*2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)*
*Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore*

# Towards Cohesion-based Metrics as Early Quality Indicators of Faulty Classes and Components

Chuan Ho Loh [1] and Sai Peck Lee [1+]

[1] Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
ervinloh@perdana.um.edu.my, saipeck@um.edu.my

**Abstract.** Measuring structural design properties of an object-oriented system is a promising approach towards early quality assessments. In object-oriented systems, cohesion is an important factor of object-oriented design quality. A few researchers refer cohesion to the degree of the relatedness of the members in a class. In an object-oriented system, classes and components are key early artifacts that lay the foundation of an object-oriented system. A few cohesion metrics have been proposed to quantify the cohesiveness of classes in an object-oriented system. In this paper, we attempt to quantify the amount of cohesion in classes and components via a suite of object-oriented design metrics. This paper proposes four object-oriented design metrics to evaluate cohesion at the class and component level. The metrics are augmented based on different definitions of LCOM. The metrics are normalized to produce values in the range [0..1], thus yielding more meaningful values than other cohesion metrics such as LCOM1 and LCOM4. The proposed metrics attempt to evaluate whether an artifact (i.e. class or component) represents one abstraction (good) or multiple abstractions (bad). If the artifact represents multiple abstractions, it should be split up into multiple artifacts (i.e. classes and components).

**Keywords:** cohesion, coupling, design metrics, object-orientation

## 1. Introduction

In an object-oriented system, a metric is the measurement of a particular characteristic of a software artifact. Metrics provide a valuable and objective insight into specific ways of enhancing each of the software quality characteristics. Object-oriented design metrics play an important role as early predictors of faulty classes and components in an object-oriented system. Object-oriented design metrics also provide a valuable and objective insight into specific ways of improving each of the quality-carrying characteristics such as maintainability, testability, and reusability. Cohesion is the degree to which an artifact (i.e. class or component) works well as a unit, while coupling is the degree to which artifacts (i.e. classes or components) are interdependent. High cohesion is a desirable property of classes and components. It is widely recognized that highly cohesive components tend to have high quality-carrying attributes [13-14]. High cohesion and low coupling are considered indications of a good design. In an object-objected system, a class or a component is cohesive when its parts are highly correlated. A class or a component with low cohesion has disparate and non-related members. The early focus on the quality of classes and components via the measurement of the structure quality of an object-oriented system helps to identify faulty classes and components at early stages. There are several different approaches to measure cohesion in object-oriented systems. Based on the underlying mechanisms used to measure the cohesion of a class, structural metrics is

[+] Corresponding author. Tel.: 016-2965 111; fax: 03-92875926
*E-mail address*: ervinloh@yahoo.com

the most popular class of cohesion metrics [15]. The major class of structural metrics investigated includes LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, TCC, and LCC [13][15].

## 2. Motivation

In object-oriented systems, classes and components are the fundamental units. The members of a class are the attributes and methods. Similarly, the members of a component are the classes. As such, class and component cohesion refers to the relatedness of the class and component members respectively. The degree of class cohesion gives an indication for the quality of class design; where a highly cohesive class is well designed and a lowly cohesive class is poorly designed. Most of the existing approaches to cohesion measurement in object-oriented systems are based on the structural information of the object-oriented design such as attribute references in methods. These measures reflect particular interpretations of cohesion and attempt to capture different aspects of cohesion. A large number of object-oriented design metrics have been proposed such as DIT, WMC, NOC, LCOM1, and LCOM3 [1-2]. A particular emphasis has been given to the measurement of object-oriented design artifacts as early quality indicators of object-oriented systems. An object-oriented design metric to evaluate class cohesion is important in the object-oriented paradigm because cohesion gives an indication of a good design of classes and components in an object-oriented system. Cohesiveness of methods within a class is desirable, since it promotes encapsulation. Lack of cohesion implies classes should probably be split into two or more subclasses. Any measure of disparateness of methods helps identify flaws in the design of classes. The dominating philosophy behind existing metrics is either class variable referencing or sharing between methods as contributing to the degree to which the methods of a class belong together [15]. In this research, we attempt to objectively and quantitatively measure cohesion at the class and component level via class variable referencing.

## 3. Related Work

Cohesion is considered amongst the most important properties to evaluate the quality of an object-oriented design. Many definitions of object-oriented cohesion metrics have been proposed and most of the metrics are based on the notion of degree of similarity of methods [1-10]. Most of the proposed metrics can be classified into object-oriented design-based metrics. Object-oriented design-based class cohesion metrics require inspecting high-level object-oriented design artifacts such as method interfaces to estimate the class cohesion. In object-oriented systems, cohesion is usually measured at the class level. Most of the proposed metrics are based on either sharing of instance variables or instance variables usage [12]. These metrics usually capture class cohesion in terms of connections among members within a class. The computation mechanism is usually based on either the number of instance variables used by methods or the number of methods pairs that share instance variables [1-10]. A class is more cohesive when a larger number of its instance variables are referenced by a method (i.e. LCOM5), or a larger number of methods pairs share instance variables (LCOM1, LCOM2, LCOM3, LCOM4, TCC, and LCC) [1-11]. LCOM is one of the significant object-oriented design metrics that can be used to evaluate object-oriented systems. LCOM2, LCOM3, LCOM4, and LCOM5 metrics are several minor improvements of LCOM1 metric. Using these different definitions of LCOM results in different measured LCOM values. Most of the existing definitions of LCOM metrics believe class variable referencing between methods as contributing to the degree to which the methods of a class belong together as the dominating viewpoint. As such, most metrics on cohesion measure relationships among the methods of a class based on this hypothesis. The major differences among the different definitions of LCOM metrics are based on the definition of the relationships among methods and the computation mechanisms.

## 4. Proposed Cohesion Metrics

Cohesion metrics measure the extent to which the methods of a class are related to each other. A cohesive class performs one function. A non-cohesive class performs two or more unrelated functions. A non-cohesive class may need to be restructured into two or more smaller classes. The general hypothesis behind the proposed cohesion metrics is that methods are related if they work on the same class-level variables. Methods are unrelated if they work on different variables altogether. In this paper, we refer

cohesion as an indication of whether an artifact (i.e. class or component) represents a single abstraction or multiple abstractions. The general idea is that if an artifact represents more than one abstraction, then the artifact should be refactored into two or more artifacts, each of which represents a single abstraction. The notations used in the metrics and the properties of the metrics are described in Table 1 and Table 2.

## 4.1. Class-based Cohesion Metrics

We propose two class-based cohesion metrics, namely Class Cohesion Index (CsCohI) and Lack of Class Cohesion Index (LCsCohI) to measure the overall density of similarity and dissimilarity of methods in a class. CsCohI measures the degree of similarity of methods in a class. The CsCohI within a class Cs is expressed as:

$$CsCohI(Cs) = \begin{cases} \frac{TCohM}{TM}, TCohM > 0 \\ 0, otherwise \end{cases}$$

Each method within a class accesses one or more attributes (i.e. instance variables). CsCohI is the number of methods that access one or more of the same attributes. If no methods access at least one attribute, then CsCohI = 0. In general, low values for CsCohI imply that the class might be better designed by breaking it into two or more separate classes. Although there are cases in which a low value for CsCohI is justifiable, it is desirable to keep CsCohI high (i.e. keep cohesion high). CsCohI < 1 indicates that the class is not quite cohesive and may need to be refactored into two or more classes. Classes with a low CsCohI can be fault-prone. A low CsCohI value indicates scatter in the functionality provided by the class. CsCohI is expressed as a nondimensional value in the range of $0 \leq CsCohI \leq 1$. Similarly, the overall degree of dissimilarity of methods within a class Cs, LCsCohI(Cs), is expressed as:

$$LCsCohI(Cs) = \begin{cases} 1 - \frac{TCohM}{TM}, TCohM > 0 \\ 0, otherwise \end{cases}$$

## 4.2. Component-based Cohesion Metrics

We propose two component-based cohesion metrics, namely Component Cohesion Average (CoCohA) and Lack of Component Cohesion Average (LCoCohA) to measure the overall density of similarity and dissimilarity of methods in the classes of within a component. CoCohA measures the degree of class cohesion indexes in a component. The CoCohA within a component Co is expressed as:

$$CoCohA(Co) = \begin{cases} \frac{TCsCohI}{TC}, TCsCohI > 0 \\ 0, otherwise \end{cases}$$

CoCohA is defined in an analogous manner and provides an indication of the overall degree of similarity of methods in the classes within a component. The CoCohA numerator is the sum of class cohesion indexes in a component, TCsCohI. The CoCohA denominator is the total classes in a component. The CoCohA numerator represents the maximum number of similarity of method situations in the classes for a component. CoCohA is expressed as a nondimensional value in the range of $0 \leq CoCohA \leq 1$. In general, a low value for CoCohA indicates a low proportion of class cohesion indexes in a component, and a high value for CoIA indicates a high proportion of class cohesion indexes in a component. A low value for CoCohA is undesirable. Similarly, the overall degree of dissimilarity of methods in the classes within a component Co, LCoCohA(Co), is expressed as:

$$LCoCohA(Co) = \begin{cases} 1 - \frac{TCsCohI}{TC}, TCsCohI > 0 \\ 0, otherwise \end{cases}$$

LCsCohI and LCoCohA are inverse metrics of CsCohI and CoCohA respectively. A high value of CsCohI, and CoCohA, and a low value of LCsCohI and LCoCohA indicate high cohesion and well-designed class and component. Similarly, a low value of CsCohI, and CoCohA, and a high value of LCsCohI and LCoCohA indicate low cohesion and poorly designed class and component. It is likely that the class and component have good subdivision. A cohesive class tends to provide a high degree of encapsulation. A lower value of CsCohI and CoCohA indicate decreased encapsulation and increased complexity, thereby increasing

the likelihood of errors. Similarly, a higher value of LCsCohI and LCoCohA indicate increased encapsulation and decreased complexity, thereby decreasing the likelihood of errors.

TABLE 3 NOTATIONS OF CSCOHI, COCOHA, LCSCOHI, LCOCOHA

| Notation | Description | C++ | Java |
|---|---|---|---|
| CsCohI | class cohesion index within a class | - | - |
| CoCohA | component cohesion average within a component | - | - |
| LCsCohI | lack of class cohesion index within a class | - | - |
| LCoCohA | lack of component cohesion average within a component | - | - |
| TC | total number of classes in a component | total number of classes in a directive | total number of classes in a package |
| TCsCohI | total of class cohesion indexes in a component | - | - |
| TCohM | methods declared and inherited in a class | all function members declared and inherited in a class excluding virtual (deferred) ones assessing at least one instance variable | all methods declared and inherited in a class excluding abstract (deferred) ones assessing at least one instance variable |
| TM | methods declared and inherited in a class | all function members declared and inherited in a class excluding virtual (deferred) ones | all methods declared and inherited in a class excluding abstract (deferred) ones |

TABLE 3 PROPERTIES OF CSCOHI, COCOHA, LCSCOHI, LCOCOHA

| **Properties for CsCohI and LCsCohI** | |
|---|---|
| Property 1 | If n is a non-negative number, then there is only a finite number of cohesive methods (i.e. number of methods assessing at least one instance variable) TCohM for which TCohM = n. |
| Property 2 | If n is a non-negative number, then there is only a finite number of attributes declared and inherited in a class TCsA for which TCsA = n. |
| Property 3 | If n is a non-negative number, then there is only a finite number of methods declared and inherited in a class TCsM for which TCsM = n. |
| Property 4 | There are distinct classes Cs1 and Cs2 for which Cs1 is the superclass of Cs2. |
| Property 5 | There are inherited attributes, $IA_1$, $IA_2$, $IA_3$,…, $IA_n$, for which $IA_1 \neq IA_2 \neq IA_3 \ldots \neq IA_n$ |
| Property 6 | There are inherited methods, $IM_1$, $IM_2$, $IM_3$,…, $IM_n$, for which $IM_1 \neq IM_2 \neq IM_3 \ldots \neq IM_n$ |
| **Properties for CoCohA and LCoCohA** | |
| Property 1 | If n is a non-negative number, then there is only a finite number of classes TCs for which TCs = n. |
| Property 2 | If n is a decimal number, then there are total class inherited indexes TCsCohI for which TCsCohI = n. |

# 5. Discussions

There are two broad uses of the proposed metrics, namely for assessment and for prediction of cohesion in classes and components. CsCohI, LCsCohI, CoCohA, and LCoCohA vary between 0 (low cohesion or high cohesion) and 1 (high cohesion or low cohesion). When CsCohI=1, each method in a class accesses at least one instance variable. This indicates the highest possible cohesion in a class. CsCohI=0 indicates extreme lack of cohesion. In this case, the class should be split into two or more classes. The underlying idea behind the CsCohI is that a class is cohesive if all its methods use at least one of its instance fields, which means that TCohM=TM and then CsCohI=1. The single responsibility principle states that if there are more than one reason to change for a class, then we have to split the functionality of the class in two or more classes. In this context, a responsibility is considered to be one reason to change. Each class handles only one responsibility. We attempt to place the metric value in a classification scheme normalized to the arithmetic mean of the metric value [0, 1] so that transformations techniques can be formulated. We propose several transformation techniques augmented and summarized from Kupin's catalog as recommendations to improve

the metric values as depicted in Table 3. The primary difference between the proposed metrics and existing metrics is the overall size of a class is considered. We believe the overall size of a class can be measured by determining the total number of methods (both inherited and private instance operations) that are encapsulated within the class, and the number of attributes (both inherited and private instance attributes) that are encapsulated by the class. The proposed metrics can be used to identify classes and components that are attempting to provide many different objectives, and consequently are likely to behave in less predictable ways than classes and components that have higher CsCohI and CoCohA, and lower LCsCohI and LCoCohA values.

TABLE 3 RECOMMENDED TRANSFORMATION TECHNIQUES

| **Value of CsCohI, CoCohA, LCsCohI or LCoCohA** | **Transformation Level** | **Transformation Type** | **Transformation Name** | **Description** |
|---|---|---|---|---|
| Low (CsCohI or CoCohA ≤ 0.5)<br><br>(LCsCohI or LCoCohA > 0.5) | Low Level | Creational Transformation | New Class | Creates a new class |
| | | | New Method | Creates a new method for a class |
| | | | New Variable | Creates a new variable for a class |
| | Extended Low Level | Move Transformation | Move Class | Moves an existing class to a target package |
| | | | Move Method | Moves an existing method from a class to a target class |
| | | | Move Variable | Moves an existing variable from a class to a target class |
| High (CsCohI or CoCohA > 0.5)<br><br>(LCsCohI or LCoCohA ≤ 0.5) | High Level | Factory Transformation | New Factory | Creates a new class and a create-method |
| | | | Add Factory Method | Creates a new create-method in an existing class |

# 6. Conclusion

As a key early artifact in the development of object-oriented systems, the quality of classes and components are crucial and serve as major determinants for the quality of object-oriented systems. Quantitative measurement instruments are useful to assess the quality of classes and components in an objective way. In the object-oriented paradigm, cohesion is considered as one of most important characteristics in object-oriented design. Cohesion can be used to identify the poorly designed classes and components in object-oriented systems. A class has a high cohesion level if its variable instances are referred by a method, facet highlighted via LCOM5 metric, or a number of methods pairs refer the same variable instances, facet highlighted via LCOM1, LCOM2, LCOM3, LCOM4, LCC, and TCC metrics. We proposed four metrics to measure the cohesion of individual classes and components within an object-oriented system, namely CsCohI, LCsCohI, CoCohA, and LCoCohA based on different definitions of LCOM. In future, we intend to evaluate the capability of the proposed metrics and other proposed variations of LCOM measures to adequately express cohesion in object-oriented systems through principal component analysis. Through principal component analysis, the relative discriminatory powers of the different variations of LCOM are empirically evaluated. We also intend to investigate the possibilities to augment the total correlation theory to empirically express the amount of redundancy or dependency (i.e. structure) existing among a set of artifacts such as methods and attributes. The applicability is based on the hypothesis that each method in a class uses a subset of the attributes of the class. We want to investigate whether these subsets exhibit a structure between the attributes. If such a structure exists in a class, then the class can be extracted into two or more other classes in order to improve (i.e. reduce or remove) the structure. As such, we can quantitatively evaluate the use of each attribute by the methods as a random binary variable with a specific probability of occurrence.

# 7. References

[1] S.R. Chidamber, and C.F. Kemerer. Towards a Metrics Suite for Object-oriented Design. Proceedings of Object Oriented Programming Systems Languages and Applications. Phoenix, Arizona, United States. 1991, pp. 197-211.

[2] W. Li, and S. Henry. Maintenance Metrics for the Object-oriented Paradigm. *Proceedings of the First International Software Metrics Symposium*. Baltimore, Maryland, United States. 1993, pp. 52-60.

[3] W. Li, and S. Henry. Object-oriented Metrics that Predict Maintainability. *Journal of Systems and Software*. 1993, **23** (2): 111-122.

[4] S.R. Chidamber, and C.F. Kemerer. A Metrics Suite for Object-oriented Design. *IEEE Transactions on Software Engineering*. 1994, **20** (6): 476-493.

[5] W. Li, S. Henry, D. Kafury, and R. Schulman. Measuring object-oriented design. *Journal of Object-oriented Programming*. 1995, **8** (4): 48-55.

[6] B. H. Sellers. *Object-Oriented Metrics: Measures of Complexity*. Prentice-Hall, 1995.

[7] M. Hitz and B. Montazeri. Measuring Coupling and Cohesion in Object-oriented Systems. *Proceedings of the International Symposium on Applied Corporate Computing*. Monterey, Mexico. 1995, pp. 25-27.

[8] J.M. Bieman, and B.K. Kang. Cohesion and Reuse in an Object-oriented System. *Proceedings of the Symposium on Software Reusability*. Seattle, Washington, United States. 1995, pp. 259-262.

[9] M. Hitz, and B. Montazeri. Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective. *IEEE Transactions on Software Engineering*. 1996, **22** (4): 267-271.

[10] B. H. Sellers, L. L. Constantine, and I. M. Graham. Coupling and Cohesion (Towards a Valid Metrics Suite for Object-oriented Analysis and Design). *Object-oriented Systems*. 1996, **3**: 143-158.

[11] S. P. Roger, *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005.

[12] A. Kupin, Design and development of program transformation tool. Master thesis, University of Munich, 2000.

[13] L. Badri, and M. Badri. A Proposal of a New Class Cohesion Criterion: An Empirical Study. *Journal of Object Technology*. 2004, **3** (4): 145-159.

[14] J.F. Gelinas, M. Badri, and L. Badri. A Cohesion Measure for Aspects. *Journal of Object Technology*. 2006, **5** (7): 75-95.

[15] A. Marcus, D. Poshyvanyk, and R. Ferenc. Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems. *IEEE Transactions on Software Engineering*. 2008, **34** (2): 287-300.