION

We identified the problem of sensitivity in measurement of class cohesion. Due to this cohesion metrics fail to capture variations in certain bands of ranges. A recently proven metric SCOM was shown to suffer from the sensitivity issue. Thereupon, a new measure of cohesion called Improved Strength of Cohesion Metric (ISCOM) was introduced to alleviate this problem. ISCOM was shown to be more sensitive and it also preserves monotonicity. The validation was carried out in terms of typical special cases and class cohesion patterns. The metric assesses the closeness of module/class components with a higher discrimination power than its recent predecessors, and at the same time it does not compromise on simplicity, ease of implementation and understandability of its peers.

REFERENCES

[1] N. Fenton, *Software measurement: a necessary scientific basis,* IEEE Transactions on Software Engineering, vol. 20, no. 3, pp. 199-206, March 1994.

[2] J. Bieman and L. Ott, *Measuring Functional Cohesion,* IEEE Transactions on Software Engineering, vol. 20, no. 8, pp. 644-657, August 1994.

[3] M.fowler, *Refactoring: Improving the design of existing code,* Addison Wesley, 1999.

[4] S. Counsell, S. Swift, and J. Crampton, *The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design,* ACM Transactions Software Engineering Methodology (TOSEM), vol. 15, no. 2, pp. 123-149, 2006.

[5] S.R. Chidamber and C.F. Kemerer, *Towards a metric suite for object oriented design,* Object-Oriented Programming Systems, Languages and Applications (OOPSLA), pp. 197-211, 1991.

[6] S.R. Chidamber and C.F. Kemerer, *A Metrics Suite for Object Oriented Design,* IEEE Transactions on Software Engineering, vol. 20, pp. 476-493, June 1994.

[7] W.Li and S.M. Henry, *Measuring object-oriented design,* Journal of Object-Oriented Programming, pp. 48-55, 1995.

[8] M.Hitz and B.Montazeri, *Chidamber & Kemerer's metrics suite: a measurement theory perspective,* IEEE Transactions on Software Engineering, vol. 22, no. 4, pp. 267-271, April 1996.

[9] B.Henderson-Sellers, *Object Oriented Metrics: Measures of Complexity* in New Jersey, Prentice-Hall, pp. 142-147, 1996.

[10] L.Xlinke, L.Zongtian, P.Biao, and X.Dahong, *A measurement tool for object oriented software and measurement experiments with it* in Proceedings of 10th International Workshop on New Approaches in Software measurement, pp. 44-54, 2000.

[11] N. Balasubramaniam, *Object-oriented metrics* in International Proceedings of Asia-Pacific Conference on Software Engineering, pp. 30-34, 1996.

[12] J.M. Bieman and B-K Kang, *Cohesion and reuse in an object-oriented system* in Proceedings of ACM symposium for software reusability, pp. 259-262, 1995.

[13] C. Bonja and E. Kidanmariam, *Metrics for class cohesion and similarity between methods* in Proceedings of the 44th Annual ACM Southeast Regional Conference, Melbourne, Florida, pp. 91-95, 2006.

[14] L.Fernandez and R.Pena, *A sensitive metric for class cohesion* in International Journal on Information Theories and Applications, vol. 13, pp. 82-91, 2006.

[15] J. A. Dallal, *Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics,* IEEE Transactions on Software Engineering, vol. 37, no. 6, pp. 788-804, December, 2011.

[16] E. Weyuker, *Evaluating software complexity measures,* IEEE Transactions on Software Engineering, vol. 14, pp. 1357-1365, 1988.

[17] H.S. Chae, Y.R. Kwon, and D. Bae, *A Cohesion Measure for Object-Oriented Classes,* Software-Practice and Experience, vol. 30, no. 12, pp. 1405-1431, 2000.

[18] Y.Zhou and B.Xu, *ICBMC: An Improved Cohesion Measure for Classes* in Proceedings of International Conference on Software Maintenance, pp. 44-53, 2002.

[19] J. A. Dallal and L. C. Briand, *A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes,* ACM Transactions Software Engineering Methodology (TOSEM), vol. 21, no. 2, pp. 8:1âĂŞ8:34, 2012.

[20] P. Joshi and R.K. Joshi, *Quality analysis of object oriented cohesion metric* in 7th International conference on Quality of Information and Communications Technology (QUATIC), pp. 319-324, 2010.

[21] A. Meneely, B. Smith and L. Williams, *Validating software metrics: a spectrum of philosophies,* ACM Transactions Software Engineering Methodology (TOSEM), vol. 21, no.4, pp. 24:1âĂŞ24:28, 2013.

[22] P. Joshi and R.K. Joshi, *Concept analysis for class cohesion* in 13th European conference on Software Maintenance and Reengineering (CSMR), pp. 237-240, 2009.