Mathematical Validation of Object-Oriented Class Cohesion Metrics

Jehad Al Dallal

***Abstract*—** Class cohesion is an object-oriented software quality attribute and refers to the extent to which the members of a class are related. Software developers use class cohesion measures 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 mathematically validated against the necessary properties of class cohesion. 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 mathematically validate sixteen class cohesion metrics using class cohesion properties. Results show that metrics differ considerably in satisfying the cohesion properties; some of them satisfy all properties, while others satisfy none.

Keywords — object-oriented class, software quality, class cohesion metric, class cohesion.

I. INTRODUCTION

A POPULAR goal of software engineering is to develop the techniques and 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's components. A module that has high cohesion performs one basic function and cannot be easily split into separate modules.

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 class members [1]. Assessing class cohesion and improving class quality accordingly during the object-oriented design phase allows for lower management costs in later phases. A class that has high cohesion cannot be easily split into separate classes. Highly cohesive classes are more understandable, modifiable, and maintainable [2].

Manuscript received September 20, 2009: Revised version received February 20, 2010.

Jehad Al Dallal is with Department of Information Science, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait (e-mail: jehad@cfw.kuniv.edu).

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 class members, researchers must validate the metric, both theoretically and empirically. Briand et al. [3] propose four properties that must be satisfied by all class cohesion metrics. If a metric does not satisfy any of these properties, the metric is ill defined, and its usefulness as a cohesion indicator is questionable [3]. These properties provide a supportive underlying theory for metrics. Empirical validation is necessary to show the usefulness of metrics. Despite its importance, few researchers focus on the theoretical validation of metrics. In this paper, we study the validity of sixteen class cohesion metrics, using the properties introduced by Briand et al. [3]. We provide mathematical proofs for the metrics that satisfy the cohesion properties and provide counter examples otherwise. Our results show that most of the metrics satisfy all or the majority of the properties.

This paper is organized as follows. Section 2 provides an overview of class cohesion metrics and necessary properties. In Section 3, sixteen class cohesion metrics are examined to determine whether they have the necessary properties. Finally, Section 4 presents conclusions and a discussion of future work.

II. RELATED WORK

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

A. Overview of class cohesion metrics

Yourdon et al. [23] propose seven levels of cohesion. These levels include coincidental, logical, temporal, procedural, communicational, sequential, and functional. The cohesion levels are listed in ascending order of their desirability. Since then, several cohesion metrics have been proposed for procedural and object-oriented programming languages. Different models are used to measure the cohesion of procedural programs, such as the control flow graph [24], the variable dependence graph [25], and program data slices [11, 26, 26, 28, 29]. Cohesion has also been measured indirectly by examining the quality of the structured designs [30, 31].

Researchers have proposed several class cohesion metrics in the literature. These metrics can be applicable based on high-level design (HLD) or low-level design (LLD) information. HLD class cohesion metrics rely on information related to class and method interfaces. The more numerous LLD class

cohesion metrics require an analysis of the algorithms used in the class methods (or the code itself if available) or access to highly precise method postconditions. The LLD cohesion metrics use finer-grained information than that used by HLD cohesion metrics. That is, based on the LLD, all method-method, method-attribute, and attribute-attribute interactions can be precisely defined. On the other hand, one advantage of HLD class cohesion metrics is that they identify potential cohesion issues early, during the HLD phase. Detecting class cohesion issues, and correcting the corresponding class artifacts later (during the LLD or implementation phase), is much more costly than performing the same tasks early (during the HLD phase). Improving class cohesion during the HLD phase saves development time, reduces development costs, and increases overall software quality.

Class cohesion metrics are based on the use or sharing of class attributes. For example, the LCOM1 metric counts the number of method pairs that do not share instance variables [15]. Chidamber and Kemerer [16] propose another version of the LCOM metric (LCOM2), which calculates the difference between the number of method pairs that do and do not share instance variables. Li and Henry [17] use an undirected graph that represents each method as a node and the sharing of at least one 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 the LCOM3 metric is extended in [18] 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 [18] introduce a connectivity metric to apply when the graph has one component. In addition, Henderson-Sellers [19] proposes a lack-of-cohesion in methods metric, LCOM5, that considers the number of methods referencing each attribute.

Bieman and Kang [4] describe two class cohesion metrics, Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC), to measure the relative number of directly connected pairs of methods and the relative number of directly or indirectly connected pairs of methods, respectively. TCC considers two methods to be connected if they share the use of at least one attribute. A method uses an attribute if the attribute appears in the method's body or the method invokes another method, directly or indirectly, that 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. Badri [5] introduces two class cohesion metrics, Degree of Cohesion-Direct (DC_D) and Degree of Cohesion-Indirect (DC_I), that 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. Briand et al. [3] propose a cohesion metric (called Coh) that computes cohesion as the ratio of the number of distinct attributes accessed in methods of a class. Fernandez and Pena [6] propose a class cohesion metric, called Sensitive Class Cohesion Metric (SCOM), that considers the cardinality of the intersection between each pair of methods. In the metric presented by Bonja and Kidanmariam [7], the degree of similarity between methods is used as a basis to measure 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. 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. The metric is called Class Cohesion (CC).

Bansiya et al. [8] 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. [9] 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 the CAMC metric. NHD calculates the average of the parameter agreements 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. Chae et al. [32] propose a metric called Cohesion Based on Member Connectivity (CBMC) that considers not only the number of interactions but also the patterns of the interactions between the methods in a class. The metric considers the ratio of the number of glue methods to the number of methods of interest. The number of glue methods equals the minimum number of methods required such that their removal causes the method-attribute interaction graph to become disjoint. Zhou et al. [33] introduce ICBMC, an improved version of CBMC, that considers the cut sets instead of glue methods. The cut set is the minimum set of edges such that their removal causes the method-attribute interaction graph to become disjoint. Related work in the area of software cohesion can be found in [10], [11], [13], and [14], and related work in the area of measuring software quality can be found in [20], [21], and [22].

B. Class cohesion metric properties

Briand et al. [3] define 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 modules' cohesion. Therefore, given two classes, c_1 and c_2, the cohesion of the merged class

c' must satisfy the following condition: cohesion(c')≤max {cohesion(c_1), cohesion(c_2)}.

III. Theoretical Validation

This section studies the theoretical validation of sixteen class cohesion metrics. The definition of each metric is overviewed, and the satisfaction of the four class cohesion necessary properties is proved mathematically or disproved, using a counter example.

A. LCOM1 [15]

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, 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. The cohesion of the merged class is less than the cohesion of each of the split classes; therefore, the LCOM1 metric satisfies Property 4.

B. LCOM2 [16]

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 $P_c \le Q_c$. 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-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, but 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, $P_c \le Q_c$, the cohesion of the original and the modified models equal 0. Otherwise, LCOM2 of the model decreases. Therefore, LCOM2 satisfies Property 3.

Property 4: When two unrelated classes are merged,

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

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

C. LCOM3 [17]

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 (i.e., k, where k is the number of methods, when the model has no interactions). 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, the LCOM3 metric satisfies Property 4.

D. LCOM4 [18]

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.

E. Connectivity [18]

Definition: When LCOM4=1,

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

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

Property 1 and Property 2: The 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 the 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 to 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 represent the class. 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 prove the connectivity metric's satisfaction of Property 4, we introduce the following *numerator-denominator cohesion proving* model:

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

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

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

Given the following conditions:
Condition 1: $N(M) \leq N(A)+N(B)$
Condition 2: $D(M) \geq D(A)+D(B)$

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

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

The connectivity metric is proved to satisfy the cohesive modules property as follows:

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

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

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

F. LCOM5 [19]

Definition: $LCOM5 = \dfrac{k - 1/l \sum_{i=1}^{l}\sum_{j=1}^{k} c_{ji}}{k-1}$, 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 shares 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. This value is defined as $k/(k-1)$, 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, but 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.

G. TCC and LCC [4], DC_D and DC_I [5], and Coh [3]

Definition: TCC, LCC, DC_D, DC_I, and Coh are defined as the relative number of cohesive interactions. They differ only in their definitions for the cohesive interactions, as discussed in Section 2.

Property 1 and Property 2: For the following discussion, the five metrics are referenced as R. The minimum value for R is 0 when the class has no cohesive interactions. The maximum value for R is 1 when the class has the maximum possible number of interactions. Therefore, the five metrics satisfy both Property 1 and Property 2.

Property 3: Since R is defined as the relative number of cohesive interactions, it increases when a cohesive interaction is added to the class model. Therefore, the five metrics satisfy Property 3.

Property 4: R is a relative metric, and, therefore, to prove its satisfaction of Property 4, we prove its satisfaction of Conditions 1 and 2 of the *numerator-denominator cohesion proving* model as follows:

When unrelated classes A and B are merged into class M, the number of interactions in M is equal to the summation of the number of interactions in classes A and B. Thus, N(M)=N(A)+N(B), which satisfies Condition 1. In addition,

$$D(M) = \frac{(k+m)(k+m-1)}{2} = \frac{k(k-1)}{2} + \frac{m(m-1)}{2} + km$$
$$= D(A) + D(B) + km \Rightarrow D(M) > D(A) + D(B)$$

R also satisfies Condition 2, and, therefore, the five metrics satisfy Property 4.

H. SCOM [6]

Definition: Given a class that has l attributes, the similarity between a pair of methods i and j, which reference the set of attributes I_i and I_j, respectively, is formally defined as follows:

$$Similarity(i, j) = \frac{\left|I_i \cap I_j\right|}{\min(\left|I_i\right|, \left|I_j\right|)} \cdot \frac{\left|I_i \cup I_j\right|}{l}$$

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.

Property 1 and Property 2: The minimum value for SCOM is equal to 0 when none of the methods share common attributes, which includes the case in which none of the methods use any attribute (i.e., the model does not have any cohesive interaction). The maximum value for SCOM is 1 when all methods share all attributes (i.e., the model has all possible cohesive interactions). Therefore, the SCOM metric satisfies both non-negativity and normalization, as well as null and maximum value cohesion properties.

Property 3: In some cases, when a cohesive interaction is added to the model, the SCOM value of the class decreases to some extent. Fig. 1 shows an example (classes A and B) for which the metric violates Property 3. This decrease is due to the fact that, in SCOM, the similarity is inversely proportional to the minimum number of attributes used in both methods. In some cases, adding a cohesive interaction increases this number and, consequently, decreases the similarity between some pairs of methods. When this decrease is greater than the increase of the similarity between some other pairs of methods in the class, the SCOM value decreases.

Property 4: To use our *numerator-denominator cohesion proving* model, we adjust the definition of similarity as follows:

$$Similarity(i, j) = \frac{\left|I_i \cap I_j\right|}{\min(\left|I_i\right|, \left|I_j\right|)} \cdot \frac{\left|I_i \cup I_j\right|}{l} = \frac{m_{ij}}{l}$$

Therefore,

$$SCOM(C) = \frac{\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} Similarity(i, j)}{\frac{k(k-1)}{2}} = \frac{\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} m_{ij}}{\frac{lk(k-1)}{2}} = \frac{N(C)}{D(C)}$$

When two unrelated classes A and B are merged into class M, m_{ij} between each pair of methods in class A and class B, does not change, because none of the parameters on which the m_{ij} value depends change. Since classes A and B are unrelated, m_{ij} between any method in class A and any method in class B equals 0 because none of the attributes are shared between the methods. Therefore, N(M)=N(A)+N(B) (i.e., satisfies Condition 1). The following proof shows that SCOM satisfies Condition 2.

$$D(M) = 0.5[(l+n)(k+m)(k+m-1)] = 0.5lk(k-1)$$
$$+ 0.5nm(m-1) + 0.5[(lm+nk)(k+m-1) + m(lk+nk)]$$
$$= D(A) + D(B) + 0.5[(lm+nk)(k+m-1) + m(lk+nk)]$$
$$> D(A) + D(B)$$

As a result, SCOM satisfies the cohesive modules property.

I. CC [7]

Definition: The similarity between a pair of methods i and j is defined as follows:

$Similarity(i, j) = \frac{\left|I_i \cap I_j\right|}{\left|I_i \cup I_j\right|}$, where I_i and I_j are the sets of attributes referenced by methods i and j, respectively. 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.

Property 1 and Property 2: The minimum value for CC equals 0 when none of the methods share common attributes, which includes the case in which none of the methods use any attribute (i.e., the model does not have any cohesive interaction). The maximum value for CC is 1 when all methods share the same set of attributes, which includes the case in which all methods share all attributes (i.e., the model has all possible cohesive interactions). Therefore, the CC metric satisfies both non-negativity and normalization, as well as null and maximum value cohesion properties.

Property 3: CC does not satisfy Property 3 in some cases. That is, when a cohesive interaction is added to a class, the counterintuitive result may be a class with a lower CC value, as depicted in classes C and D, shown in Fig. 1. This occurs because the addition of a cohesive interaction may increase the similarities between pairs of methods and decrease the similarities between other pairs of methods. In this case, the cohesion increases if the summation of the similarities between pairs of methods increases, and vice versa.

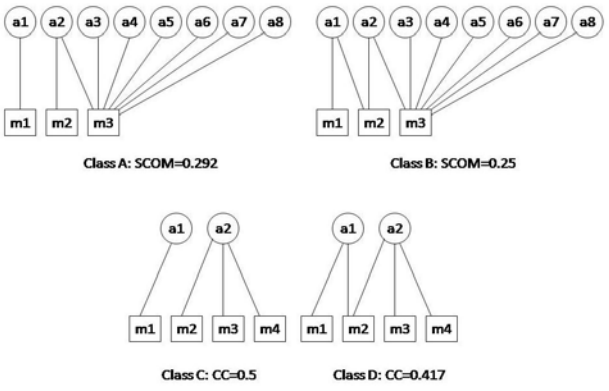

Fig. 1: Violation of CC and SCOM in terms of monotonicity [1].

Property 4: When two unrelated classes A and B are merged into class M, the similarity between each pair of methods in class A and class B does not change. This is because the similarity of a pair of methods is defined as the

ratio of the number of shared attributes between both methods to the number of attributes used by both methods. These two numbers remain the same in class M. Since classes A and B are unrelated, no similarities exist between methods in class A and methods in class B. Therefore, N(M)=N(A)+N(B) (i.e., CC satisfies Condition 1). CC also satisfies Condition 2 (the proof is identical to the corresponding one given above for the R metric). As a result, CC satisfies the cohesive modules property.

J. CAMC [8]

Definition: The ratio of the total number of 1s in the parameter occurrence matrix to the total size of the matrix.

Property 1 and Property 2: The minimum value for CAMC is $CAMC_{min} = (k+l-1)/kl$ when each parameter type is used by only one method, and the class type is used by all methods. The maximum value for CAMC is 1 when all methods have the same parameter types. Since the minimum value for CAMC is greater than 0, the metric does not satisfy Property 1. Since the model of the class used by CAMC cannot be free of cohesive interactions, the null and maximum value property is not applicable.

Property 3 and Property 4: CAMC is defined as the relative number of cohesive interactions in the model representing the class. Therefore, similar to R metrics, CAMC satisfies the monotonicity and cohesive modules properties.

K. NCAMC

Definition: The CAMC metric can be normalized by linearly scaling the interval $[(k+l-1)/kl,1]$ to [0,1] as follows:

$$NCAMC = \frac{CAMC - CAMC_{\min}}{1 - CAMC_{\min}} = \frac{CAMC - \frac{k+l-1}{kl}}{1 - \frac{k+l-1}{kl}}$$

$$= \frac{\frac{a}{kl} - \frac{k+l-1}{kl}}{1 - \frac{k+l-1}{kl}} = \frac{a-k-l+1}{kl-k-l+1}$$

If there is only one method in the class, the method uses all the parameter types, so CAMC=1. Therefore, NCAMC=1 if k=1, and NCAMC is calculated using the above formula otherwise. Counsell et al. [8] suggest omitting the type of class from the parameter occurrence matrix and calculating CAMC using the modified matrix. Given the parameter occurrence matrix without the type of class, CAMC can be calculated as follows [8]:

$$CAMC = \frac{a+k}{k(l+1)}$$

In this case, the minimum number of 1s in the matrix equals *l*, and the normalized metric can be calculated as follows:

$$NCAMC = \frac{CAMC - \frac{l+k}{k(l+1)}}{1 - \frac{l+k}{k(l+1)}} = \frac{\frac{a+k}{k(l+1)} - \frac{l+k}{k(l+1)}}{1 - \frac{l+k}{k(l+1)}} = \frac{a-l}{l(k-1)} \quad (1)$$

Note that the above normalized metrics are undefined when k=1. In this case, the class has one method that uses all the parameter types, and, by definition of the CAMC metric, the cohesion is equal to 1. We consider the normalized cohesion in this case to also be equal to 1. One of the major criticisms of the CAMC metric is that it does not distinguish between the cohesion of different matrices with the same number of 1s.

Property 1 and Property 2: The minimum value for NCAMC is when the matrix has the minimum number of 1s, which is *l*. By substituting *a* in Formula 1 by *l*, we get NCAMC=0. The maximum value for NCAMC is when the matrix has the maximum number of 1s, which is *kl*. By substituting *a* in Formula 1 by *kl*, we get NCAMC=1. As discussed above, NCAMC=1 when k=1. NCAMC ranges in the interval [0, 1], and, therefore, it is normalized. Since the model of the class used by NCAMC cannot be free of cohesive interactions, the null and maximum value property is not applicable.

Property 3: Given specific methods and parameter types, adding a cohesive method-method interaction to the class means making a parameter type shared between two methods. This is represented in the matrix by changing two entries in a column in the matrix from 0 to 1 if both methods were not using the parameter type or changing one entry from 0 to 1 if one of the methods was using the parameter type. As a result, adding a cohesive method-method interaction implies incrementing *a* in Formula 1 by *n*, where *n* is either 1 or 2. When k=1, all entries of the matrix are equal to 1, and no more cohesive interactions can be added. When k>1,

$$NCAMC_{new} = \frac{(a+n)-l}{l(k-1)} = \frac{n}{l(k-1)} + \frac{a-l}{l(k-1)} \geq$$

$$\frac{a-l}{l(k-1)} = NCAMC_{old}$$

Therefore, NCAMC satisfies the monotonicity property.

Property 4: Merging two unrelated classes *c1* and *c2* means that none of the methods in each of the two classes share common parameter types. If class *c1* has *k* methods and *l* distinct parameter types, and class *c2* has *m* methods and *n* distinct parameter types, the parameter occurrence matrix of the merged class *c3* will have *k*+*m* rows and *l*+*n* columns. In this case,

$$NCAMC_{c3} = \frac{a_{c1} + a_{c2} - (l+n)}{(l+n)(k+m-1)}$$

$$= \frac{(a_{c1} - l) + (a_{c2} - n)}{l(k-1) + n(m-1) + lm + nk} \quad (2)$$

The cohesion of a system consisting of more than one module is the weighted sum of the cohesion of each module in the system [3]. As a result, the cohesion of the two classes *c1* and *c2* is as follows:

$$NCAMC_{c1} + NCAMC_{c2} = \frac{(a_{c1} - l) + (a_{c2} - n)}{l(k-1) + n(m-1)} \quad (3)$$

$$> NCAMC_{c3}$$

Since the cohesion of the merged class is less than the cohesion of the system consisting of the two classes, the

NCAMC metric satisfies the cohesive modules property. The above case is applicable when each of the classes has more than one method. If each of the two classes has one method, the cohesion of each class is equal to 1, and, therefore, the cohesion of the system consisting of the two classes is equal to 1. The cohesion of the merged class is equal to 0 (i.e., by substituting a_{c1} by l and a_{c2} by n in Formula 2), which is less than the cohesion of the split classes. Finally, if one of the classes has one method, by substituting a_{c1} by l or a_{c2} by n in Formula 2, Proposition 3 is still satisfied. As a result, in all cases, NCAMC satisfies the cohesive modules property.

L. NHD [9]

Definition: $NHD = 1 - \frac{2}{lk(k-1)} \sum_{j=1}^{l} x_j (k - x_j)$,

where k is the number of methods, l is the number of distinct parameter types, and x_j is the number of 1s in the jth column of the parameter occurrence matrix (i.e., number of methods that use parameter j).

Property 1 and Property 2: The NHD metric has the minimum value when each column in the matrix that models the class has the maximum possible disagreements, by setting x_j=k/2 in the NHD formula [9]. In this case, NHD_{min} = (k-2)/[2(k-1)]. The maximum value for NHD is equal to 1 when the matrix contains only 1s (i.e., the class has all possible interactions). Since the minimum value for NHD is greater than 0, the metric does not satisfy Property 1. Since the model of the class used by NHD cannot be free of cohesive interactions, the *null and maximum value* property is not applicable.

Property 3: Adding a cohesive interaction to the class is represented in the matrix by changing two entries in a column in the matrix from 0 to 1 if neither method used the parameter type, or changing one entry from 0 to 1 if one of the methods was using the parameter type. If a column n in the matrix has x_n>k/2, where x_n is the number of 1s in the column, according to the NHD formula, the value of NHD after adding the cohesive interaction is less than it was before adding the cohesive interaction, which violates Property 3.

Property 4: In some cases, NHD violates the cohesive modules property. For example, consider two classes, A and B, where each has two methods: One of the methods has two parameter types, and the other method does not have any parameter types. In this case, the cohesion of each class is equal to 0. When the two classes are merged, the new matrix is 4×4, and the NHD value of the merged class is 0.5, which is greater than the NHD value of each of classes A and B.

IV. Conclusions and Future Work

This paper shows how to prove or disprove the satisfaction of class cohesion metrics to the necessary properties for class cohesion. Table I summarizes the results found in this paper and another related paper [34]. The results show that among the considered class cohesion metrics:

- Eight (42%) metrics satisfy all properties.
- Six (37%) metrics satisfy only three properties.
- Four (21%) metrics satisfy only two properties.
- One (5%) metric does not satisfy any property.
- Ten (53%) metrics satisfy Property 1.
- Fourteen (74%) metrics satisfy Property 2.
- Fifteen (79%) metrics satisfy Property 3.
- Seventeen (89%) metrics satisfy Property 4.

In general, this means that 42% 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 I: 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
TCC	Yes	Yes	Yes	Yes
LCC	Yes	Yes	Yes	Yes
DC_D	Yes	Yes	Yes	Yes
DC_I	Yes	Yes	Yes	Yes
Coh	Yes	Yes	Yes	Yes
SCOM	Yes	Yes	No	Yes
CC	Yes	Yes	No	Yes
CAMC	No	N.A.	Yes	Yes
NCAMC	Yes	N.A.	Yes	Yes
NHD	No	N.A.	No	No
CBMC [34]	Yes	Yes	No	Yes
ICBMC [34]	Yes	Yes	Yes	Yes

In the future, we plan to theoretically validate 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 WI03/07.

References

[1] J. Al Dallal and L. Briand, A precise method-method interaction-based cohesion metric for object-oriented classes, TR, *Simula Research Laboratory*, 2009, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, in press.

[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] 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.

[4] 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.

[5] 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..

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] J. Al Dallal, A design-based cohesion metric for object-oriented classes, *International Journal of Computer Science and Engineering*, 2007, Vol. 1, No. 3, pp. 195-200.

[11] J. Al Dallal, Software similarity-based functional cohesion metric, *IET Software*, 2009, Vol. 3, No. 1, pp. 46-57.

[12] J. Al Dallal, Theoretical validation of object-oriented lack-of-cohesion metrics, proceedings of the *8th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS 2009)*, Cambridge, UK, February 2009.

[13] J. Al Dallal and L. Briand, An object-oriented high-level design-based class cohesion metric, TR, *Simula Research Laboratory*, 2009.

[14] J. Al Dallal, Measuring the discriminative power of object-oriented class cohesion metrics, *IEEE Transactions on Software Engineering*, In press.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] D. Kushwaha and A. Misra, A complexity measure based on information contained in the software, *5th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS 2006)*, Madrid, Spain, Feb. 2006.

[21] J. Alghamdi, Measuring software coupling, *Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, p.6-12, February 16-19, 2007, Corfu Island, Greece.

[22] 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*, p.1-6, February 13-15, 2005, Salzburg, Austria.

[23] E. Yourdon and L. Constantine, Structured Design, Prentice-Hall, Englewood Cliffs, 1979.

[24] T. Emerson, A discriminant metrics for module cohesion, *In Proceedings of the 7th International Conference on Software Engineering*, 1984, pp. 294-303.

[25] A. Lakhotia, Rule-based approach to computing module cohesion, *Proceedings of the 15th international conference on Software Engineering*, Baltimore, US, 1993, pp. 35-44.

[26] L. Ott and J. Thuss, Slice based metrics for estimating cohesion, *Proceedings of the First International Software Metrics Symposium*, Baltimore, 1993, pp. 71-81.

[27] J. Bieman and L. Ott, Measuring functional cohesion, *IEEE Transactions on Software Engineering*, Vol. 20, No. 8, 1994, pp. 644-657.

[28] T. Meyers and D. Binkley, An empirical study of slice-based cohesion and coupling metrics, *ACM Transactions on Software Engineering Methodology*, 17(1), 2007, pp. 2-27.

[29] J. Al Dallal, Efficient program slicing algorithms for measuring functional cohesion and parallelism, *International Journal of Information Technology*, Vol. 4, No. 2, 2007, pp. 93-100.

[30] D. Troy and S. Zweben, Measuring the quality of structured designs, *Journal of Systems and Software*, 2, 1981, pp. 113-120.

[31] J. Bieman and B. Kang, Measuring design-level cohesion, *IEEE Transactions on Software Engineering*, Vol. 24, No. 2, 1998, pp. 111-124.

[32] H. S. Chae, Y. R. Kwon, and D. Bae, A cohesion measure for object-oriented classes, *Software—Practice & Experience*, 30(12), 2000, pp.1405-1431.

[33] Y. Zhou, B. Xu, J. Zhao, and H. Yang, ICBMC: an improved cohesion measure for classes, *Proc. of International Conference on Software Maintenance*, 2002, pp. 44-53.

[34] Y. Zhou, J. Lu, H. Lu, and B. Xu, A comparative study of graph theory-based class cohesion measures, *ACM SIGSOFT Software Engineering Notes*, 29(2), 2004, pp. 13-13.