Theoretical Validation of Object-Oriented Lack-of-Cohesion Metrics

JEHAD AL DALLAL
Department of Information Science
Kuwait University
P.O. Box 5969, Safat 13060
KUWAIT
jehad@cfw.kuniv.edu

Abstract: - Class cohesion refers to the degree of relatedness of class attributes and methods. Software developers use class cohesion measure to assess the quality of their products and to guide the restructuring of poorly designed classes. Several class cohesion metrics are proposed in the literature, and a few of them are theoretically validated against the class cohesion necessary properties. Metrics that violate class cohesion properties are not well defined, and their utility as indictors of the relatedness of class members is questionable. The purpose of this paper is to theoretically validate six lack-of-cohesion based metrics. Results show that most of the metrics considered satisfy the majority of the class cohesion necessary properties.

Key-Words: - object-oriented class, software quality, class cohesion metric, class cohesion.

1 Introduction

A popular goal of software engineering is to develop the techniques and the tools needed to develop high-quality applications that are more stable and maintainable. In order to assess and improve the quality of an application during the development process, developers and managers use several metrics. These metrics estimate the quality of different software attributes, such as cohesion, coupling, and complexity.

The cohesion of a module refers to the relatedness of the module components. A module that has high cohesion performs one basic function and cannot be split into separate modules easily. Highly cohesive modules are more understandable, modifiable, and maintainable [1].

Since the last decade, object-oriented programming languages, such as C++ and Java, have become widely used in both the software industry and research fields. In an object-oriented paradigm, classes are the basic modules. The members of a class are its attributes and methods. Therefore, class cohesion refers to the relatedness of the class members.

Researchers have introduced several metrics to indicate class cohesion. In order to increase the likelihood that a cohesion metric is well defined and serves as a good indicator for the relatedness of the class members, researchers must validate the metric theoretically and empirically. Briand et al. [2] propose four properties that must be satisfied by all class cohesion metrics. If a metric does not satisfy any of these properties, the usefulness of the metric is questionable and it is ill-defined [2]. These properties provide a supportive underlying theory for the metrics. Empirical validation is necessary to show the usefulness of the metrics. Despite its importance, few researchers focus on the theoretical validation of metrics. In this paper, we theoretically study the validity of six lack-of-cohesion based metrics using the properties introduced by Briand et al. Our results show that none of the metrics satisfy all the properties. However, it is shown that most of the metrics satisfy the majority of the properties.

This paper is organized as follows. Section 2 provides an overview of the class cohesion metrics and necessary properties. In Section 3, the satisfaction of six lack-of-cohesion metrics to the necessary properties is supported or refuted. Finally, Section 4 includes conclusions and a discussion of future work.

2 Related Work

This section overviews the considered lack of cohesion metrics and other class cohesion metrics. In addition, it includes a summary of the class cohesion necessary properties that all class cohesion metrics must satisfy.

2.1 Overview of class cohesion metrics

Researchers have proposed several class cohesion metrics in the literature. These metrics are based on the use or sharing of the class attributes. For example, LCOM1 metric counts the number of method pairs that do not share instance variables [3]. Chidamber and Kemerer [4] propose another version for LCOM metric (LCOM2), which calculates the difference between the number of method pairs that

do and do not share instance variables. Li and Henry [5] use an undirected graph that represents each method as a node and the sharing of at least an instance variable as an edge.

The lack-of-cohesion in methods, LCOM3, is defined as the number of connected components in the graph. The model used in LCOM3 metric is extended in [6] by adding an edge between a pair of methods if one of them invokes the other. Here, we refer to the metric that uses the extended model as LCOM4. Hitz and Montazeri [6] introduce a connectivity metric to apply when the graph has one component. In addition, Henderson-Sellers [7] proposes a lack-of-cohesion in methods metric, LCOM5, that considers the number of methods referencing each attribute.

Bieman and Kang [8] describe two class cohesion metrics, TCC and LCC, to measure the relative number of directly connected pairs of methods and relative number of directly or indirectly connected pairs of methods, respectively. These two metrics consider two methods to be connected if they share at least one instance variable. Badri [9] introduces two class cohesion metrics, DC_D and DC_I, that are similar to TCC and LCC, respectively, but differ by considering two methods connected when one of them invokes the other. Wang et al. [10] introduce a DMC class cohesion metric based on a dependence matrix that represents the dependence degree among the instance variables and methods in a class. Fernandez and Pena [11] propose class cohesion metrics that consider the cardinality of intersection between each pair of methods. In the metric presented by Bonja and Kidanmariam [12], the degree of similarity between methods is used as a basis to measure the class cohesion. The similarity between a pair of methods is defined as the ratio of the number of shared attributes to the number of distinct attributes referenced by both methods. The cohesion is defined as the ratio of the summation of the similarities between all pairs of methods to the total number of possible pairs of methods. Chen et al. [1] use dependence analysis to explore the attribute-attribute, attribute-method, and method-method interactions. The cohesion is measured as the relative number of interactions. Ratio of Cohesive Interactions (RCI) is a metric that considers the data-to-data (DD) and data-to-subroutine (DS) interactions in Ada object-systems [13]. The RCI metric is defined as the ratio of the number of cohesive interactions of a module to the total number of possible cohesive interactions.

Bansiya et al. [14] propose a design-based class cohesion metric called Cohesion Among Methods in a Class (CAMC). In this metric, only the method-method interactions are considered. The CAMC metric uses a parameter occurrence matrix that has a row for each method and a column for each data type that appears at least once as the type of a parameter in at least one method in the class. The value in row *i* and column *j* in the matrix equals 1 when the *i*th method has a parameter of *j*th data type. Otherwise, the value equals 0. The CAMC metric is defined as the ratio of the total number of 1s in the matrix to the total size of the matrix.

Counsell et al. [15] propose a design-based class cohesion metric called Normalized Hamming Distance (NHD). In this metric, only the method-method interactions are considered. The metric uses the same parameter occurrence matrix used by CAMC metric. NHD calculates the average of the parameter agreement between each pair of methods. The parameter agreement between a pair of methods is defined as the number of places in which the parameter occurrence vectors of the two methods are equal.

2.2 Class cohesion metric necessary properties

Briand et al. [2] define four properties for cohesion metrics. The first property, Property 1, called nonnegativity 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 and 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 module's cohesion. Therefore, given two classes, c_1 and c_2, the cohesion of the merged class c' must satisfy the following condition: $\text{cohesion}(c') \leq \max\{\text{cohesion}(c_1), \text{cohesion}(c_2)\}$. Briand et al. [2] disapproved that LCOM1, LCOM2, LCOM4, and LCOM5 satisfy Property 1 and LCOM2 satisfies Property 3. They neither supported nor refuted the other properties.

3 Theoretical Validation

This section studies the theoretical validation of five lack-of-cohesion metrics. The definition of each metric is overviewed and the satisfaction of the metric to the six class cohesion

necessary properties is approved or disapproved.

3.1 LCOM1 [3]

Definition: LCOM1=P, where P is the number of pairs of methods that do not share common attributes.

Property 1 and Property 2: The minimum value for LCOM1 is 0 when each pair of methods shares at least one common attribute (i.e., the model has the maximal number of cohesion interactions). The maximum value for LCOM1 depends on the number of methods in a class. That is, $k(k\text{-}1)/2$, where k is the number of methods, when none of the methods share common attributes (i.e., the model does not have cohesion interactions). Therefore, LCOM1 satisfies Property 2, but it does not satisfy Property 1.

Property 3: Adding a cohesive interaction to the model implies decreasing the number of unrelated pairs of methods, hence decreasing LCOM1 and increasing the cohesion. Therefore, LCOM1 satisfies Property 3.

Property 4: Unrelated classes are classes that have no common attributes and methods. Given that P_C and Q_C are the number of pairs of methods with and without shared attributes in a class C, then for classes A, B, and M, where A and B are unrelated classes and M is their merged class version, $Q_M=Q_A+Q_B$. The LCOM1 of the merged class is calculated as follows:

$$LCOM1(M) = P_M = \frac{(k+m)(k+m-1)}{2} - Q_M$$

$$= \frac{(k+m)(k+m-1)}{2} - [Q_A + Q_B]$$

$$= \frac{(k+m)(k+m-1)}{2} - [(P_A - \frac{k(k-1)}{2}) + (P_B - \frac{m(m-1)}{2})]$$

$$= km + P_A + P_B = km + LCOM1(A) + LCOM1(B)$$

where k and m are the number of methods in classes A and B, respectively. Hence, the cohesion of the merged class is less than the cohesion of each of the split classes, and therefore, LCOM1 metric satisfies Property 4.

3.2 LCOM2 [4]

Definition:

$$LCOM2 = P - Q = P - (NP - P) = 2P - NP$$

$$= 2LCOM1 - NP = 0.5[4LCOM1 - k(k-1)]$$

Property 1 and Property 2: The minimum value for LCOM2 is 0 when $P_c \leq Q_c$. Therefore, when the model has the maximum number of interactions, LCOM2 becomes 0 because, in this case, the number of pairs that do not share common attributes is less than those that share common attributes (i.e., $0<k(k\text{-}1)/2$). However, the maximum value for LCOM2 depends on the number of methods in a class. That is, $k(k\text{-}1)/2$, where k is the number of methods, when none of the methods share common attributes (i.e., the model does not have cohesion interactions). Therefore, LCOM2 satisfies Property 2 and it does not satisfy Property 1.

Property 3: Adding a cohesive interaction to the model implies increasing Q and decreasing P. If in the original model, $P_c \leq Q_c$, the cohesion of the original and the modified models equal to 0. Otherwise, LCOM2 of the model decreases. Therefore, LCOM2 satisfies Property 3.

Property 4: When two unrelated classes are merged,

$$LCOM2(M) = P_M - Q_M = (km + P_A + P_B) - (Q_A + Q_B)$$

$$= km + (P_A - Q_A) + (P_B - Q_B)$$

$$= km + LCOM2(A) + LCOM2(B)$$

Hence, the cohesion of the merged class is less than the cohesion of each of the split classes, and therefore, LCOM2 metric satisfies Property 4.

3.3 LCOM3 [5]

Definition: LCOM3 is defined as the number of connected components in the graph.

Property 1 and Property 2: The minimum value for LCOM3 is 1 when there is a direct or indirect cohesive interaction between each method and another. The maximum value for LCOM3 depends on the number of methods in a class. That is, k, where k is the number of methods, when the model has no interactions. As a result, the value of LCOM3 ranges in the interval [1,k], and therefore, the metric does not satisfy Property 1. In addition, the metric does not satisfy Property 2 because the value of LCOM3 is not equal to 0 when the model has the maximum possible interactions.

Property 3: When adding a cohesive interaction to the model, the number of connected components either decreases by 1 when the interaction connects two disjoint components or remains the same when the interaction does not connect two disjoint components. Therefore, LCOM3 satisfies Property 3.

Property 4: Two unrelated classes are graphically represented by two disjoint graphs. Therefore, when two unrelated classes A and B are merged into class M, the total number of disjoint components increases by 1 (i.e., LCOM3(M)=LCOM3(A)+ LCOM3(B)+1). Hence, LCOM3 metric satisfies Property 4.

3.4 LCOM4 [6]

The only difference between LCOM4 and LCOM3 is in the definition of the cohesive

interactions. The above discussion about the validity of LCOM3 is independent from the definition of the cohesive interactions, and therefore, both metrics have the same properties. However, when the graph is connected, the following connectivity metric is used.

3.5 Connectivity [6]

Definition: When LCOM4=1, then,

$$connectivity = 2 * \frac{e-(k-1)}{(k-1)(k-2)},$$

where e is the number of edges and k is the number of nodes.

Property 1 and Property 2: Connectivity metric is defined only for the cases where LCOM4 is equal to 1. When LCOM4 equals 1, the graph that represents the class is connected, and the number of edges in the graph is not less than k-1. The minimum value for connectivity metric is equal to 0 when the model to which the metric can be applied has the minimum possible number of interactions. The maximum value for connectivity is equal to 1 when the model has the maximum possible number of interactions (i.e., $e=k(k-1)/2$). Therefore, the connectivity metric satisfies Property 1 when it is applied on the models for which it is defined. However, a combination of LCOM4 and Connectivity does not satisfy Property 1. Property 2 is not applicable for the connectivity metric because the metric is undefined when the model of the class has no interactions. The combination of LCOM4 and Connectivity satisfies Property 2 because Connectivity solves the problem of the maximum value.

Property 3: Adding a cohesive interaction to the class implies adding an edge to the models that represents the class. As a result, the connectivity value increases, and therefore, the metric satisfies Property 3. Hence, the combination of LCOM4 and Connectivity satisfies Property 3.

Property 4: When two unrelated classes are merged, the model of the resulting class will have the number of edges equal to the summation of the number of edges in the models of both classes (i.e., $e_M=e_A+e_B$). To proof the satisfaction of connectivity metric to Property 4, we introduce the following model that facilitates the proof:

$$\frac{N(A)}{D(A)} \geq \frac{N(B)}{D(B)} \Rightarrow D(B)N(A) \geq D(A)N(B)$$

$$\Rightarrow [D(B)+D(A)]N(A) \geq D(A)[N(A)+N(B)]$$

$$\Rightarrow \frac{N(A)}{D(A)} \geq \frac{N(A)+N(B)}{D(A)+D(B)}$$

Given the following conditions:
Condition 1: N(M)≤N(A)+N(B)
Condition 2: D(M)≥D(A)+D(B)

$$\frac{N(A)}{D(A)} \geq \frac{N(A)+N(B)}{D(A)+D(B)} \geq \frac{N(M)}{D(M)}$$

This means that max{cohesion(A),cohesion(B)}≥ cohesion(M). Therefore, if a cohesion metric satisfies Conditions 1 and 2, it satisfies Property 4.

The connectivity metric is proved to satisfy the cohesion modules properties as follows:

$$N(M) = 2(e_M - (k+m-1)) = 2e_M - 2(k+m-1)$$
$$= 2(e_A + e_B) - 2(k+m-1)$$
$$= 2(e_A - (k-1)) + 2(e_B - (m-1)) - 2$$
$$= N(A) + N(B) - 2 < N(A) + N(B)$$

$$D(M) = (k+m-1)(k+m-2)$$
$$= (k-1)(k-2) + (m-1)(m-2) + m(k-1) +$$
$$k(m-1) + (k+m-2)$$
$$> (k-1)(k-2) + (m-1)(m-2) = D(A) + D(B)$$

Therefore, the metric satisfies Conditions 1 and 2, and, therefore, it satisfies Property 4. Hence, the combination of LCOM4 and Connectivity satisfies Property 4.

3.6 LCOM5 [7]

Definition:

$$LCOM5 = \frac{k - 1/l \sum_{i=1}^{l} \sum_{j=1}^{k} c_{ji}}{k-1},$$

where k is the number of methods, l is the number of attributes, and c_{ji} is the binary value at row j and column i in the binary matrix that represents which attribute is used in which method.

Property 1 and Property 2: The minimum value for LCOM5 is equal to 0 when each pair of methods share at least one common attribute (i.e., the model has the maximum number of cohesion interactions). The maximum value for LCOM5 depends on the number of methods in a class. That is, $k/(k-1)$, where k is the number of methods, when none of the methods share common attributes (i.e., the model does not have cohesion interactions). Therefore, LCOM5 satisfies Property 2, and it does not satisfy Property 1.

Property 3: The following proof shows that LCOM5 satisfies Property 3.

$$LCOM5(c') = \frac{k - 1/l \,(1 + \sum_{i=1}^{l} \sum_{j=1}^{k} c_{ji})}{k-1}$$

$$= \frac{k - 1/l \sum_{i=1}^{l} \sum_{j=1}^{k} c_{ji}}{k-1} - \frac{1}{l(k-1)} < LCOM5(c)$$

Property 4: In some cases, LCOM5 does not satisfy Property 4. For example, given two classes A and B such that each class has two methods and two attributes, and none of the methods use any attribute, LCOM5(A)=LCOM5(B)=2. When both classes are merged into class M, LCOM5(M)=(4-0)/(4-1)=1.33. Therefore, in this case, LCOM5(M)< min{LCOM5(A),LCOM5(B)}, which violates Property 4.

4 Conclusions and Future Work

This paper shows how to approve or disapprove the satisfaction of the lack-of-cohesion metrics to the class cohesion necessary properties. Table 1 summarizes the results. The results show that none of the lack-of-cohesion metrics satisfy Property 1, whereas the majority of the metrics satisfy Properties 2 and 4, and all of them satisfy Property 3. The dissatisfaction of the metric to some or all of the properties does not indicate that the metric is not a cohesion indicator. However, it raises questions about its usability as a cohesion indicator. Briand et al. show empirically that LCOM1, LCOM2, and LCOM4 are cohesion indicators, whereas, connectivity, LCOM4+connectivity, and LCOM5 are not [16,17].

Table 1: Summary of the theoretical validation results

Metric	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	N.A.	Yes	Yes	Yes
LCOM4+ connectivity	No	Yes	Yes	Yes
LCOM5	No	Yes	Yes	No

In the future, we plan to theoretically validate the other existing class cohesion metrics and empirically explore the relationships between the theoretical and empirical validation results.

Acknowledgment

The author would like to acknowledge the support of this work by Kuwait University Research Grant WI04/07.

References

[1] 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.

[2] 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.

[3] S.R. Chidamber and C.F. Kemerer, Towards a Metrics Suite for Object-Oriented Design, *Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, Special Issue of SIGPLAN Notices, Vol. 26, No. 10, 1991, pp. 197-211.

[4] 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.

[5] W. Li and S.M. Henry, Maintenance metrics for the object oriented paradigm. *In Proceedings of 1st International Software Metrics Symposium*, Baltimore, MD, 1993, pp. 52-60.

[6] 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.

[7] B. Henderson-Sellers, Software Metrics, Prentice Hall, Hemel Hempstaed, U.K., 1996.

[8] 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, pp. 259-262, 1995.

[9] 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.

[10] J. Wang, Y. Zhou, L. Wen, Y. Chen, H. Lu, and B. Xu, DMC: a more precise cohesion measure for classes. *Information and Software Technology*, Vol. 47, No. 3, 2005, pp. 167-180.

[11] L. Fernández, and R. Peña, A sensitive metric of class cohesion, *International Journal of Information Theories and Applications*, Vol. 13, No. 1, 2006, pp. 82-91.

[12] 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.

[13] L. C. Briand , S. Morasca , and V. R. Basili, Defining and validating measures for object-based high-level design, *IEEE Transactions on Software Engineering*, Vol. 25, No. 5, 1999, pp. 722-743.

[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, pp. 47-52. 1999.

[15] 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 (TOSEM)*, Vol. 15, No. 2, 2006, pp.123-149.

[16] L. Briand, J. Wust, and H. Lounis, Replicated case studies for investigating quality factors in object-oriented designs, Empirical Software Engineering, 6(1), 2001, pp. 11-58.

[17] L. C. Briand, J. Wust, J. Daly, and V. Porter, Exploring the relationship between design measures and software quality in object-oriented systems, Journal of System and Software, 51(3), 2000, pp. 245-273.