Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics

Jehad Al Dallal

Abstract—Several object-oriented cohesion metrics have been proposed in the literature. These metrics aim to measure the relationship between class members, namely, methods and attributes. Different metrics use different models to represent the connectivity pattern of cohesive interactions (CPCI) between class members. Most of these metrics are normalized to allow for easy comparison of the cohesion of different classes. However, in some cases, these metrics obtain the same cohesion values for different classes that have the same number of methods and attributes but different CPCIs. This leads to incorrectly considering the classes to be the same in terms of cohesion, even though their CPCIs clearly indicate that the degrees of cohesion are different. We refer to this as a lack of discrimination anomaly (LDA) problem. In this paper, we list and discuss cases in which the LDA problem exists, as expressed through the use of 16 cohesion metrics. In addition, we empirically study the frequent occurrence of the LDA problem when the considered metrics are applied to classes in five open source Java systems. Finally, we propose a metric and a simulation-based methodology to measure the discriminative power of cohesion metrics. The discrimination metric measures the probability that a cohesion metric will produce distinct cohesion values for classes with the same number of attributes and methods but different CPCIs. A highly discriminating cohesion metric is more desirable because it exhibits a lower chance of incorrectly considering classes to be cohesively equal when they have different CPCIs.

Keywords—Cohesive interactions, connectivity pattern, discrimination metric, discriminative power, lack of discrimination anomaly, object-oriented class cohesion.

1 Introduction

Cohesion is a major software quality attribute referring to the relatedness of the components in a software module [11]. A class is the basic unit in object-oriented programs, and it consists of two types of members: attributes and methods. Class cohesion indicates the extent to which the members of a class are related. Several class cohesion metrics have been introduced in the literature to measure class cohesion. These metrics use different formulas which are applied to different class representation models. In addition, they are applicable at either high or low-level design phases. High-level design (HLD) metrics such as those proposed by Briand et al. [14], Bansiya et al. [9], Counsell et al. [23], and Al Dallal and Briand [5], require information that is available during the HLD phase such as the types of attributes and method parameters. Low-level design (LLD) metrics, such as those proposed by Chidamber and Kemerer [21], Chidamber and Kemerer [22], Hitz and Montazeri [31], Bieman and Kang [10], Chen et al. [19], Badri and Badri [8], Wang et al. [39], Bonja and Kidanmariam [12], and Fernández and Peña [25], require information that is available during the LLD phase, such as attributes referenced by the methods. Typically, the cohesion-related information is collected and depicted using a model that represents the connections between the class members under consideration. The way in which the members are cohesively connected is referred to here as the Connectivity Pattern of Cohesive Interactions (CPCI) [18], [40].

Cohesion is measured during the design phase to predict software quality. Measuring class cohesion allows for comparing different class designs in terms of cohesion and selecting the best design. In addition, it can be used as an indicator to improve the quality of weakly cohesive classes. To validate a cohesion metric theoretically, Briand et al. [13] propose two related theoretical properties: normalization and monotonicity. Normalization holds that the cohesion measure belongs to a specific interval [0, Max]. Monotonicity holds that adding cohesive interactions to the module cannot decrease its cohesion. The normalization property allows for easy comparison of the cohesion of different classes. The monotonicity property ensures that more cohesive classes do not exhibit lower cohesion values than less cohesive ones. However, neither normalization nor monotonicity necessarily ensures that classes of clearly different cohesion levels will be distinguished. For example, in Fig. 1, rectangles, circles, and links represent the methods, attributes, and use of attributes by methods, respectively, of two classes that exhibit the same cohesion values according to several cohesion metrics, such as Tight Class Cohesion [10] and Degree of Cohesion-Direct [8]. However, Class B is clearly more cohesive than Class A because several members of Class A are completely disjoint and have no cohesive interactions among themselves or with the other class members, whereas all members of Class B are directly or indirectly related.

In some cases, a cohesion metric exhibits both the normalization and monotonicity properties, but it produces the same cohesion values for widely spread variations in

• *The author is with the Department of Information Sciences, Kuwait University, PO Box 5969, Safat 13060, Kuwait. E-mail: j.aldallal@ku.edu.kw.*

Manuscript received 11 Nov. 2009; revised 12 Jan. 2010; accepted 21 Feb. 2010; published online 10 Nov. 2010.
Recommended for acceptance by K. Inoue.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2009-11-0323.
Digital Object Identifier no. 10.1109/TSE.2010.97.

0098-5589/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

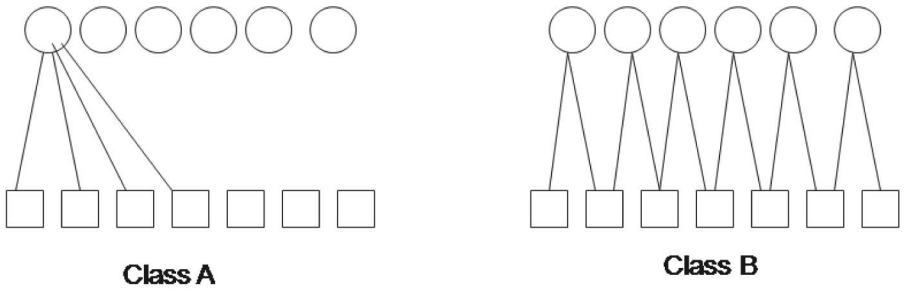

Fig. 1. Sample CPCIs for hypothetical Classes A and B.

CPCIs. In these cases, the chances that the metric will differentiate between differently cohesive classes are low; thus, the metric is ill-defined and its usefulness as a cohesion indicator is questionable. Note that, due to the normalization property, it is possible to obtain the same cohesion values for different classes that have different numbers of methods or different numbers of attributes. This is because the metrics that satisfy the normalization property measure the cohesion degree relatively. However, it is an anomaly for a cohesion metric to obtain the same cohesion value for different classes that have the same number of methods and attributes but different CPCIs. We refer to such an anomaly as a Lack of Discrimination Anomaly (LDA).

Obtaining the same cohesion value, using a cohesion metric, for two differently cohesive classes indicates the presence of an anomaly in the metric definition. In this case, the metric incorrectly indicates that both classes are the same in terms of cohesion, and therefore they deserve the same refractoring attention. As a result, if the percentage of cases for which the metric incorrectly considers two classes to be the same in terms of cohesion is high, this may well discourage software developers from applying the metric to assess the quality of their products.

In this paper, we introduce a property called discriminative power, defined as *the ability of a class cohesion metric to obtain different cohesion values for classes of the same number of methods and attributes but different CPCIs*. We study the discriminative power of 16 cohesion metrics theoretically and empirically. In a theoretical study, we list and discuss the cases for which each of the considered cohesion metrics violates intuition from the perspective of discrimination. In an empirical study, we investigate the occurrence of the LDA problem for the metrics considered when they are applied to the classes of five open source systems. The results show that the LDA problem occurs relatively often and thus deserves attention. Finally, we introduce a discrimination-measuring metric and a simulation-based methodology to obtain and compare the discriminative power of the 16 cohesion metrics under consideration. The discrimination metric measures the probability that a cohesion metric will differentiate between two classes that have the same number of attributes and methods but different CPCIs. The metric is applied to each class model, where a class model is defined by its number of methods and attributes. For each class model, all possible different CPCIs are considered. Cohesion is measured for each of these CPCIs using 16 cohesion metrics. The results are used by the discrimination metric to measure and compare the discriminative power of these cohesion metrics. Results show that the cohesion metrics differ considerably in terms of their discriminative power and that there is a strong correlation between the discriminative power of a metric and its ability to predict faulty classes. Therefore, our proposed discriminative power property can be used as a quality feature for cohesion metrics. Highly discriminating cohesion metrics are expected to be better defined, and thus more desirable, than those of low discriminative power. As a result, we recommend considering the discriminative power property whenever a new metric is introduced and adopting cohesion metrics that offer high discriminative value.

The paper is organized as follows: Section 2 reviews related work. Section 3 lists and discusses LDA cases of the class cohesion metrics under consideration. Section 4 describes an empirical study that investigates the occurrence of the LDA problem in open source systems. Section 5 defines the discrimination-measuring metric and process and showcases their application to class cohesion metrics. Finally, Section 6 concludes the paper and discusses future work.

2 Related Work

Several metrics have been proposed in the literature to measure class cohesion during HLD and LLD phases. In this paper, we consider 16 metrics, including LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, Coh, TCC, LCC, DC_D, DC_I, CC, SCOM, LSCC, CAMC, NHD, and SNHD, as defined in Table 1. The first 13 metrics are applicable during the LLD phase and the rest are applicable during the HLD phase.

Some authors have reported problems in discrimination with some class cohesion metrics, but without further discussion or study. For example, Briand et al. [13] criticize LCOM3 because a connected component can have different degrees of connectivity. They also mention that LCOM2 has little discriminative power. The same criticism is leveled at LCOM2 by Bonja and Kidanmariam [12]. Counsell et al. [23] indicate that CAMC does not distinguish between classes that have the same number of methods, the same number of distinct parameter types, and the same total number of cohesive interactions but that exhibit different connectivity patterns. They also indicate that NHD fails to distinguish between classes that have the same number of methods and distinct parameter types, where each parameter type is used in the same number of methods, regardless of the connectivity pattern. Bonja and Kidanmariam [12] provide examples to show that CC has more discriminative power than LCOM2 or CAMC. Fernández and Peña [25] provide examples showing that SCOM has more discriminative power than LCOM2, LCOM3, and LCOM5. Finally,

TABLE 1
The Definitions of the Class Cohesion Metrics Considered

Class Cohesion Metric	Definition/Formula
Lack of Cohesion of Methods (LCOM1) (Chidamber and Kemerer 1991)	LCOM1= Number of pairs of methods that do not share attributes.
LCOM2 (Chidamber and Kemerer 1994)	P= Number of pairs of methods that do not share attributes. Q= Number of pairs of methods that share attributes. $LCOM2 = \begin{cases} P-Q & if\ P-Q \geq 0 \\ 0 & otherwise \end{cases}$
LCOM3 (Li and Henry 1993)	LCOM3= Number of connected components in the graph that represents each method as a node and the sharing of at least one attribute as an edge.
LCOM4 (Hitz and Montazeri 1995)	Similar to LCOM3 and additional edges are used to represent method invocations.
LCOM5 (Henderson-Sellers 1996)	LCOM5=$(a-kl)/(l-kl)$, where l is the number of attributes, k is the number of methods, and a is the summation of the number of distinct attributes accessed by each method in a class.
Coh (Briand et al. 1998)	Coh=a/kl, where a, k, and l have the same definitions above.
Tight Class Cohesion (TCC) (Bieman and Kang 1995)	TCC= Relative number of directly connected pairs of methods, where two methods are directly connected if they are directly connected to an attribute. A method m is directly connected to an attribute when the attribute appears within the method's body or within the body of a method invoked by method m directly or transitively.
Loose Class Cohesion (LCC) (Bieman and Kang 1995)	LCC=Relative number of directly or transitively connected pairs of methods, where two methods are transitively connected if they are directly or indirectly connected to an attribute. A method m, directly connected to an attribute j, is indirectly connected to an attribute i when there is a method directly or transitively connected to both attributes i and j.
Degree of Cohesion-Direct (DC_D) (Badri 2004)	DC_D= Relative number of directly connected pairs of methods, where two methods are directly connected if they satisfy the condition mentioned above for TCC or if the two methods directly or transitively invoke the same method.
Degree of Cohesion-Indirect (DC_I) (Badri 2004)	DC_I= Relative number of directly or transitively connected pairs of methods, where two methods are transitively connected if they satisfy the same condition mentioned above for LCC or if the two methods directly or transitively invoke the same method.
Class Cohesion (CC) (Bonja and Kidanmariam 2006)	CC= Ratio of the summation of the similarities between all pairs of methods to the total number of pairs of methods. The similarity between methods i and j is defined as: $Similarity(i,j) = \frac{\lvert I_i \cap I_j \rvert}{\lvert I_i \cup I_j \rvert}$, where I_i and I_j are the sets of attributes referenced by methods i and j, respectively.
Class Cohesion Metric (SCOM) (Fernandez and Pena 2006)	SCOM= Ratio of the summation of the similarities between all pairs of methods to the total number of pairs of methods. The similarity between methods i and j is defined as: $Similarity(i,j) = \frac{\lvert I_i \cap I_j \rvert}{\min(\lvert I_i \rvert, \lvert I_j \rvert)} \cdot \frac{\lvert I_i \cup I_j \rvert}{l}$, where l is the number of attributes
Low-level design Similarity-based Class Cohesion (LSCC) (Al Dallal and Briand 2009b)	$LSCC(C) = \begin{cases} 0 & \text{if } k=0 \text{ or } l=0, \\ 1 & \text{if } k=1, \\ \frac{\sum_{i=1}^{l} x_i(x_i-1)}{lk(k-1)} & \text{otherwise.} \end{cases}$ Where l is the number of attributes, k is the number of methods, and x_i is the number of methods that reference attribute i.
Cohesion Among Methods in a Class (CAMC) (Counsell et al. 2006)	CAMC= a/kl, where l is the number of distinct parameter types, k is the number of methods, and a is the summation of the number of distinct parameter types of each method in the class. Note that this formula is applied on the model that does not include the self parameter type used by all methods.
Normalized Hamming Distance (NHD) (Counsell et al. 2006)	$NHD = 1 - \frac{2}{lk(k-1)} \sum_{j=1}^{l} x_j(k-x_j)$, where k and l are as defined above for CAMC and x_j is the number of methods that have a parameter of type j.
Scaled Normalized Hamming Distance (NHD) (Counsell et al. 2006)	SNHD=the closeness of the NHD metric to the maximum value of NHD compared to the minimum value.

Al Dallal [2] proposes an HLD cohesion metric and provides examples to compare the proposed metric to CAMC and NHD in terms of discriminative power. In this paper, we propose a formal definition for discriminative power and illustrate how to measure it. In addition, we show examples to illustrate which of the 16 metrics considered have LDA problems.

The validity of a metric has to be studied both empirically and theoretically [34]. Empirical validation tests whether the measured and predicted values are consistent with each other. Theoretical validation tests whether the metric exhibits the required properties of the measured attribute. Several researchers have addressed how to empirically validate class cohesion metrics, including [14], [16], [17], [15], [7], [28], [1], [36], [5] and [6]. Several properties are introduced to validate software metrics theoretically, as summarized in Table 2. The first four properties were introduced by Briand et al. [13], the following six properties were explained by Chidamber

TABLE 2
A Summary of Software Metric Properties

Name	Definition
Non-negativity and normalization	The cohesion measure belongs to a specific interval [0, Max].
Null value and maximum value	The cohesion of a class equals 0 if the class has no cohesive interactions, and the cohesion of a class is equal to Max if all possible interactions within the class are present.
Monotonicity	The addition of cohesive interactions to the class cannot decrease its cohesion.
Cohesive modules	The merging of two unrelated classes into one class does not increase the class's cohesion.
Non-coarseness	Not every class has the same cohesion value.
Granularity	Given an application, there is finite number of classes that have a certain cohesion value as produced by the metric.
Non-uniqueness	There exist at least two classes with the same cohesion values.
Implementation/design details are important	Classes that provide identical functionalities can have different cohesion values.
No equivalence of interaction	Given three classes x, y, and z, if x and y exhibit the same cohesion value, the classes resulting from merging z with each of x and y can exhibit different cohesion values.
Renaming	Renaming a class, an attribute, or a method does not affect the cohesion value.
Simplicity	The definition of the metric is understandable.
Accuracy	The metric is related to the measured attribute.
Automation	The measuring process using the metric can be automated.
Value of implementation	The metric can be measured during early development stages or it evaluates the success of the implementation.
Sensitivity	The metric produces different values for nonequivalent entities.
Measured scale	The metric is of rate or absolute scale rather than nominal or ordinal scales.
Prescription	The metric indicates how to enhance the measured element.
The representation condition of measurement theory	The metric obtains values that are consistent with intuition concerning the attribute being measured. This means that if we have a relation such as "more cohesive than", the representation condition requires that a measurement mapping M must map entities (i.e. classes) into numbers and empirical relations into numerical relations such that the empirical relations preserve and are preserved by the numerical relations. In terms of cohesion, this means that class A is more cohesive than class B if and only if M(A)>M(B).

and Kemerer [22], the next seven properties were listed by Fernández and Peña [25], and the last property was discussed by Fenton and Pfleeger [24]. Note that some of the properties are specific for class cohesion metrics and others are generally associated with almost any software metric. Authors [13], [14], [41], [20], [30], [12], [25], [3], [4], [5], and [6] used some of these properties—and especially the first four—to validate several class cohesion metrics. The sensitivity property is close to our introduced discriminative power property, but ours is more refined and is adapted for class cohesion metrics. We therefore propose that our property can be added to the list of those that can be used to validate class cohesion metrics theoretically. In the next section, when discussing the LDA cases, we follow the representation condition of measurement theory [24] to indicate which class is less or more cohesive than the other, though the problem of lack of consensus applies in some cases. Note that the representation condition is a more strict condition than any of the others listed in Table 2, and "Sensitivity" is a weaker formulation of the same concept.

3 Lack of Discrimination Anomaly Cases

In this section, we list and discuss the LDA cases, namely, those in which each metric fails to discriminate between levels of class cohesion. Sixteen cohesion metrics are considered: LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, Coh, CAMC, TCC, LCC, DC_D, DC_I, CC, SCOM, LSCC, NHD, and SNHD. In the following discussion, the class structural information taken in to account by all the considered metrics, based on the links between methods and attributes/parameter-types is used as a basis to evaluate cohesion intuitively. Other approaches, such as the ones that consider the class semantic information to evaluate cohesion (e.g.,

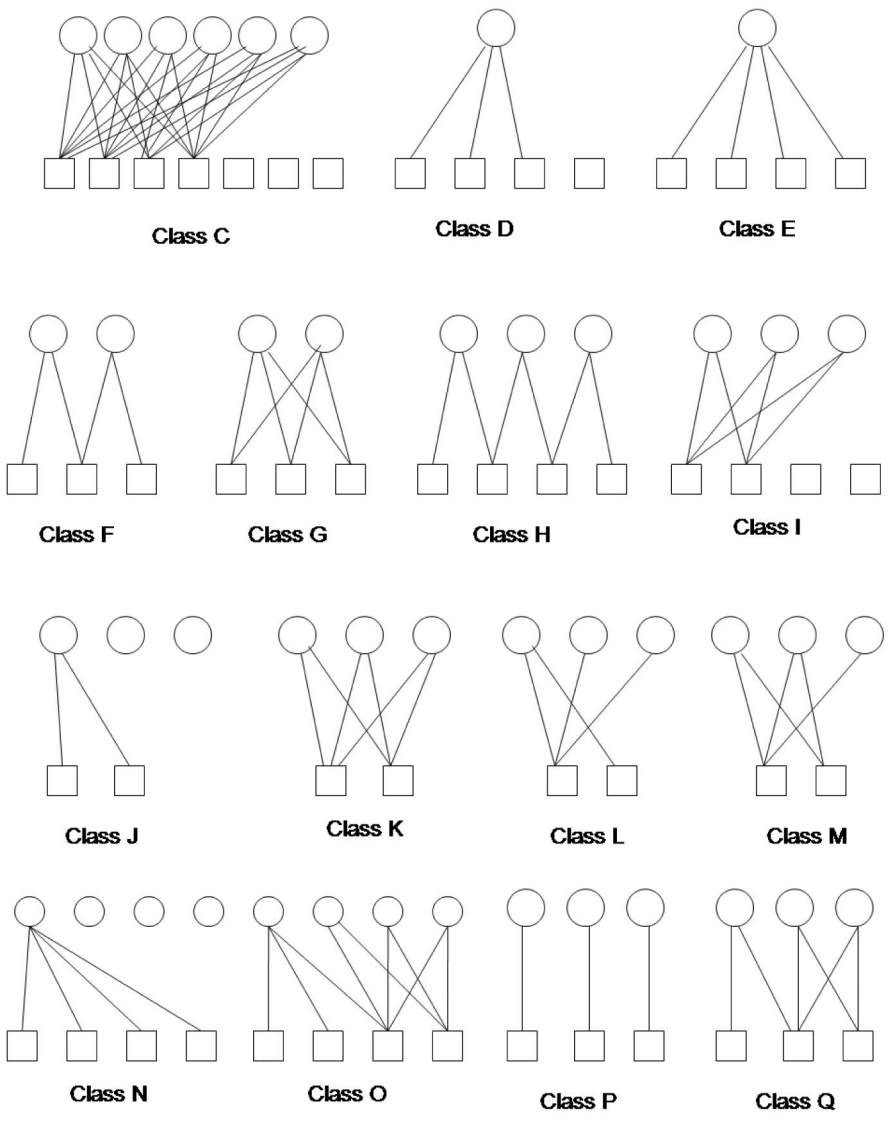

Fig. 2. Sample CPCIs for hypothetical Classes C-Q.

[36]), are not considered because none of the metrics used takes such information into account, though, ideally, combinations of structural and semantic information have to be considered to better evaluate cohesion intuitively.

3.1 LCOM1

LCOM1 obtains the same value for two classes with the same class model and different CPCIs in two LDA cases. The first LDA case occurs when the number of method pairs that share common attributes is the same in both classes, regardless of the number of shared attributes. For example, Class A, shown in Fig. 1, and Class C, shown in Fig. 2, have the same LCOM1 value ($\mathrm{LCOM1} = 15$), whereas four methods in Class A share a common attribute and each of the corresponding methods in Class C share six attributes. In this case, intuition indicates that Class C is more cohesive than Class A because it has more cohesive interactions. In Fig. 2, rectangles, circles, and links represent methods, attributes, and use of attributes by methods, respectively. The second LDA case occurs when the number of method pairs that share common attributes is the same in both classes, regardless of which attributes are shared. For example, Classes A and B, shown in Fig. 1, have the same LCOM1 value ($\mathrm{LCOM1} = 15$), even though four methods in Class A share the same common attribute and each pair of methods in Class B shares a different attribute. In this case, intuition says that Class B is more cohesive than Class A because all the members of Class B are directly or indirectly connected, whereas most of the members of Class A are disjoint.

3.2 LCOM2

LCOM2 is correlated to LCOM1. Given that k is the number of methods in a class, NP is the number of possible pairs of methods, and Q and P are the number of pairs of methods with and without shared attributes, respectively; if $P \geq Q$, the relation between LCOM2 and LCOM1 will be obtained as follows:

$$\begin{aligned} LCOM2 &= P - Q = P - (NP - P) = 2P - NP \\ &= 2LCOM1 - NP = 0.5[4LCOM1 - k(k-1)]. \end{aligned}$$

Therefore, LCOM2 will obtain the same value for two classes with the same class model and different CPCIs in the same two LDA cases indicated above for LCOM1. In

addition, the two classes will have the same LCOM2 value if $P \leq Q$. In this case, the value of LCOM2 is zero, regardless of the magnitude of the difference between P and Q in each of the two classes. This is an LDA because, in some cases, one of the two classes can be clearly more cohesive than the other. For example, Classes D and E, shown in Fig. 2, have the same LCOM2 value ($\mathrm{LCOM2} = 0$), whereas some of the methods in Class D share a common attribute and all of the methods in Class E share a common attribute. In this case, intuition indicates that Class E is more cohesive than Class D because the attribute in Class E is shared by more methods.

3.3 LCOM3 and LCOM4

LCOM3 and LCOM4 employ the same formula. The only difference between them is in their definitions of cohesive interactions, as discussed in Section 2. Both metrics have the same first LDA case indicated above for LCOM1 because neither LCOM3 nor LCOM4 considers the number of shared attributes between pairs of methods in the cohesion calculation. In addition, two classes will have the same LCOM3 and LCOM4 value if the number of disjoint components is the same in both classes, regardless of the CPCI of each of the disjoint components. For example, Classes F and G, shown in Fig. 2, have the same LCOM3 and LCOM4 values ($\mathrm{LCOM3} = \mathrm{LCOM4} = 1$), whereas each attribute in Class F is shared by two methods and each attribute in Class G is shared by three methods. In this case, intuition states that Class G is more cohesive than Class F because the attributes in Class G are shared by more methods.

3.4 LCOM5, Coh, and CAMC

Given two classes that have the same models, LCOM5 and Coh are inversely correlated, and thus they have the same LDA cases. The relation between the formulas of LCOM5 and Coh is as follows:

$$\begin{aligned}\mathrm{LCOM5} &= (a - kl)/(l - kl) = (Coh.kl - kl)/(l - kl)\\ &= kl(Coh - 1)/(l - kl) = kl(1 - Coh)/(kl - l),\end{aligned}$$

where k is the number of methods and l is the number of attributes.

Both LCOM5 and Coh have the same value for two classes that share the same class model but that exhibit different CPCIs, provided that the two classes have the same number of attributes referenced by methods and regardless of the distribution of these references. This can be an LDA in some cases. For example, Classes H and I, shown in Fig. 2, have the same LCOM5 and Coh values ($\mathrm{LCOM5} = 0.67$ and $\mathrm{Coh} = 0.5$), whereas all members in Class H are directly or indirectly connected and some of the members of Class I are disjoint.

The formula for CAMC is similar to that of Coh in the sense that it calculates the ratio of the cohesive interactions. However, CAMC and Coh differ in their definitions for cohesive interactions and the design phase during which they are applicable. Therefore, CAMC gives the same value for two classes sharing the same class model but with different CPCIs if the two classes have the same number of distinct types of parameters in their methods, regardless of the distribution of these cohesive interactions. The same example of Classes H and I applies for CAMC ($\mathrm{CAMC} = 0.5$) but, in this case, the circles represent distinct parameter types.

3.5 TCC, LCC, $\mathrm{DC_D}$, and $\mathrm{DC_I}$

TCC, LCC, $\mathrm{DC_D}$, and $\mathrm{DC_I}$ have the same formulas, but they differ in their cohesive interaction definitions. TCC is correlated with LCOM1. Given that NP is the number of possible pairs of methods and Q and P are the number of pairs of methods with and without shared attributes, respectively, the relation between TCC and LCOM1 is as follows:

$$\begin{aligned}\mathrm{TCC} = Q/NP = (NP - P)/NP = (NP - LCOM1)/NP\\ = 1 - LCOM1/NP.\end{aligned}$$

As a result, TCC, LCC, $\mathrm{DC_D}$, and $\mathrm{DC_I}$ have the same two LDA cases indicated above for LCOM1. In addition, both LCC and $\mathrm{DC_I}$ have the same value for two classes that share the same class model but that exhibit different CPCIs, provided that the two classes have the same number of connected methods, regardless of whether the methods are connected directly or transitively. For example, Classes F and G, shown in Fig. 2, have the same LCC and $\mathrm{DC_I}$ values ($\mathrm{LCC} = \mathrm{DC_I} = 1$), whereas each attribute in Class F is shared by two methods and each attribute in Class G is shared by three methods. In this case, intuition states that Class G is more cohesive than Class F because the attributes in Class G are shared by more methods.

3.6 CC

CC outputs the same value for two classes with the same class model and different CPCIs if the ratio of shared attributes between each pair of methods to the distinct number of attributes referenced by each of the corresponding pair of methods is the same, regardless of the number of shared attributes. For example, Classes J and K, shown in Fig. 2, have the same CC value ($\mathrm{CC} = 1$), whereas the methods in Class J share an attribute and the methods in Class K share all of the attributes. In this case, intuition suggests that Class K is more cohesive than Class J because the methods in Class K share more attributes and because some of the attributes in Class J are not referenced by any method, whereas all the attributes in Class K are referenced by all methods.

3.7 SCOM

SCOM outputs the same value for two classes, as described above, if the SCOM-defined similarity between each pair of corresponding methods has the same value. This is considered an LDA when the two classes have the same SCOM value but one of the classes has additional cohesive interactions between two methods beyond those that exist between the corresponding methods in the other class. For example, counterintuitively, Classes K, L, and M, shown in Fig. 2, have the same SCOM value ($\mathrm{SCOM} = 1$), whereas Class M has more cohesive interactions than Class L, and Class K has more cohesive interactions than each of Classes L and M. In this case, intuition tells us that Class M is more cohesive than Class L because it has more cohesive interactions, and that Class K is the most cohesive class

because it has more cohesive interactions than each of Classes L and M. This LDA occurs because, if there is a pair of methods that reference set of attributes I_1 and I_2 such that $I_2 \subseteq I_1$, the degree of similarity between the two methods remains the same, regardless of the number of attributes included in I_2.

3.8 LSCC

LSCC gives the same value for two classes C_1 and C_2, as described above, in two LDA cases. The first LDA case occurs when each attribute in Class C_1 is referenced by the same number of methods that reference the corresponding attributes in Class C_2, regardless of the distribution of these methods and their connectivity with other methods. This is an LDA in some cases. For example, Classes H and I in Fig. 2 have the same LSCC values ($\mathrm{LSCC} = 0.167$), but all members of class H are directly or indirectly connected, while some of the members of Class I are disjoint. The second LDA case arises when the result of $\sum_{i=1}^{n} x_i(x_i - 1)$ is the same for both classes, where n is the number of attributes and x_i is the number of methods that reference attribute i, regardless of the distributions or the number of references. This is an LDA in some cases. For example, Classes N and O, shown in Fig. 2, have the same LSCC value ($\mathrm{LSCC} = 0.25$), whereas all members in Class O are directly or indirectly connected and some of the members of Class N are disjoint.

3.9 NHD and SNHD

NHD gives the same value for two classes, as described above, for the same first LDA case described above for LSCC. The only difference in this case is that the circles in Fig. 2 for Classes H and I represent distinct parameter types when considering NHD, instead of representing attributes as in the case of LSCC. In addition, the two classes will have the same NHD values if the result of $\sum_{i=1}^{n} x_i(n - x_i)$ is the same for both classes, where n is the number of distinct parameter types and x_i is the number of methods that reference parameter type i, regardless of the distributions or the number of references. This is an LDA in some cases. For example, Classes P and Q, shown in Fig. 2, have the same NHD values ($\mathrm{NHD} = 0.33$), where in Class P, each distinct type is a type of parameter of a method, while in Class Q, each distinct type is a type of parameter of two methods. Notice also that Class P has three disjoint components, whereas all members of Class Q are directly or indirectly connected.

SNHD measures the closeness of the NHD metric to the maximum value of NHD compared to the minimum value. SNHD has the value zero when both the maximum and minimum values of NHD are the same, regardless of the number of distinct parameter types referenced by each method. For example, counterintuitively, Classes L and M, shown in Fig. 2, have the same SNHD value ($\mathrm{SNHD} = 0$), whereas Class M has more cohesive interactions than Class L. In this case, intuition tells us that Class M is more cohesive than Class L because it has more cohesive interactions. This LDA occurs because each of Class L and Class M has the same maximum and minimum NHD values (i.e., the corresponding value for Class L is 0.33 and the corresponding value for Class M is 0.67).

The above LDA cases show that, in some cases, it is an anomaly for a metric to have the same value for two classes of the same model in terms of the number of attributes/parameter-types and number of methods, but different CPCIs. This section reemphasizes the importance of studying the LDA problem. However, it provides no data on how often LDA cases are expected to occur, and it therefore fails to indicate whether the problem deserves consideration. In the following section, we empirically investigate how often this problem occurs using five open source Java systems. Cohesion metrics for which the LDA problem occurs frequently are more likely to be bad cohesion indicators because, in many cases, such metrics cannot distinguish between classes of different cohesion levels.

4 LDA Problem Occurrence: an Empirical Study

We empirically studied the occurrence of the LDA problem when several cohesion metrics are applied to specific Java systems. The selected systems are overviewed and the empirical study settings and results are reported as follows.

4.1 Software Systems and Cohesion Metrics

We chose five Java open source software systems from different domains, including Art of Illusion v.2.5 [32], FreeMind v.0.8.0 [26], GanttProject v.2.0.5 [27], JabRef v.2.3 beta 2 [33], and Openbravo v.0.0.24 [37]. Art of Illusion consists of 488 classes and is a 3D modeling, rendering, and animation studio system. FreeMind consists of 456 classes and is a hierarchical editing system. GanttProject consists of 496 classes and is a project-scheduling application featuring resource management, calendaring, and importing and exporting (MS Project, HTML, PDF, spreadsheets). JabRef consists of 599 classes and is a graphical application for managing bibliographical databases. Openbravo consists of 452 classes and is a point-of-sale application designed for touch screens. We randomly chose these five open source systems from http://sourceforge.net. The restrictions placed on the choice of these systems were that they

1. are implemented using Java,
2. are relatively large in terms of the number of classes,
3. are from different domains, and
4. have available source code.

We developed our own Java tool to automate the cohesion measurement process for Java classes using the 16 cohesion metrics discussed in Section 3. The tool analyzed the Java source code, extracted information required to build the matrices, and calculated cohesion values for 1,673 classes selected from among 2,491 classes from the five open source systems. We excluded all classes for which at least one of the metrics was undefined. For example, classes consisting of single methods were excluded because their TCC, LCC, $\mathrm{DC_D}$, $\mathrm{DC_I}$, CC, SCOM, LCOM5, NHD, and SNHD values were undefined. In addition, classes without any attributes were excluded because their LCOM5, SCOM, Coh, NHD, and SNHD values were undefined. Classes in which none of the methods have parameters are excluded because their CAMC, NHD, and SNHD values were undefined. Excluding all classes that have undefined cohesion values using the

TABLE 3
Descriptive Statistics for Our Chosen Cohesion Measures

Metric	Min	25%	Mean	Med	75%	Max	Std. Dev.
LSCC	0	0.03	0.24	0.10	0.30	1	0.31
SCOM	0	0.06	0.33	0.20	0.50	1	0.34
CC	0	0.06	0.28	0.17	0.38	1	0.31
LCOM1	0	2.00	72.40	12.00	45.00	7875	327.57
LCOM2	0	0.00	55.36	4.00	32.00	7875	305.11
LCOM3	0	1.00	1.25	1.00	1.00	11	0.83
LCOM4	0	1.00	1.19	1.00	1.00	9	0.73
LCOM5	0	0.53	0.71	0.78	0.92	2	0.35
Coh	0	0.18	0.41	0.33	0.58	1	0.29
TCC	0	0.17	0.48	0.44	0.75	1	0.35
LCC	0	0.22	0.59	0.65	1.00	1	0.38
DC_D	0	0.20	0.49	0.46	0.75	1	0.34
DC_I	0	0.28	0.60	0.66	1.00	1	0.37
CAMC	0.01	0.15	0.27	0.23	0.33	1	0.16
NHD	0	0.47	0.56	0.63	0.74	1	0.25
SNHD	-1	-0.33	-0.06	0.00	0.00	1	0.52

metrics under consideration allows us to perform the same analysis for all metrics on the same set of classes and, therefore, compare the respective results in an unbiased manner.

Table 3 lists descriptive statistics for each cohesion measure—including the minimum, 25 percent quartile, mean, median, 75 percent quartile, maximum value, and standard deviation. As indicated by Briand et al., LCOM1, LCOM2, LCOM3, and LCOM4 are not normalized, and they feature extreme outliers due to accessor methods that typically reference single attributes.

4.2 Empirical Study Settings and Results

We developed a tool to analyze the outputs of the class cohesion measuring tool from the perspective of discrimination. The tool groups together sets of classes that have the same model (i.e., number of methods and number of attributes/parameter-types). For each group, the tool counts 1) the distinct cohesion values and 2) the number of classes that have the same cohesion values but different CPCIs. We refer to these classes as the classes with repeated cohesion values. This analysis was performed for each metric separately. As a result, using a cohesion metric, a class is considered to have a repeated cohesion value if another class in the same group has the same cohesion value and a different CPCI. The number of classes with repeated cohesion values indicates how often the LDA problem occurs in the systems being analyzed. The discriminative power of a cohesion metric is expected to increase as 1) the number of distinct cohesion values increases and 2) the number of classes with repeated cohesion values decreases. Hence, these two measures are inversely correlated but are not complementary. This is because distinct cohesion values can be either unique or repeated. A class that has a distinct cohesion value is counted with classes that have repeated cohesion values only when the value is not unique. Table 4 shows the number of classes considered in each application. In addition, it lists the number and percentages of distinct *method-attribute* models and distinct *method-parameter-type* models of these classes for each system alone and for all systems together. For example, Table 4 shows that 394 classes in the Illusion system were considered in the empirical study

TABLE 4
Distinct Models for the Selected Classes

System	No. of considered classes	No. and percentage of distinct method-attribute models	No. and percentage of distinct methods-parameter-types models
Illusion	394	251 (63.7%)	187 (47.5%)
FreeMind	336	117 (34.8%)	120 (35.7%)
Gantt	332	115 (34.6%)	98 (29.5%)
JabRef	296	142 (48.0%)	100 (33.8%)
Openbravo	315	146 (46.3%)	105 (33.3%)
All Systems	**1673**	**437 (26.1%)**	**298 (17.8%)**

TABLE 5
Percentages of Classes with Repeated Cohesion Values

Metric	Illusion	FreeMind	Gantt	JabRef	Openbravo	All Systems
SCOM	14.2	50.6	48.2	38.5	34.0	**45.8**
CC	12.4	51.5	47.3	38.5	36.5	**46.8**
LSCC	13.2	52.7	51.8	40.2	35.2	**50.0**
DC_D	20.8	53.0	53.3	43.2	39.4	**52.5**
LCOM5	15.7	54.8	50.3	39.2	37.8	**52.8**
Coh	15.7	54.8	50.3	39.2	37.8	**52.8**
LCOM1	19.3	54.2	53.9	42.6	37.8	**53.4**
TCC	20.6	54.8	53.3	43.2	40.0	**53.7**
LCOM2	29.4	56.3	59.3	47.0	45.1	**61.2**
DC_I	31.0	55.1	58.7	48.0	44.4	**62.7**
LCC	30.5	57.4	58.7	48.0	44.4	**64.2**
NHD	40.6	61.0	62.3	56.4	60.6	**67.1**
CAMC	41.9	59.2	59.6	55.4	60.6	**69.0**
SNHD	42.4	64.0	70.8	64.2	68.3	**70.8**
LCOM3	49.7	70.5	71.1	60.5	64.4	**77.8**
LCOM4	51.3	71.7	70.8	61.8	66.0	**79.4**

and that these classes have 251 different method-attribute models and 187 different method-parameter-type models. As a result, the percentages of distinct method-attribute and method-parameter-type models out of the total number of classes are 63.7 percent and 47.5 percent, respectively. Table 4 shows that there is no direct relation between the percentage of distinct models and the number of classes in different systems. For example, the Gantt system has a larger number of considered classes and smaller percentages of distinct models than the JabRef system, whereas the Illusion system has a larger number of considered classes and larger percentages of distinct models than the JabRef system. This is because the number of class models in a system can vary widely. The results in Table 4 also show that the number of distinct method-attribute models is generally larger than the number of distinct method-parameter-types models. This is because, when considering classes in systems, we note that the number of attributes in classes tends to be larger than the number of distinct parameter types. When all classes in all systems are considered together, the probability of identifying classes whose method-attribute or method-parameter-type models overlap becomes higher than when each system is considered alone. This means that the percentage of distinct models is expected to become smaller, which is the case in the last row of Table 4.

Table 5 lists the percentage of classes that exhibit repeated cohesion values for each metric and for each application in ascending order, consistent with the results in the last column. This column shows the percentage of classes with repeated cohesion values when classes from all five systems are considered together. Table 6 is similar to Table 5, but the percentages of *distinct cohesion values* for each of the studied metrics are listed in descending order, consistent with the results reported in the last column.

The results in Tables 5 and 6 lead to the following conclusions:

1. Using our analyzed cohesion metrics, the percentages of classes with repeated cohesion values across all systems are relatively high (i.e., they range between 45.8 percent and 79.4 percent). This indicates that the LDA problem deserves attention.
2. As expected, the results in Tables 5 and 6 are inversely correlated; the metrics that show more classes with repeated cohesion values are expected to have fewer distinct cohesion values.
3. SCOM most often shows the lowest percentages of classes with repeated cohesion values, and it exhibits the highest percentages of distinct cohesion values. By contrast, LCOM3 and LCOM4 show the highest percentages of classes with repeated cohesion values and the lowest percentages of distinct cohesion values. Accordingly, SCOM is expected to be the most discriminating metric among those considered, whereas LCOM3 and LCOM4 are expected to be the worst.
4. LCOM1 is associated with a lower percentage of classes with repeated cohesion values and a higher percentage of distinct cohesion values than LCOM2, LCC, and DC_I. As a result, LCOM1 is expected to be more discriminating than LCOM2, LCC, and DC_I. This result is consistent with our discussion in Section 3, where we demonstrate that LCOM2, LCC, and DC_I have more LDA problems than LCOM1.
5. LCOM5 and Coh have the same discrimination results, consistent with our expectations as discussed in Section 3. Note that despite the fact that the CAMC formula is correlated with the formulas for both LCOM5 and Coh, CAMC has different discrimination results because it is applied to a

TABLE 6
The Percentages of Distinct Cohesion Values

Metric	Illusion	FreeMind	Gantt	JabRef	Openbravo	All Systems
SCOM	90.9	57.7	64.8	70.9	73.0	**62.5**
CC	92.1	57.1	64.5	70.9	71.4	**61.6**
LSCC	91.6	56.5	62.7	70.6	72.4	**59.8**
LCOM5	89.8	55.7	63.3	70.9	71.1	**57.9**
Coh	89.8	55.7	63.3	70.9	71.1	**57.9**
DC_D	86.5	57.1	61.4	67.2	69.8	**57.7**
LCOM1	87.8	55.4	59.9	67.2	70.8	**56.8**
TCC	86.8	55.4	60.8	67.2	69.5	**56.8**
LCOM2	81.0	53.9	55.1	63.5	65.7	**49.9**
DC_I	79.9	56.3	56.6	63.5	66.3	**49.7**
LCC	80.2	54.2	56.0	63.5	66.3	**48.5**
NHD	72.1	49.7	50.6	54.1	52.1	**43.4**
CAMC	72.1	51.8	54.2	56.1	53.0	**42.6**
SNHD	70.6	47.3	42.5	48.0	45.4	**38.5**
LCOM3	67.8	41.4	43.4	53.4	49.8	**33.9**
LCOM4	67.0	39.9	43.4	52.4	48.9	**32.5**

method-parameter-type model, whereas LCOM5 and Coh are applied to a method-attribute model.

6. TCC, DC_D, and LCOM1 exhibit very similar discrimination results. This is consistent with our expectations as discussed in Section 3 because these metrics have the same LDA problems. The minor differences between the discrimination results are due to differences in their definitions of cohesive interaction. The same observation applies when comparing the discrimination results of LCOM3 and LCOM4.
7. The discrimination results for the metrics whose definitions of method-method cohesive interactions are more precise (SCOM, CC, and LSCC) are better than those for metrics with less precise definitions (TCC, LCC, DC_D, DC_I, LCOM1, and LCOM2). Note that the latter metrics consider two methods fully cohesive if they share a common attribute, whereas the former differentiates between the cohesions of methods that share more or fewer attributes.

4.3 Threats to Validity

Several factors may restrict the generalizability of our findings and thus limit the interpretation of our results. The first factor is that all five of the systems considered here are implemented in Java. The second is that all of these systems are open source and thus may not be representative of software from the commercial sector, although this is common practice in the research community. The third factor is that although they are not artificial examples, the selected systems are not large in terms of the number and sizes of classes and in terms of covering most of the possible CPCIs. To generalize the results, we would recommend focusing on different systems written in different programming languages, selected from different domains, and that include real-life, large-scale software.

The goal of this empirical study was to investigate whether the LDA problem deserves attention and consideration, which we conclude is true. In addition, the results approximately indicate the comparative discriminative power of each of the cohesion metrics studied. To accurately measure the discriminative power of a cohesion metric, all possible CPCIs of each model have to be accounted for, which is difficult to achieve empirically unless an extremely large number of classes is selected in order to cover all possible CPCIs. In the next section, we solve this problem by simulating all possible CPCIs for our selected class models and using them as a way to measure the discriminative power of each cohesion metric.

5 Measuring Discriminative Power

This section defines the metric that measures the discriminative power of class cohesion metrics and proposes a methodology for applying the defined metric. The methodology is applied to several class models to measure the discriminative power of 16 class cohesion metrics. Finally, limitations of the proposed methodology are discussed.

5.1 Metric Definition

A class model is defined by its number of methods and attributes/parameter-types, regardless of the CPCI. For example, although Classes A and B in Fig. 1 have different CPCIs, they have the same model because they have the same number of methods and attributes. Given a cohesion metric and a class model, the discriminative power of a class cohesion metric (DPC) is the probability that it will obtain different cohesion values for classes of the same model but with different CPCIs. Formally, given a metric m, we consider a model with k methods and l attributes/parameter-types: *DCP(k,l)*, the number of possible distinct

Algorithm: Compute *DPC(m,k,l)*
Inputs: cohesion metric *m*, number of methods *k*, and number of attributes/parameter-types *l*.
Output: The discrimination value *DPC(m,k,l)*.
Steps:
1. Create an empty *unique matrices* list
2. Create an empty *distinct cohesion values* list
3. Create a binary matrix *CPCI* of size *k*×*l*
4. *for* each possible combination of 1's and 0's in the matrix *CPCI* do
   4.1. *for* each matrix *reordered* representing a possible permutation of rows and columns of the matrix *CPCI* do
      4.1.1. *if reordered*∈ *unique matrices* list, *then* go to Step 4 (i.e., consider next combination).
   4.2. Add *CPCI* to *unique matrices* list.
   4.3. Calculate the cohesion value *v* using the metric *m*.
   4.4. *if v*∉ *distinct cohesion values* list, *then* add *v* to the *distinct cohesion values* list.
5. *DPC(m,k,l)*= number of matrices in *unique matrices* list divided by number of values in *distinct cohesion values* list.

Fig. 3. Algorithm to compute DPC.

CPCIs for the class model and *DCV(m,k,l)*, the number of distinct cohesion values when the metric m is applied to all possible distinct CPCIs of the class model. The discrimination metric is then defined as follows:

$$DPC(m,k,l) = \frac{DCV(m,k,l)}{DCP(k,l)}.$$

A highly discriminating cohesion metric is more desirable because the probability of its differentiating between cohesively different classes is high. Discrimination and monotonicity are complementary cohesion properties. Given a cohesion metric, the monotonicity property ensures that if a cohesive interaction is added to the model, the modified model will exhibit a cohesion value that is the same as or higher than the cohesion value of the original model (i.e., the model before adding the cohesive interaction). As a result, a cohesion metric that complies with the monotonicity property and that exhibits a high DPC value has 1) no chance of exhibiting a lower cohesion value, 2) a low probability of obtaining the same cohesion value, and 3) a high probability of obtaining a higher cohesion value for the class after taking into account the added cohesive interaction.

5.2 Measuring the DPC Process

Models with larger numbers of methods and attributes have much larger numbers of possible distinct CPCIs. For example, the model with three methods and three attributes has 36 distinct CPCIs, whereas the model with five methods and five attributes has 5,624 distinct CPCIs. Therefore, if the discrimination metric considers all models together, its value will be dominated by the larger models because they have much larger number of distinct CPCIs. To solve this problem, the DPC of a metric is measured for each model separately. The discriminative power of different metrics can be compared by considering the DPC values for different models. The algorithm in Fig. 3 shows how to measure the DPC value for a specific cohesion metric and a model. The algorithm uses a binary $k \times l$ matrix to represent the CPCI, where k is the number of methods and l is the number of attributes/parameter-types in the class of interest. The matrix has rows indexed by the methods and columns indexed by the attributes/parameter-types, and so for $1 \leq i \leq k$, $1 \leq j \leq l$, we write:

$$m_{ij} = \begin{cases} 1 & \text{if the } i\text{th method references the } j\text{th attribute} \\ & \text{(if the } i\text{th method has a parameter of the } j\text{th type)}, \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm generates matrices that include all possible combinations of ones and zeros. Because the order of the rows and columns of each of the matrices is not important (i.e., the order of methods and attributes in a class is not important), some of the generated matrices represent identical CPCIs. Therefore, each time the algorithm generates a matrix with a combination of ones and zeros, we check each possible permutation of rows and columns against each of the pregenerated matrices contained in a list called *unique matrices*. If no match is detected, the matrix is added to the list of *unique matrices*. Otherwise, the matrix is ignored because it represents a CPCI that has already been accounted for. Fig. 4 shows all possible unique matrices for a matrix of size 2×2. Every other possibility can be made to

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Fig. 4. All possible unique matrices of size 2×2.

match one of these matrices by permuting the rows and columns. When the matrix is added to the list of *unique matrices*, the corresponding class cohesion can be measured and the resulting value is added to a list called the *distinct cohesion values* only if the value does not already exist in the list. This list includes distinct cohesion values for the CPCIs represented in the list of *unique matrices*. Hence, detecting that a cohesion value was already added to the *distinct cohesion values* list suggests the existence of an LDA problem (i.e., several classes have the same cohesion values and the same model but different CPCIs). We implemented the algorithm using a Java-based tool. After generating all possible matrices and building the *unique matrices* list, the tool generated the *distinct cohesion values* list for each of the 16 cohesion metrics under consideration and calculated the DPC value of each of the cohesion metrics. This is expressed as the ratio of the number of values in the *distinct cohesion values* list to the number of matrices in the *unique matrices* list.

5.3 DPC Results

We executed our Java tool on several sizes of matrices; the results are listed in Table 7. The first and second columns of Table 7 show the number of methods and the number of attributes/parameter-types, respectively. We note that only CAMC, NHD, and SNHD use parameter-types, and the rest of the considered metrics use attributes. The third column reports the number of unique CPCIs. In CAMC, NHD, and SNHD, each column must include at least one nonzero entry. This means that the number of possible unique CPCIs for CAMC, NHD, and SNHD is less than that for all other metrics. For example, the third, fifth, sixth, and seventh matrices in Fig. 4 are the only possible unique CPCIs for CAMC, NHD, and SNHD because each of these matrices includes at least one nonzero entry in each column. The number of possible unique CPCIs for CAMC, NHD, and SNHD is reported in the fourth column of Table 7. The other columns list DPC values for the 16 metrics, and the best result for each model is highlighted in boldface. Note that the table does not include any results for models that have one method because most of the considered metrics are undefined in such a case. The DPC value 0.00 reported in Table 7 for some metrics indicates that the resulting value is very small (i.e., the value is less than 0.005 and greater than zero). Due to size constraints, we only show four graphs, in Figs. 5, 6, 7 and 8, that depict the change in DPC values, the number of unique CPCIs, and the number of distinct cohesion values for several metrics. Similar graphs for other models and metrics can be drawn using the results in Table 7. Fig. 5 depicts the change in DPC values for six selected metrics as the number of attributes/parameter-types changes across five methods. Fig. 6 depicts the change in the number of unique CPCIs and number of distinct cohesion values as the number of attributes/parameter-types changes across five methods for six selected cohesion metrics. Fig. 7 depicts the change in DPC values for six selected metrics as the number of methods changes across five attributes/parameter-types. Finally, Fig. 8 depicts the change in the number of unique CPCIs and the number of distinct cohesion values for seven selected cohesion metrics as the number of methods changes across five attributes/parameter-types. Due to the large range of values for the number of unique CPCIs compared to the number of distinct cohesion values, we used a logarithmic scale for the y-axis in Figs. 6 and 8.

The results listed in Table 7 lead to the following conclusions:

1. In most cases, the DPC of a cohesion metric decreases as the size of the model (i.e., number of methods and attributes/parameter-types) increases. This means that mostly the likelihood of LDA cases occurring increases as the size of the model increases. This is due to the fact that mostly the number of unique CPCIs increases much faster than the number of distinct cohesion values as the size of the model increases, as depicted in Figs. 6 and 8. However, this observation does not always hold. For example, for the model with two methods, the DPC values of CC and SCOM increase as the number of attributes increases from six to seven. In addition, for the model with two parameter types, the DPC value of NHD increases as the number of methods increases from 9 to 10.
2. SCOM has the largest DPC value for most of the models considered, which indicates that SCOM is usually the most discriminative cohesion metric among those considered.
3. CAMC and NHD have the largest DPC values among the metrics considered for models with two methods. This is in part due to the relatively low number of possible CPCIs for CAMC and NHD compared to the number of possible CPCIs for the other metrics.
4. CAMC, Coh, and LCOM5 have the highest possible DPC value for models with a single attribute/parameter-type. In this case, each possible CPCI has a different cohesion value.
5. LCOM3 and LCOM4 have the smallest DPC values for all models, suggesting that LCOM3 and LCOM4 are the least discriminative cohesion metrics among those considered. This result is consistent with the results of our previous empirical validation studies [5], [6], which determined that LCOM3 and LCOM4 were not reliable cohesion indicators.
6. When the metrics are ordered consistent with the average DPC value shown in Table 7, this order is usually the same as that of metrics ordered according to their expected discriminative power, as obtained in the empirical study described in Section 4. The main difference is in the order of the CAMC and NHD metrics. For example, CAMC is found to be the third-best metric in terms of average DPC value (Table 7) and the third worst metric in terms of expected discrimination level (Tables 5 and 6). As mentioned previously, CAMC has high-DPC values for models with two methods and for models that have single attributes. We counted the number of classes that conformed to these models; they comprise 19 percent of all considered classes. This means that most of the

TABLE 7
DPC Values for the Considered Metrics

No. of methods	No. of attributes/ parameter-types	No. of unique CPCIs	No. of unique CPCIs for CAMC, NHD, & SNHD	DPC values										
				SCOM	CC	LSCC	Coh, LCOM5	LCOM1, TCC, DC_D	LCOM2	LCC, DC_I	NHD	SNHD	CAMC	LCOM3, LCOM4
2	1	3	2	0.67	0.67	0.67	**1.00**	0.67	0.67	0.67	**1.00**	**1.00**	**1.00**	0.33
2	2	7	4	0.43	0.43	0.43	0.71	0.29	0.29	0.29	**0.75**	0.50	**0.75**	0.29
2	3	13	6	0.38	0.38	0.31	0.54	0.15	0.15	0.15	**0.67**	0.33	**0.67**	0.15
2	4	22	9	0.32	0.32	0.23	0.41	0.09	0.09	0.09	**0.56**	0.22	**0.56**	0.09
2	5	34	12	0.35	0.32	0.18	0.32	0.06	0.06	0.06	**0.50**	0.17	**0.50**	0.06
2	6	50	16	0.30	0.26	0.14	0.26	0.04	0.04	0.04	**0.44**	0.13	**0.44**	0.04
2	7	70	20	0.34	0.27	0.11	0.21	0.03	0.03	0.03	**0.40**	0.10	**0.40**	0.03
2	8	95	25	0.32	0.24	0.09	0.18	0.02	0.02	0.02	**0.36**	0.08	**0.36**	0.02
2	9	125	30	0.33	0.23	0.08	0.15	0.02	0.02	0.02	**0.33**	0.07	**0.33**	0.02
2	10	161	36	0.30	0.20	0.07	0.13	0.01	0.01	0.01	**0.31**	0.06	**0.31**	0.01
3	1	4	3	0.75	0.75	0.75	**1.00**	0.75	0.75	0.75	0.67	0.67	**1.00**	0.25
3	2	13	9	0.54	0.38	0.46	0.54	0.31	0.23	0.23	0.33	0.22	**0.56**	0.15
3	3	36	23	**0.42**	0.33	0.25	0.28	0.11	0.08	0.08	0.17	0.09	0.30	0.08
3	4	87	51	**0.31**	0.26	0.14	0.15	0.05	0.03	0.03	0.10	0.04	0.18	0.03
3	5	190	103	0.29	**0.30**	0.08	0.08	0.02	0.02	0.02	0.06	0.02	0.11	0.02
3	6	386	196	**0.23**	0.21	0.05	0.05	0.01	0.01	0.01	0.04	0.02	0.07	0.01
3	7	734	348	0.23	**0.27**	0.03	0.03	0.01	0.00	0.00	0.02	0.01	0.04	0.00
3	8	1324	590	0.21	**0.25**	0.02	0.02	0.00	0.00	0.00	0.02	0.01	0.03	0.00
3	9	2284	960	0.21	**0.25**	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.02	0.00
4	1	5	4	0.80	0.80	0.80	**1.00**	0.80	0.60	0.80	0.75	0.50	**1.00**	0.20
4	2	22	17	**0.55**	0.41	0.41	0.41	0.32	0.18	0.23	0.35	0.12	0.41	0.09
4	3	87	65	**0.37**	0.29	0.17	0.15	0.08	0.05	0.06	0.15	0.03	0.15	0.03
4	4	317	230	**0.23**	0.17	0.07	0.05	0.02	0.01	0.02	0.06	0.01	0.06	0.01
4	5	1053	736	0.17	**0.18**	0.03	0.02	0.01	0.00	0.00	0.02	0.01	0.02	0.00
4	6	3250	2197	**0.11**	0.07	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00
4	7	9343	6093	0.09	**0.11**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	1	6	5	0.83	0.83	0.83	**1.00**	0.83	0.67	0.83	0.60	0.40	**1.00**	0.17
5	2	34	28	**0.53**	0.38	0.41	0.32	0.32	0.18	0.21	0.21	0.07	0.32	0.06
5	3	190	156	**0.29**	0.23	0.14	0.08	0.06	0.03	0.04	0.06	0.02	0.08	0.02
5	4	1053	863	**0.14**	0.09	0.03	0.02	0.01	0.01	0.01	0.01	0.01	0.02	0.00
5	5	5624	4571	**0.07**	0.07	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	6	28576	22952	**0.03**	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	1	7	6	0.86	0.86	0.86	**1.00**	0.86	0.71	0.86	0.67	0.33	**1.00**	0.14
6	2	50	43	**0.52**	0.38	0.38	0.26	0.32	0.18	0.18	0.23	0.05	0.26	0.04
6	3	386	336	**0.22**	0.18	0.09	0.05	0.04	0.02	0.02	0.06	0.01	0.05	0.01
6	4	3250	2864	**0.08**	0.04	0.02	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00
6	5	28576	25326	**0.02**	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	1	8	7	0.88	0.88	0.88	**1.00**	0.88	0.75	0.88	0.57	0.29	**1.00**	0.13
7	2	70	62	**0.51**	0.36	0.36	0.21	0.31	0.17	0.17	0.15	0.03	0.21	0.03
7	3	734	664	**0.16**	0.13	0.07	0.03	0.03	0.02	0.02	0.02	0.01	0.03	0.00
7	4	9343	8609	**0.04**	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	1	9	8	0.89	0.89	0.89	**1.00**	0.89	0.67	0.89	0.63	0.25	**1.00**	0.11
8	2	95	86	**0.49**	0.37	0.34	0.18	0.31	0.16	0.17	0.17	0.02	0.17	0.02
8	3	1324	1229	**0.12**	0.10	0.05	0.02	0.02	0.01	0.01	0.03	0.01	0.02	0.00
9	1	10	9	0.90	0.90	0.90	**1.00**	0.90	0.70	0.90	0.56	0.22	**1.00**	0.10
9	2	125	115	**0.50**	0.34	0.31	0.15	0.30	0.15	0.15	0.12	0.03	0.15	0.02
9	3	2284	2159	**0.09**	0.08	0.04	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.00
10	1	11	10	0.91	0.91	0.91	**1.00**	0.91	0.73	0.91	0.60	0.20	**1.00**	0.09
10	2	161	150	**0.47**	0.34	0.29	0.13	0.29	0.15	0.14	0.13	0.03	0.13	0.01
Average				**0.38**	0.34	0.27	0.31	0.23	0.18	0.20	0.26	0.13	0.34	0.06

considered classes (81 percent) conform to models for which CAMC has relatively low-DPC values, which explains why CAMC was expected to be the third-worst metric in the empirical study. This case demonstrates one of the threats to the validity of the empirical study, as indicated in Section 4.3.

5.4 Correlation Analysis

In a previous study [5], [6], we empirically investigated the validity of the same 16 metrics as cohesion indicators. The study used the receiver operating characteristic (ROC) curve [29] to indicate the merit of each metric in predicting faulty classes. We investigated the correlation between the

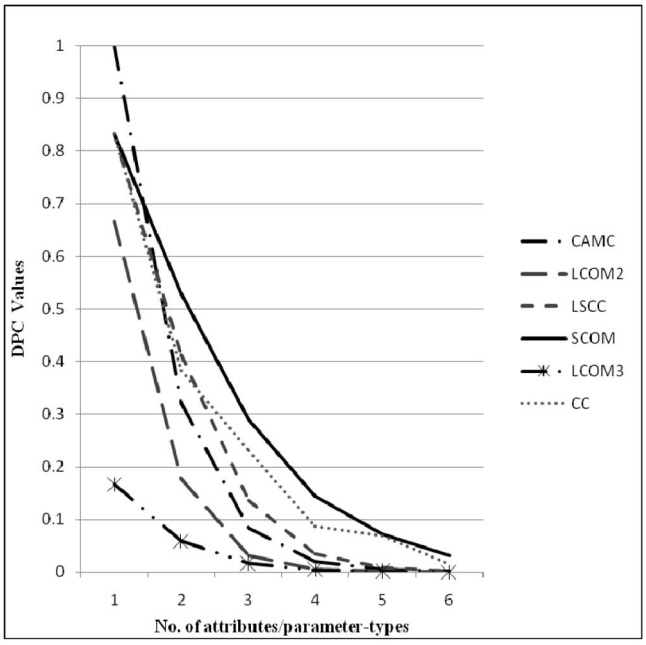

Fig. 5. The change in DPC values as the number of attributes/parameter-types changes (Number of methods = 5).

empirical validation results obtained previously and the discriminative power results obtained here. Table 8 shows the resulting Nonparametric Spearman correlations [38] among the ROC area results obtained in the previous study, the average DPC values obtained here, the percentages of classes with repeated cohesion values (the rightmost column of Table 5), and the percentages of distinct cohesion values (the rightmost column of Table 6). They are all statistically significant ($\text{p-value} < 0.001$).

Interestingly, Table 8 shows that there is a strong correlation (0.85) between the ROC area and the average DPC, which indicates that the metrics that produce high average DPC values predict faulty classes better than do those that produce low average DPC values. In addition, the

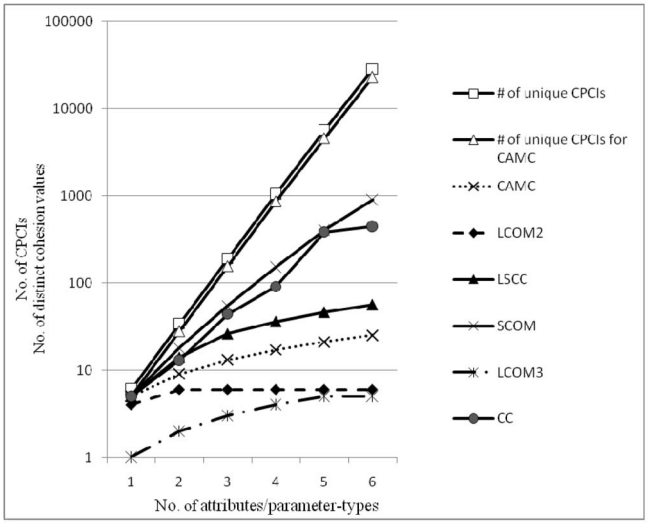

Fig. 6. The change in the number of unique CPCIs and the number of distinct cohesion values (logarithmic scale) as the number of attributes/parameter-types changes (Number of methods = 5).

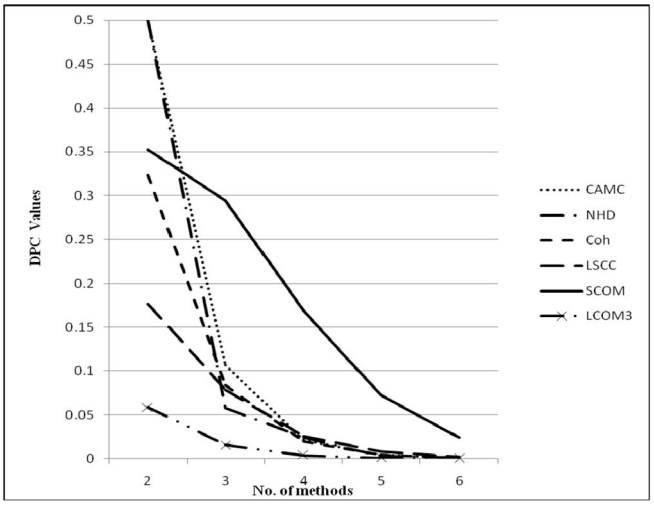

Fig. 7. The change in DPC values as the number of methods changes (Number of attributes/parameter-types = 5).

correlation between the percentages of classes with repeated cohesion values and the percentages of distinct cohesion values is strong (-0.99) and its coefficient sign confirms the second result obtained in Section 4.2 (i.e., the metrics that show more classes with repeated cohesion values are expected to have fewer distinct cohesion values). The other pairs of considered variables are moderately intercorrelated.

5.5 Limitations

Given a cohesion metric and a class model, our DPC measuring methodology obtains the exact DPC value because it considers all possible CPCIs. However, the DPC metric and its measuring methodology have several limitations:

1. The DPC metric can be used to compare the discriminative power of different cohesion metrics. No threshold is introduced to assess the fitness of a cohesion metric in terms of its discriminative power. This issue is left open for further research.

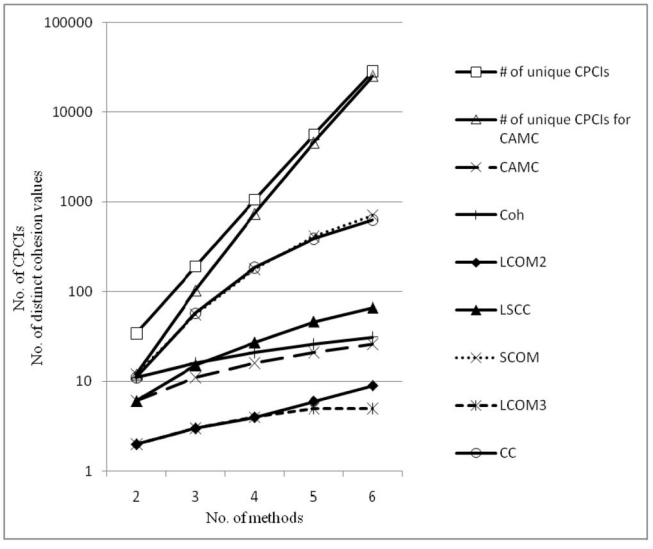

Fig. 8. The change in the number of unique CPCIs and number of distinct cohesion values (logarithmic scale) as the number of methods changes (Number of attributes/parameter-types = 5).

TABLE 8
Spearman Rank Correlations between the Empirical Validation and Discriminative Power Results

Variables	Average DPC	Percentages of classes with repeated cohesion values	Percentages of distinct cohesion values
ROC area	0.85	-0.78	0.82
Average DPC	1	-0.73	0.75
Percentages of classes with repeated cohesion values		1	-0.99

2. The DPC metric is model-dependent. Given a cohesion metric, the models differ greatly in terms of DPC values, as shown in Table 7. In some cases, a cohesion metric has higher DPC values than another metric for some models but it has lower DPC values than the same metric for other models. For example, as shown in Table 7, SCOM has higher DPC values than CC for some models and lower DPC values than CC for other models, which makes it difficult to decide which of the two is generally more discriminating. There are two possible solutions to this problem. The first solution is to find the average DPC using certain models. The solution works for some metrics because their DPC values decrease greatly as the size of the model increases. In this case, the number of models for which significant DPC values are produced is limited. However, this solution is not appropriate for other metrics that produce significant DPC values for an unlimited number of models. For example, Coh and LCOM5 produce a DPC value of 1 for any model that has a single attribute, regardless of the number of methods. Another problem is that, in some circumstances, such as in the case of CAMC, the average DPC is dominated by special case models (i.e., the cases in which the models have a single parameter type or two methods). These two problems can be solved by discarding the outlier values that represent special or insignificant cases and using the average DPC values, considering the remaining models as a basis for measuring the overall discriminative power of a given cohesion metric.
3. The algorithm illustrated in Fig. 3 is computationally intensive. The number of possible CPCIs increases exponentially as the size of the model increases. Given k methods and l attributes, the number of possible CPCIs with n cohesive interactions can be calculated as follows: $\binom{kl}{n} = \frac{kl!}{n!(kl-n)!}$, and therefore the total number of possible CPCIs for all possible number of cohesive interactions is $\sum_{n=0}^{kl} \binom{kl}{n} = 2^{kl}$. To check whether the CPCI is distinct, all different permutations of the methods and attributes are generated. Given a model that features k methods and l attributes, the number of different permutations of attributes and methods is the product of the number of permutations of k and the number of permutations of l, which equals $k!l!$. Each time a permutation is generated, it is compared with each of the previously generated unique CPCIs. As a result, the time complexity of the algorithm shown in Fig. 3 is bounded by $k!.l!.2^{2kl}$. Note that the number of distinct CPCIs for models of relatively large size is much smaller than the number of possible CPCIs. For example, the percentage of the distinct number of CPCIs for a model with five methods and five attributes out of the total number of possible CPCIs is $(5{,}624/2^{25}) \times 100 = 0.017\%$.

We ran our DPC measuring tool on a Pentium 4, 3.4 GHz PC and it took approximately two weeks to output the distinct CPCIs for a model of size 5×6. We propose two approaches to solving this problem. The first method is to introduce an algorithm that generates the distinct CPCIs directly, instead of calculating all possible CPCIs and finding the distinct ones among them. The second approach is to propose a mathematical model to compute the DPC in terms of the probability of outputting the same cohesion value for different CPCIs, given a cohesion metric and a model. These two solutions will be the subject of further research.

6 Conclusions and Future Work

This paper introduced a method to measure the discriminative power of class cohesion metrics. First, the discrimination problem was explained and the cases in which 16 selected class cohesion metrics suffered from the lack of discrimination anomaly problem were listed and discussed. The occurrence of the LDA problem in five open source Java systems was empirically studied. Results showed that the LDA problem deserved attention and consideration. In addition, results indicated the expected ranking of the 16 class cohesion metrics in terms of discriminative power. A metric was then defined and a measuring process was proposed and applied to precisely measure the discriminative power of the sixteen class cohesion measures. The results show that the considered metrics are ordered according to their discriminative power from the strongest to the weakest as follows: SCOM, CC, CAMC, (Coh and LCOM5), LSCC, NHD, (LCOM1, TCC, and $\mathrm{DC_D}$), (LCC and $\mathrm{DC_I}$), LCOM2, SNHD, and (LCOM4 and LCOM3), where the metrics listed between parentheses have the same discriminative power. When ranked according to discriminative power, the exact results for the cohesion metrics were mostly similar to those expected, as derived from the empirical study. Note that a metric can have a high discriminative power, but it violates other mathematical properties of class cohesion metrics and vice versa. Discrimination is a complementary mathematical property that can be used with other existing properties to provide a supportive underlying theory for class cohesion metrics. The class cohesion metric that has a relatively high-discriminative power and satisfies the other properties is expected to be a well-defined metric. On the other hand, the class cohesion metric that exhibits relatively low-discriminative power or violates any of the other necessary properties is expected to be ill-defined and its use as a cohesion indicator is questionable.

We suggest using the discrimination metric whenever a new class cohesion metric is defined and comparing the discriminative power of the new metric with those measured in this study. If the newly introduced metric is found to have relatively low-discriminative power, the researcher may decide to revise the metric definition to solve the discrimination weaknesses. Software developers are advised to apply cohesion metrics with relatively high-discriminative power because such metrics are expected to be more reliable in indicating cohesion.

Our research does not aim to measure the discriminative power of all class cohesion metrics. Instead, our goal was to introduce a metric and a methodology to measure discriminative power and to show how to apply this framework to several class cohesion metrics. The same metric and methodology can be applied to measure the discriminative power of other predefined or newly proposed class cohesion metrics. In addition, the same concept can be applied to compare the discriminative power of metrics used for other quality attributes such as coupling, reusability, and maintainability. As discussed earlier, a better algorithm in terms of time complexity is required to speed up the discrimination measuring process. In addition, a model needs to be introduced to mathematically measure discriminative power. That is, given a cohesion metric and a model, a mathematical formula would be required to calculate the probability that the cohesion metric would lead to different cohesion values for different CPCIs. Alternatively, the mathematical formula could inversely calculate the probability that the cohesion metric would obtain the same cohesion values for different CPCIs of the same model.

In this paper, we have studied the correlation between discriminative power and empirical validation results. Our conclusion is that they are strongly correlated. In the future, we plan to study the correlation between other theoretical properties and empirical validation results. Empirically validating class cohesion metrics is a time-consuming and labor-intensive process, but theoretically validating these metrics is often much easier. If the correlation between theoretical and empirical results is strong, a researcher can be confident that when a metric is found to be theoretically valid, it will be empirically valid as well.

ACKNOWLEDGMENTS

The author would like to acknowledge the support of this work by Kuwait University Research Grant WI03/07. In addition, the author would like to thank Professor Steve Counsell for his insightful comments that helped improve the paper and Saqiba Sulman for assisting in collecting the required data.

REFERENCES

[1] K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Investigating Effect of Design Metrics on Fault Proneness in Object-Oriented Systems," *J. Object Technology,* vol. 6, no. 10, pp. 127-141, 2007.

[2] J. Al Dallal, "A Design-Based Cohesion Metric for Object-Oriented Classes," *Proc. Int'l Conf. Computer and Information Science and Eng.,* Nov. 2007.

[3] J. Al Dallal, "Software Similarity-Based Functional Cohesion Metric," *IET Software,* vol. 3, no. 1, pp. 46-57, 2009.

[4] J. Al Dallal, "Mathematical Validation of Object-Oriented Class Cohesion Metrics," *Int'l J. Computer Science,* vol. 4, no. 2, pp. 45-52, 2010.

[5] J. Al Dallal and L. Briand, "An Object-Oriented High-Level Design-Based Class Cohesion Metric," *Information and Software Technology,* vol. 52, no. 12, pp. 1346-1361, 2010.

[6] J. Al Dallal and L. Briand, "A Precise Method-Method Interaction-Based Cohesion Metric for Object-Oriented Classes," *ACM Trans. Software Eng. and Methodology,* vol. 20, no. 6, Nov. 2011.

[7] M. Alshayeb and W. Li, "An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes," *IEEE Trans. Software Eng.,* vol. 29, no. 11, pp. 1043-1049, 2003.

[8] L. Badri and M. Badri, "A Proposal of a New Class Cohesion Criterion: An Empirical Study," *J. Object Technology,* vol. 3, no. 4, pp. 145-159, 2004.

[9] J. Bansiya, L. Etzkorn, C. Davis, and W. Li, "A Class Cohesion Metric for Object-Oriented Designs," *J. Object-Oriented Programming,* vol. 11, no. 8, pp. 47-52, 1999.

[10] J.M. Bieman and B. Kang, "Cohesion and Reuse in an Object-Oriented System," *Proc. Symp. Software Reusability,* pp. 259-262, 1995.

[11] J. Bieman and L. Ott, "Measuring Functional Cohesion," *IEEE Trans. Software Eng.,* vol. 20, no. 8, pp. 644-657, Aug. 1994.

[12] C. Bonja and E. Kidanmariam, "Metrics for Class Cohesion and Similarity between Methods," *Proc. 44th Ann. ACM Southeast Regional Conf.,* pp. 91-95, 2006.

[13] L.C. Briand, J. Daly, and J. Wuest, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," *Empirical Software Eng.—An Int'l J.,* vol. 3, no. 1, pp. 65-117, 1998.

[14] L.C. Briand, S. Morasca, and V.R. Basili, "Defining and Validating Measures for Object-Based High-Level Design," *IEEE Trans. Software Eng.,* vol. 25, no. 5, pp. 722-743, Sept./Oct. 1999.

[15] L.C. Briand and J. Wust, "Empirical Studies of Quality Models in Object-Oriented Systems," *Advances in Computers,* pp. 97-166, Academic Press, 2002.

[16] L.C. Briand, J. Wust, J. Daly, and V. Porter, "Exploring the Relationship between Design Measures and Software Quality in Object-Oriented Systems," *J. System and Software,* vol. 51, no. 3, pp. 245-273, 2000.

[17] L.C. Briand, J. Wüst, and H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," *Empirical Software Eng.,* vol. 6, no. 1, pp. 11-58, 2001.

[18] 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.

[19] Z. Chen, Y. Zhou, and B. Xu, "A Novel Approach to Measuring Class Cohesion Based on Dependence Analysis," *Proc. Int'l Conf. Software Maintenance,* pp. 377-384, 2002.

[20] Z. Chen, B. Xu, and Y. Zhou, "Measuring Class Cohesion Based on Dependence Analysis," *J. Science and Technology,* vol. 19, no. 6, pp. 859-866, 2004.

[21] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object-Oriented Design," *Object-Oriented Programming Systems, Languages, and Applications,* special issue of SIGPLAN Notices, vol. 26, no. 10, pp. 197-211, 1991.

[22] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Trans. Software Eng.,* vol. 20, no. 6, pp. 476-493, June 1994.

[23] S. Counsell, S. Swift, and J. Crampton, "The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design," *ACM Trans. Software Eng. and Methodology,* vol. 15, no. 2, pp. 123-149, 2006.

[24] N. Fenton and S. Pfleeger, *Software Metrics A Rigorous and Practical Approach,* second ed. ITP, 1997.

[25] L. Fernández and R. Peña, "A Sensitive Metric of Class Cohesion," *Int'l J. Information Theories and Applications,* vol. 13, no. 1, pp. 82-91, 2006.

[26] FreeMind, http://freemind.sourceforge.net/, Aug. 2009.

[27] GanttProject, http://sourceforge.net/projects/ganttproject/, Aug. 2009.

[28] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," *IEEE Trans. Software Eng.,* vol. 3, no. 10, pp. 897-910, Oct. 2005.

[29] J.A. Hanley and B.J. McNeil, "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve," *Radiology,* vol. 143, no. 1, pp. 29-36, 1982.

[30] B. Henderson-Sellers, *Object-Oriented Metrics Measures of Complexity.* Prentice-Hall, 1996.
[31] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object Oriented Systems," *Proc. Int'l Symp. Applied Corporate Computing,* pp. 25-27, 1995.
[32] Illusion, http://sourceforge.net/projects/aoi/, Aug. 2009.
[33] JabRef, http://sourceforge.net/projects/jabref/, Aug. 2009.
[34] B. Kitchenham, S.L. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation," *IEEE Trans. Software Eng.,* vol. 21, no. 12, pp. 929-944, Dec. 1995.
[35] W. Li and S.M. Henry, "Maintenance Metrics for the Object Oriented Paradigm," *Proc. First Int'l Software Metrics Symp.,* pp. 52-60, 1993.
[36] A. Marcus, D. Poshyvanyk, and R. Ferenc, "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems," *IEEE Trans. Software Eng.,* vol. 34, no. 2, pp. 287-300, Mar./Apr. 2008.
[37] Openbravo, http://sourceforge.net/projects/openbravopos, Aug. 2009.
[38] S. Siegel and J. Castellan, *Nonparametric Statistics for the Behavioral Sciences,* second ed. McGraw-Hill, 1988.
[39] 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, pp. 167-180, 2005.
[40] B. Xu and Y. Zhou, "Comments on 'A Cohesion Measure for Object-Oriented Classes,'" *Software—Practice and Experience,* H.S. Chae, Y.R. Kwon, and D.H. Baevol, eds., vol. 31, no. 14, pp. 1381-1388, 2001.
[41] Y. Zhou, L. Wen, J. Wang, Y. Chen, H. Lu, and B. Xu, "DRC: A Dependence Relationships Based Cohesion Measure for Classes," *Proc. 10th Asia-Pacific Software Eng. Conf.,* pp. 1-9, 2003.

Jehad Al Dallal received the BSc (1995) and MSc (1997) degrees in computer engineering from Kuwait University. In 2003, he received the PhD degree in computer science in the area of software engineering from the University of Alberta in Canada. He is currently an assistant professor in the Department of Information Sciences, Kuwait University. He has completed several research projects in the areas of software testing, software metrics, and communication protocols. In addition, he was involved in developing more than 20 software systems. He has also served as a technical committee member for several international conferences.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.