International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:11, 2014

Theoretical Exploration for the Impact of Accounting for Special Methods in Connectivity-Based Cohesion Measurement

Jehad Al Dallal

Abstract**—**Class cohesion is a key object-oriented software quality attribute that is used to evaluate the degree of relatedness of class attributes and methods. Researchers have proposed several class cohesion measures. However, the effect of considering the special methods (i.e., constructors, destructors, and access and delegation methods) in cohesion calculation is not thoroughly theoretically studied for most of them. In this paper, we address this issue for three popular connectivity-based class cohesion measures. For each of the considered measures we theoretically study the impact of including or excluding special methods on the values that are obtained by applying the measure. This study is based on analyzing the definitions and formulas that are proposed for the measures. The results show that including/excluding special methods has a considerable effect on the obtained cohesion values and that this effect varies from one measure to another. For each of the three connectivity-based measures, the proposed theoretical study recommended excluding the special methods in cohesion measurement.

Keywords**—**Object-oriented class, software quality, class cohesion measure, class cohesion, special methods.

I. Introduction

DEVELOPING the techniques and the tools needed to develop high-quality applications that are more stable and maintainable is a key goal of software engineering. Developers and managers use several measures to quantify and improve the quality of an application during the development process. These measures estimate the quality of different software attributes, such as cohesion, coupling, and complexity.

The cohesion of a module refers to the relatedness of the module components. A module that has high cohesion performs one basic function and cannot be split into separate modules easily. Highly cohesive modules are more understandable, modifiable, and maintainable [1], [2].

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

Researchers have introduced several measures to indicate class cohesion during high or low level design phases. These measures follow different approached to estimate the cohesion of a class. For example, some of the measures are based on counting the number of pairs of methods that share common attributes [3], [4]. Some others are more precise and they are based on measuring the similarity between each pair of methods in terms of the ratio of the shared common attributes [2], [5], [6]. Other measures consider the connectivity pattern of a graph that represents the cohesive relations between methods and attributes in a class. In this case, the cohesion is measured as the connectivity degree of the graph. In this paper, we consider three class cohesion measures: CBMC [7], PCCC [8], and OL_n [9]. These measures as well as some other measures have been empirically studied [10]-[20].

Classes include special types of methods, such as constructors, destructors, and access and delegation methods. Constructors are used to initialize most or all of the attributes in the class and destructors are used to deinitialize most or all of the attributes. Access methods are classified as either setters or getters. A setter method initializes a single attribute and a getter method returns the reference/value of a single attribute. Finally, a delegation method is used to inquire about the status of a single attribute. Each of these special methods has its own characteristics, which can artificially affect the class cohesion value. Incorrectly determining whether to include or exclude the special methods in cohesion measurement can lead to improper re-designing decisions and actions based on the misleading class cohesion values that are obtained. However, the original definitions for the considered measures do not differentiate between the different types of methods, which makes these measures ill-defined. The impact of including/excluding special methods in cohesion measures on the obtained values and refactoring and fault prediction activities is empirically studied by Al Dallal [16]. However, this impact is not thoroughly theoretically studied yet.

In this paper, we analyze the definitions and formulas of the considered cohesion measures to study the impact of including/excluding each type of special methods on the values that can be obtained by the measures. Based on the analysis, a recommendation is provided for each measure for whether to include or exclude special methods in cohesion measurement.

This paper is organized as follows. Section II provides an overview of the class cohesion measures proposed in literature. Section II reports the theoretical analysis and

Jehad Al Dallal is with the Department of Information Sciences, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait (e-mail: j.aldallal@ku.edu.kw).

results. Finally, Section IV concludes the paper and discusses future work.

II. Related Work

Several class cohesion measures have been proposed in the literature. These measures can be applicable based on high-level design (HLD) [11], [13], [21]-[23] or low-level design (LLD) information [1], [3]-[6]. HLD class cohesion measures rely on information related to class and method interfaces. The more numerous LLD class cohesion measures require an analysis of the algorithms used in the class methods (or the code itself if available) or access to highly precise method postconditions. Class cohesion measures are based on the use or sharing of class attributes. For example, Bieman and Kang [3] describe two class cohesion measures, Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC), to measure the relative number of directly connected pairs of methods and the relative number of directly or indirectly connected pairs of methods, respectively. TCC considers two methods to be connected if they share the use of at least one attribute. A method uses an attribute if the attribute appears in the method's body or the method invokes another method, directly or indirectly, that has the attribute in its body. LCC considers two methods to be connected if they share the use of at least one attribute directly or transitively. Badri [4] introduces two class cohesion measures, Degree of Cohesion-Direct (DCD) and Degree of Cohesion-Indirect (DCI), that are similar to TCC and LCC, respectively, but differ by considering two methods connected also when both of them directly or transitively invoke the same method. Fernández and Peña [5] and Al Dallal and Briand [1] proposed class cohesion measures that account for the similarity between each pair of methods in terms of the number of attributes shared between the methods.

Several class cohesion measures considered the connectivity patterns for the graph that represent the relationship between the methods and attributes in a class. In this paper, we consider the three measures defined in Table I. Related work in the area of software cohesion can be found in [5], [21], [24], [25].

III. Theoretical Analysis

Here, we theoretically study the effect of including or excluding special methods on the values that are obtained by applying each of the three considered measures. This study is based on analyzing the definitions and formulas that are proposed for the measures. In addition, this study is based on the following typical observations:

1. Potentially, constructors and destructors can reference most if not all of the attributes of the class. As a result, there is a higher chance that each one of the constructors and destructors references more distinct attributes than any other method in the class.
2. Each one of the access or delegation methods references a single attribute. Peer access and delegation methods are the setter, getter, and delegation methods that reference the same attribute. Non-peer access and delegation methods reference different attributes.

TABLE I
Definitions of the Three Connectivity-Based Class Cohesion Measures

Measure	Definition
Cohesion Based on Member Connectivity (CBMC) [7]	CBMC(G)=F_c(G)×F_s(G), where F_c(G)=|M(G)|/|N(G)|, M(G)=the number of glue methods in graph G, N(G)=the number of non-special methods in graph G, $F_s(G) = [\sum_{i=1}^{n} CBMC(G^i)]/n,$ n=the number of child nodes of G, and glue methods is the minimum set of methods for which their removal causes the class-representative graph to become disjointed.
Path Connectivity Class Cohesion (PCCC) [8]	$PCCC(C) = \begin{cases} 0 & \text{if } l = 0 \text{ and } k > 1, \\ 1 & \text{if } l > 0 \text{ and } k = 0, \\ \frac{NSP(G_c)}{NSP(FG_c)} & \text{otherwise.} \end{cases}$, where NSP is the number of simple paths in graph G_c, FG_c is the corresponding fully connected graph, and a simple path is a path in which each node occurs once at most.
OL_n [9]	OL_n= The average strength of the attributes, wherein the strength of an attribute is the average strength of the methods that reference that attribute. The strength of a method is initially set to 1 and is computed, in each iteration, as the average strength of the attributes that it references, where *n* is the number of iterations that are used to compute OL.

3. A non-special method can reference any number of distinct attributes; however, typically, it references a lower number of distinct attributes in comparison to constructors and destructors. A non-special method may not reference any attributes, although, theoretically, this is an unusual case.
4. Constructors and destructors as well as access and delegation methods have almost the same characteristics in terms of the number of referenced attributes. That is, each constructor or destructor potentially references most or all attributes, whereas each access or delegation method references a single attribute. Therefore, for the rest of this section, the discussion regarding the impact of including the constructors is applicable to destructors as well. Similarly, the discussion regarding the access methods also applies to delegation methods.

For each of the considered measures, the cases in which the inclusion of constructors and access methods causes the measure value to increase or decrease are identified and analyzed. Based on this analysis, a recommendation to include or exclude the special methods is given. Analytically, special methods have no influence on the cohesion of the class. Therefore, if the inclusion of the special methods usually causes the measure value to increase or decrease, the recommendation will be to exclude them from the cohesion measurement. On the other hand, the recommendation will be to include the special methods if this inclusion slightly changes or does not usually change the obtained measure value and if the inclusion does not increase the cohesion computational complexity.

A. CBMC

The CBMC value is proportional to the minimum number of glue methods (i.e., the minimum number of methods for which removal causes the class-representative graph to become disjointed) and is inversely proportional to the number of considered methods. When the constructor is considered, it will be the best candidate method to be included in the minimum set of glue methods because, typically, the constructor references all attributes. In this case, if the graph, without including the constructor, is connected and then the constructor is included, the minimum set of glue methods will include the constructor and the glue methods of the original graph. Unless the original graph is fully connected, the minimum number of glue methods of the original graph will be fewer than the number of methods that are represented in the graph. Therefore, the CBMC value increases when a method is added, and that method increases the number of glue methods by one, which is the case when a constructor that references all attributes is added. It is important to note that when one or more of the methods do not reference attributes, and, thus, the class-representative graph is disjointed and the CBMC value of the class is zero, the inclusion of the constructor does not change the CBMC value even if the constructor references all of the attributes.

The inclusion of nodes that represent access methods to a connected class-representative graph does not change the minimum set of glue methods. This is because, in this case, the attribute that is referenced by the access method is also referenced by some other methods. Therefore, the removal of the access method does not make the connected graph disjointed. However, the inclusion of access methods increases the number of considered methods and, thus, decreases the CBMC value. As a result, the recommendation is to exclude both constructors and access methods from the CBMC cohesion measurement.

B. PCCC

PCCC is based on counting the number of simple paths from each node to each other node in the class-representative graph. The number of simple paths that are initiated from a node that represents a method greatly depends on the number of attributes that are referenced by that method. Therefore, the inclusion of the constructor increases the average number of simple paths and consequently increases the PCCC value because the constructor references most or all of the attributes in the class. Oppositely, the inclusion of access methods decreases the average number of simple paths and, consequently, decreases the PCCC value because each access method references a single attribute. As a result, the recommendation is to exclude both constructors and access methods from the PCCC measurement.

C. OL_n

The OL_n value depends on the average strength of the class attributes. The strength of an attribute depends on the total strengths of the methods that reference this attribute. Recursively, the strength of each method depends on the total strengths of the attributes that are referenced by the method. Consequently, the constructor potentially has the highest strength because it references all or most of the attributes, and, therefore, when the constructor is considered, the attributes that are connected to the constructor have an average strength that is higher than that when the constructor is excluded. In other words, including the constructor potentially increases the OL_n value.

An access method is connected to one attribute, and, therefore, it has a relatively low strength, which consequently lowers the strength of the accessed attribute. On average, when the access methods are considered, the attributes that are referenced by access methods are expected to have lower strengths than those when the access methods are excluded. Therefore, the inclusion of access methods potentially decreases the OL_n value. Similar to CBMC, the OL_n value for a class with a disjointed representative graph is zero. Therefore, OL_n has effects that are similar to those that are specified for CBMC when the nodes that represent the constructors or access methods are added to an already disjointed graph. As a result, the recommendation is to exclude both constructors and access methods from the OL_n measurement.

IV. Conclusions and Future Work

This paper provides a theoretical analysis for the impact of including several types of special methods in cohesion measurement performed using three different widely applied connectivity-based cohesion measures. The definitions and formulas of the measures are analyzed to figure out the impact of including special methods on the values obtained using the considered measures. The analysis showed that the effect of including special methods varies among the types of the special methods considered and among the measures themselves. Finally, the analysis showed that the values obtained using the considered measures are expected to be artificially affected by the inclusion of the special methods. This indicates the importance of considering this issue whenever a measure is introduced.

In the future, we plan to extend the analysis to include more existing measures and to study the impact of accounting for special methods in cohesion measurement on practical issues of interest for software practitioners such as reusability, maintainability, and testability.

Acknowledgment

The author would like to acknowledge the support of this work by Kuwait University Research Grant WI01/12.

References

[1] Al Dallal, J. and Briand, L., A Precise method-method interaction-based cohesion metric for object-oriented classes, ACM Transactions on Software Engineering and Methodology (TOSEM), 2012, Vol. 21, No. 2, pp. 8:1-8:34.
[2] Al Dallal, J. Object-oriented class maintainability prediction using internal quality attributes, Information and Software Technology, 2013, Vol. 55, No. 11, pp. 2028-2048.

[3] Bieman, J. and Kang, B., Cohesion and reuse in an object-oriented system, Proceedings of the 1995 Symposium on Software reusability, Seattle, Washington, United States, 1995, pp. 259-262.
[4] Badri, L. and Badri, M., A Proposal of a new class cohesion criterion: an empirical study, Journal of Object Technology, 3(4), 2004, pp. 145-159.
[5] Fernández, L., and Peña, R., A sensitive metric of class cohesion, International Journal of Information Theories and Applications, 13(1), 2006, pp. 82-91.
[6] Bonja, C. and Kidanmariam, E., Metrics for class cohesion and similarity between methods, Proceedings of the 44th Annual ACM Southeast Regional Conference, Melbourne, Florida, 2006, pp. 91-95.
[7] HChae, H.S., Kwon, Y. R., and Bae, D. A cohesion measure for object-oriented classes, Software—Practice & Experience, 30(12), 2000, pp.1405-1431.
[8] Al Dallal, J., Fault prediction and the discriminative powers of connectivity-based object-oriented class cohesion metrics, Information and Software Technology, 2012, Vol. 54, No. 4, pp. 396-416.
[9] Yang, X., Research on Class Cohesion Measures, M.S. Thesis, Department of Computer Science and Engineering, Southeast University, 2002.
[10] Al Dallal, J., A design-based cohesion metric for object-oriented classes, International Journal of Computer Science and Engineering, 2007, Vol. 1, No. 3, pp. 195-200.
[11] Al Dallal, J. and Briand, L., An object-oriented high-level design-based class cohesion metric, Information and Software Technology, 2010, Vol. 52, No. 12, pp. 1346-1361.
[12] Al Dallal, J., Improving object-oriented lack-of-cohesion metric by excluding special methods, proceedings of the 10th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS 2011), Cambridge, UK, February 2011.
[13] Counsell, S., Swift, S., and Crampton, J., The interpretation and utility of three cohesion metrics for object-oriented design, ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 15, No. 2, 2006, pp.123-149.
[14] Briand, L. C., Wüst, J., and Lounis, H., Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs, Empirical Software Engineering, 6(1), 2001, pp. 11-58.
[15] Marcus, M., Poshyvanyk, D., and Ferenc, R., Using the conceptual cohesion of classes for fault prediction in object-oriented systems, IEEE Transactions on Software Engineering, 34(2), 2008, pp. 287-300.
[16] Al Dallal, J., The impact of inheritance on the internal quality attributes of java classes, Kuwait Journal of Science, 2012, Vol. 39, No. 2A, pp. 131-154.
[17] Al Dallal, J., Incorporating transitive relations in low-level design-based class cohesion measurement, Software: Practice and Experience, 2013, Vol. 43. No. 6, pp. 685-704.
[18] Al Dallal, J., Constructing models for predicting extract subclass refactoring opportunities using object-oriented quality metrics, Information and Software Technology, 2012, Vol. 54, No. 10, pp. 1125-1141.
[19] Al Dallal, J. and Morasca, S., Predicting object-oriented class reusability using internal quality attributes, Empirical Software Engineering, Vol. 19, No. 4, 2014, pp. 775-821.
[20] Al Dallal, J., The impact of accounting for special methods in the measurement of object-oriented class cohesion on refactoring and fault prediction activities, Journal of Systems and Software, 2012, Vol. 85, No. 5, pp. 1042-1057.
[21] Al Dallal, J., Measuring the discriminative power of object-oriented class cohesion metrics, IEEE Transactions on Software Engineering, 2011, Vol. 37, No. 6, pp. 788-804.
[22] Al Dallal, J., Improving the applicability of object-oriented class cohesion metrics, Information and Software Technology, 2011, Vol. 53, No. 9, pp. 914-928.
[23] Al Dallal, J., Transitive-based object-oriented lack-of-cohesion metric, Procedia Computer Science (Elsevier), Volume 3, 2011, pp. 1581-1587.
[24] Al Dallal, J., Software similarity-based functional cohesion metric, IET Software, 2009, Vol. 3, No. 1, pp. 46-57.
[25] Al Dallal, J., The effects of incorporating special methods into cohesion measurement on class instantiation reuse-proneness prediction, IET Software, in press, 2014.