DEVELOPMENT OF A NOVEL OBJECT ORIENTED COHESION METRIC

S. A. Salem

Department of Electronics, Communications & Computer Engineering
Faculty of Engineering, Helwan University, Helwan, Cairo, Egypt
sameh_salem@h-eng.helwan.edu.eg

Received 14 March 2013, accepted 18 April 2013

ABSTRACT

Class cohesion is considered as one of the most important software quality assessment. Unfortunately, most of cohesion metrics that have been developed do not consider the different intersections among class elements in measuring class cohesion. This paper introduces a novel class cohesion metric which considers the different cohesion intersections. The proposed cohesion metric is tested on more than 35K classes from 16 open-source projects. Experimental results show that the proposed cohesion metric achieves a higher discrimination power along with a vast difference compared with other competitive and well known cohesion metrics. Therefore, it is highly recommended to use the proposed metric for evaluating the software design quality.

Keywords: Object-Oriented Metrics, Class Cohesion, Software Quality.

1. Introduction

Cohesion is considered amongst the most important properties to evaluate the quality of an object oriented design [1, 2, 3]. Cohesion metric measures how well the methods of a class are related to each other. 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. In other words, a cohesive class performs one function while a non-cohesive class performs two or more unrelated functions.

Class cohesion is the degree of the relatedness of the members in the class [1, 2]. Cohesion metrics can be used for different purposes including assessment of design quality [3, 4], prediction of software quality and fault proneness [5, 6], modularization of software [7, 8], identification of reusable components [9, 10], etc. In the literature, 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 [2,11,12,13,14,15,16,17,18,19], the most popular class of cohesion metrics [20], semantic metrics [21], information entropy-based metrics [22], slice-based metrics [23], metrics based on data mining [24], and metrics for specific types of applications like knowledge-based [25] and distributed systems [26].

The class of structural metrics is the most investigated category of cohesion metrics [20]. In this category, most of the developed object-oriented class cohesion metrics focus on measuring the correlation between pairs of methods in the class, e.g. Chidamber and Kemerer Lack of Cohesion in Methods (*LCOM1* and *LCOM2*) metrics [11], [12], Li et al. *LCOM3* metric [13], Hitz et al. *LCOM4* metric [14], Bieman and Kang Tight Class Cohesion (*TCC*) and Loose Class Cohesion (*LCC*) metrics [15], Badri et al. Lack of Cohesion in the Class-Direct (LCC_D) and Lack of Cohesion in the Class-Indirect (LCC_I) metrics [2], Bonja and Kidanmariam Class Cohesion (*CC*) metric [16], and Fernández Sensitive class COhesion Metric (*SCOM*) [17].

Alternatively, Henderson-Sellers [18] proposed *LCOM5* metric as a different approach for measuring the class cohesion by measuring the attribute-method correlation. Because the *LCOM5* metric has a drawback as it is not normalized into ranging between 0 and 1, Braind et al. [19] proposed the *Coh* metric by enhancing the *LCOM5* metric to be normalized.

In a previous study, the effect of inheritance on identifying classes that are nominated for software refactoring is studied. The study concluded that most of the developed cohesion metrics ignore the different types of intersections between attributes and methods within the class, this leads to inaccurate results in cohesion measurements [27]. Therefore, there is a need for design quality measure that is able to examine the class cohesion and taking into consideration the different intersections between class elements.

In this paper, novel class cohesion metric is proposed that enables software assessors to determine classes which are poorly designed and nominated for software refactoring.

This paper is structured as follows. Section 2 describes the novel class cohesion metric. Section 3 illustrates the assessment process. Section 4 provides the experimental results and discussion. Finally, Section 5 draws conclusion.

2. Novel class cohesion metric

This section describes class cohesion analysis along with a detailed description of the proposed metric.

2.1. Class cohesion analysis

Taking into consideration certain types of intersection between class elements introduce a Lack of Discrimination Anomaly (*LDA*) in object-oriented class cohesion metric values and give inaccurate results in cohesion measurements [28]. So, it is essential to consider the different types of intersection and connections in the class when designing the cohesion metric. It should be noted that the proposed cohesion metric is considering the different intersections and connections within the class design as follows:

2.1.1. Connection category

In this type, some cohesion metrics consider only attribute-method connections in the class while other metrics consider only method-method connections.

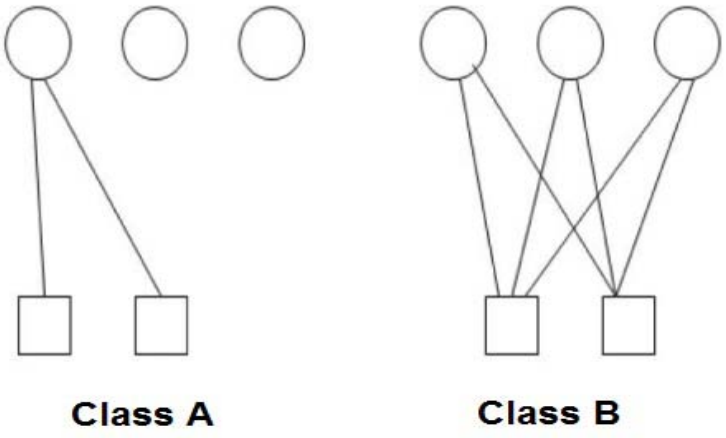

Fig. 1. Two classes with different attribute-method connnections

Figure 1 shows two classes with different connections. Each class has two methods and three attributes. The rectangles and circles represent the methods and attributes respectively, while links represent the attributes used by methods. Although, the two classes are exhibiting two different designs, some method-method cohesion metrics have the same cohesion value for both classes [28], and that means the two classes are totally cohesive which give inaccurate results as some attributes in class A are not used. This gives misleading assessment as these metrics do not include intersections between attributes and methods when assessing the class design, while the proposed cohesion metric consider the above issue while measuring the class cohesion.

2.1.2. Connection type

In connection type, two class members in the class could be connected if there is a direct/indirect attribute or method connecting them, also these class members could be tightly or loosely connected.

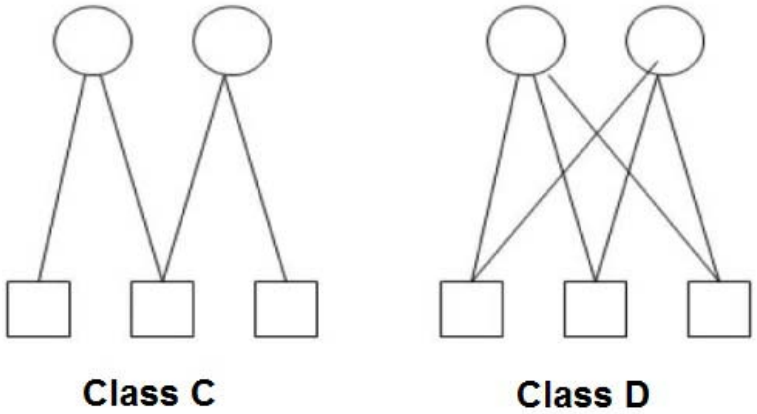

Fig. 2. Two classes with an indirect/direct connections

Figure 2 illustrates two classes with different designs. It should be noted that the metrics that consider only the direct connection between methods do not detect the indirect connection between methods, while the metrics that consider only loose connections (such as *LCC* and LCC_D metrics [2, 15]) have the same cohesion value in both classes though they have different designs (e.g. class C and D).

Consequently, considering only one type in measuring class cohesion (like direct/indirect or tight/loose connected class members) reduces the metric reliability and its discrimination power.

2.1.3. Connection density

Many of the developed cohesion metrics consider pairs of elements as totally cohesive if there is at least one shared element (attribute or method) connecting these elements. Other metrics (such as *CC* and *SCOM* metrics [16, 17]) takes into consideration the number of shared elements in measuring class cohesion.

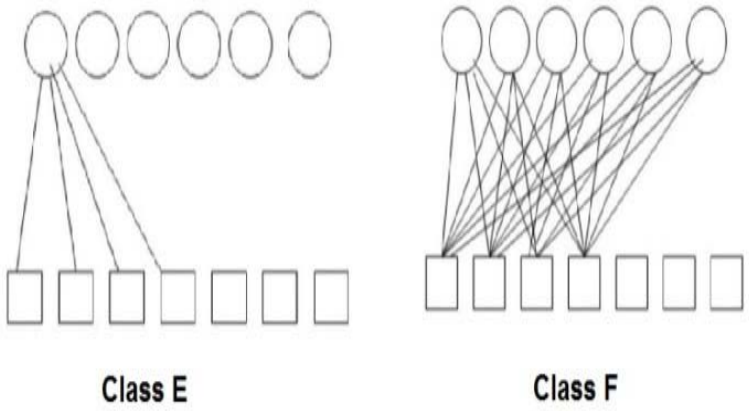

Fig. 3. Two classes with shared element(s)

Figure 3 illustrates two classes with different designs; some method-method cohesion metrics give the same cohesion value for both classes [28]. This is because these metrics do not consider the number of shared elements when assessing class cohesion.

As shown in Fig. 3, excluding the number of shared elements in measuring class cohesion reduces the metric discrimination power.

Therefore, it is essential to consider all these different types of connections in the design of the proposed cohesion metric (i.e. connection category, connection type, and connection density), and that promises and gives more accurate and reliable assessment criteria.

2.2. Proposed cohesion metric

This section illustrates a detailed description of the proposed cohesion metric.

Two components are used to define the proposed metric, the Method-Method component and Attribute-Method component. In this context, a cohesion value for a class C can be measured as follows:

$$Proposed\ Metric\ (C) = \frac{w_M MM(C) + w_A AM(C)}{(w_A + w_M)} \quad (1)$$

where *MM(C)* is the Method-Method cohesion component for class C, while *AM(C)* is the Attribute-Method cohesion component for class C.

It should be noted that the proposed metric definition considers both attribute-method connection (via *AM*(C) component) and the method-method connection (via the *MM*(C) component) in order to increase the metric discrimination power. For simplicity, the weights for each measure w_A and w_M are considered as ones overall the scope of this study.

2.2.1.Method-method class cohesion component

The Method-Method cohesion component for class C is defined as follows:

$$MM(C) = \frac{w_1 DT_{MM}(C) + w_2 IT_{MM}(C) + w_3 L_{MM}(C)}{w_1 + w_2 + w_3} \quad (2)$$

where

DT_{MM} (C) is the **D**irectly and **T**ightly Connected Method-Method cohesion value for class C.

IT_{MM} (C) is the **I**ndirectly and **T**ightly Connected Method-Method cohesion value for class C

L_{MM} (C) is the **L**oosely Connected Method-Method cohesion value for class C.

As demonstrated, the Method-Method component takes into consideration the different connection types (direct/indirect and tight/loose) in measuring the method-method intersections. The w_1, w_2, and w_3 are the weights for each measure.

The DT_{MM} (C) and IT_{MM} (C) metrics of class C is measured as follows:

For a class C having ***a*** attributes and ***m*** methods, consider two methods M_i and M_j. Let IV_{iA} and IV_{jA} be the set of attributes used **directly** (in case of DT_{MM} (C) metric) or **indirectly** (in case of IT_{MM} (C) metric) by methods M_i and M_j respectively, and IV_{iM} and IV_{jM} be the set of methods called directly (in case of DT_{MM} (C) metric) or indirectly (in case of IT_{MM} (C) metric) by methods M_i and M_j respectively.

An attribute is directly used by a method M if this attribute appears as a data token in the method M. An attribute is indirectly used by a method M if this attribute is directly used by another method M' which is called directly or indirectly by method M [15].

The Connection density C_{Dij} between two methods M_i and M_j is defined as follows:

$$C_{Dij} = \frac{w_A \alpha_{Ai,j} + w_M \alpha_{Mi,j}}{w_A + w_M} \tag{3}$$

where:

$$\alpha_{Mi,j} = \frac{card(I_{iM} \cap IV_{jM})}{\min(card(I_{iM}), card(IV_{jM}))} \times \frac{card(I_{iM} \cup IV_{jM})}{m-2}$$

$$\alpha_{Ai,j} = \frac{card(I_{iA} \cap IV_{jA})}{\min(card(I_{iA}), card(IV_{jA}))} \times \frac{card(I_{iA} \cup IV_{jA})}{a}$$

where card is the cardinality.

The C_{Dij} component takes into consideration the connection density by measuring the normalized number of shared attributes and methods between pairs of methods in the class. For example in Figure 3 the *MM(C)* metric value for class *F* is higher than the metric value in class *E* because the number of shared elements between pairs of methods in class *F* are greater than those in class *E*. The w_A and w_M are the weights for each measure.

Let CC_{MM} (C) is the percentage of pairs of methods of a class C which are tightly, and it is defined as follows:

$$CC_{MM}(C) = 1 - \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{Dij} \tag{4}$$

where CC_{MM} (C) equals to DT_{MM} (C) or IT_{MM} (C) if the C_{Dij} component measures the direct or indirect connections between pairs of methods respectively.

Loosely Connected Method-Method Metric

The loose connection is defined as follows, a method M_1 is loosely connected with a method M_n if there is a sequence of methods M_2, M_3, ..., M_{n-1} such that M_i and M_{i+1} are tightly connected for i = 1, ... n − 1.

L_{MM} (C) is defined as the percentage of pairs of methods of the class C which are loosely connected, it is defined as follows:

$$L_{MM}(C) = 1 - \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{Lij} \tag{5}$$

where $C_{Lij} = \begin{cases} 1 & \text{if } M_i \text{ and } M_j \text{ are loosely connected} \\ 0 & \text{otherwise} \end{cases}$

2.2.2. Attribute-method class cohesion component

The Attribute-Method cohesion component for class C could be measured as follows:

$$AM(C) = \frac{w_1 DT_{AM}(C) + w_2 IT_{AM}(C) + w_3 IL_{AM}(C)}{w_1 + w_2 + w_3} \tag{6}$$

where DT_{AM} (C) is the **D**irectly and **T**ightly Connected Attribute-Method cohesion component for a class C, DL_{AM} (C) is the **D**irectly and **L**oosely Connected Attribute-Method cohesion value for class C, and IL_{AM} (C) is the **I**ndirectly and **L**oosely Connected Attribute-Method cohesion value for class C.

Therefore, the *AM(C)* metric takes into consideration both the direct/indirect and tight/loose intersections between attributes and methods. The w_1, w_2, and w_3 are the weights for each measure.

Each attribute-method cohesion metric value could be measured as follows.

For a class C having ***a*** attributes and ***m*** methods, the attribute usage matrix (**m**×***a*** matrix) is defined, this matrix has a row for each method and ***a*** column for each attribute in the class. For each method M_i and attribute A_j the element C_{ij} in the attribute usage matrix is defined as follows:

$$C_{Lij} = \begin{cases} 1 & \text{if method } M_i \text{ is connected to attribute } A_j \\ 0 & \text{otherwise} \end{cases}$$

An attribute A_j is connected to method M_i in one of the following cases:

a) *Direct Tight Connection:* That means method M_i accesses A_j directly.

b) *Direct Loose Connection:* That means method M_i is loosely connected to attribute A_j through direct calls between methods pairs.

c) *Indirect Loose Connection:* That means method M_i is loosely connected to attribute A_j through indirect calls between methods pairs.

The proposed metric enhances the loose method-method connection by generalizing the concept to be applied on attribute-method connections. Thus, a method M_1 is loosely connected with an attribute A_j if there is a sequence of methods M_2, M_3, ..., M_{n-1} such that M_i and M_{i+1} are tightly connected for i = 1, ... n − 1.

For two methods M_i and M_j the cohesion value D_{ij} for pairs of methods is measured as follows:

$$D_{ij} = \frac{I_c \times I_t}{I_{min} \times a} \tag{7}$$

where I_c be the number of attributes that have value one in both rows i and j in the attribute usage matrix, I_t be the number of attributes that have value one in at least one of the rows i and j in the attribute usage matrix, and I_{min} be the minimum number of attributes that have value one in the rows i and j in the attribute usage matrix.

Finally CC_{AM} (C) metric of class C could be measured as follows:

$$CC_{AM}(C) = 1 - \frac{2}{m(m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} C_{Dij} \tag{8}$$

where CC_{AM} (C) equals to DT_{AM} (C), DL_{AM} (C), or IL_{AM} (C) if the D_{ij} component measures the direct tight connection, direct loose connection, or indirect loose connection between attributes and methods respectively.

3. ASSESMENT PROCESS

This section describes the proposed steps to evaluate the novel class cohesion metric.

3.1. Experimental procedures

Experiments are carried out on more than 35K classes from more than 16 open-source projects (e.g. GWT [29], JDK [30], JRubby [31], Jboss application server [32], and JUnit [33], ...) using the most used cohesion metrics. The experiments are applied by measuring the proposed class cohesion metric in addition to other 12 well known cohesion metrics [28]. It should be noted that open-source projects have been selected to cover the following criteria:

1. Variation in vendors (selected projects are developed in different organizations with different organizations scales).
2. Variation in categories (selected projects are distributed in different domains like games, tools, application server, development, graphics, and communications).
3. Variation in scale (some projects are in range of a few hundred of classes, other projects are in a scale of thousands of classes).

3.2. Evaluation process

In order to evaluate the object-oriented class cohesion metrics, AL Dallal [28] introduced the Discriminative Power of a class Cohesion metric (DPC) which is defined as the probability that cohesion metric provides different cohesion values for classes of the same model but with different CPCIs (Connectivity Pattern of Cohesive Interactions).

Given a metric *s*, a model with *m* methods, and *a* attributes. Let DCP (*m*, *a*) is the number of possible distinct CPCIs for the class model, and DCV (*s*, *m*, *a*) is the number of

distinct cohesion values when the metric *s* is applied to all possible distinct CPCIs of the class model. The DPC value is then defined as follows:

$$DPC(s,m,a) = \frac{DCV(s,m,a)}{DCP(m,a)} \tag{9}$$

A highly discriminating cohesion metric is more desirable because the metrics that produce high average DPC values predict faulty classes better than do those that produce low average DPC values [28]. In order to study the DPC for the proposed class cohesion metric, the proposed metric is applied on different generated class designs and the DPC is calculated and compared with DPC value for other well known class cohesion metrics [28].

In order to calculate the DPC for each cohesion metric, a Cohesion Measurement Tool (CMT) and a DPC Analysis Tool (DAT) were developed. Figure 4 illustrates the measurement process that is applied on different open-source projects.

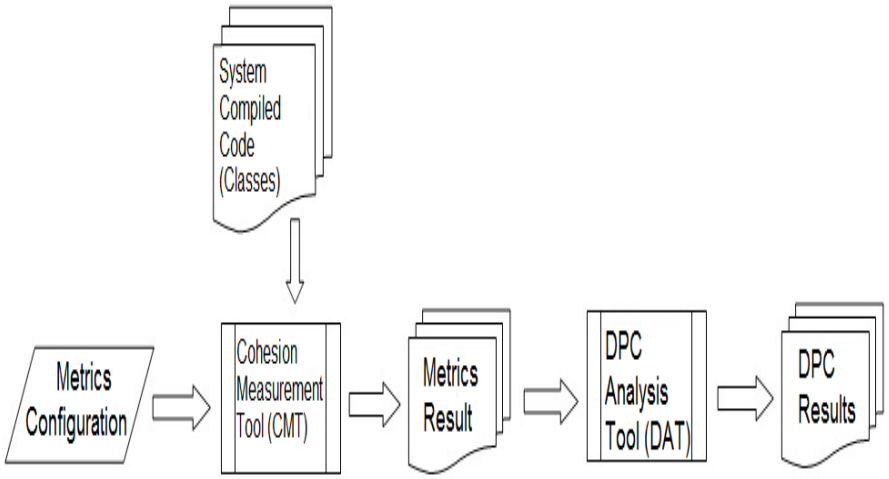

Fig. 4. Assesment Process

4. Results and discussion

This section describes the experimental results obtained by applying the mentioned criteria in section 3. Additionally, the assessment of the proposed cohesion metric is illustrated using different generated patterns Metric Evaluation Using Different Class Designs

The proposed class cohesion metric is applied on different generated designs and the DPC value is calculated for method-method and attribute-method with various combinations. In order to study each effect separately, the tool is configured to consider the following cases.

1. Case 1: DPC is calculated for all possible attribute-method combinations (with fixed method-method design) using different cohesion metrics.

2. Case 2: DPC is calculated for all possible method-method combinations (with fixed attribute-method design) using different cohesion metrics.
3. Case 3: DPC is calculated for all possible method-method and attribute-method combinations, using different cohesion metrics.

Figure 5 depicts the change in DPC values for different metrics as the number of attributes changes across three methods. These results are sample results for Case 1, While Figure 6 depicts the change in DPC values for different metrics as the number of methods changes. These results are sample result for Case 2.

Figures 7 and 8 depict the change in DPC values for different metrics as the number of attributes changes across two and five methods respectively. These results are sample results for Case 3 which means all possible method-method and attribute-method combinations are generated.

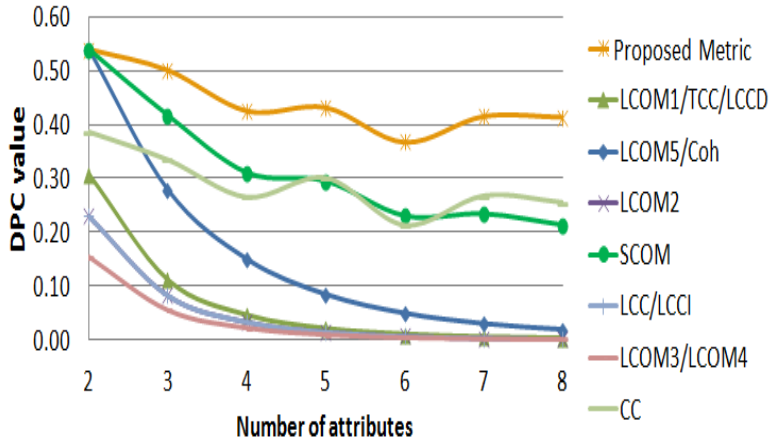

Fig. 5. The change in DPC values as the number of attributes changes (Number of methods=3 and fixed method-method design) – (Case 1)

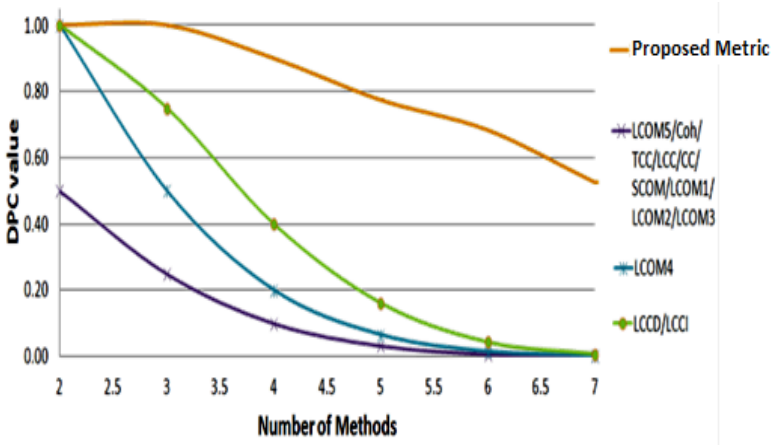

Fig. 6. The change in DPC values as the number of methods changes (Case 2)

It should be noted that the results that are obtained from the DPC values can be analyzed as follows:

1. In case 1 as shown in Fig. 5 that includes only all possible attribute-method changes, different method-method metrics have the same DPC values although these metrics differ in their designs. These could be motivated as the design of these metrics does not include many cases of attribute-method connectivity. It should be noted that the proposed metric has the higher DPC value than the other existing metrics, and this could be explained as our proposed attribute-method metric considers both the direct/indirect connections between both attributes-methods and the effect of method-method connections.
2. In case 2 as shown in Fig. 6 that includes only all possible method-method changes, the attribute-method metrics (*LCOM5* and *Coh* metrics) have the same value, and that is because all generated designs have the same attribute-method relations. Again, our proposed metric has the higher DPC values compared with the other existing metrics. This is because our proposed method-method metric considers both the direct/indirect connections between both attributes and methods, and the density of the connection between methods in addition to the distribution of the connections between class elements.
3. In case 3 as shown in Figs. 7 and 8, our proposed metric have the highest DPC values compared with the other existing metrics.

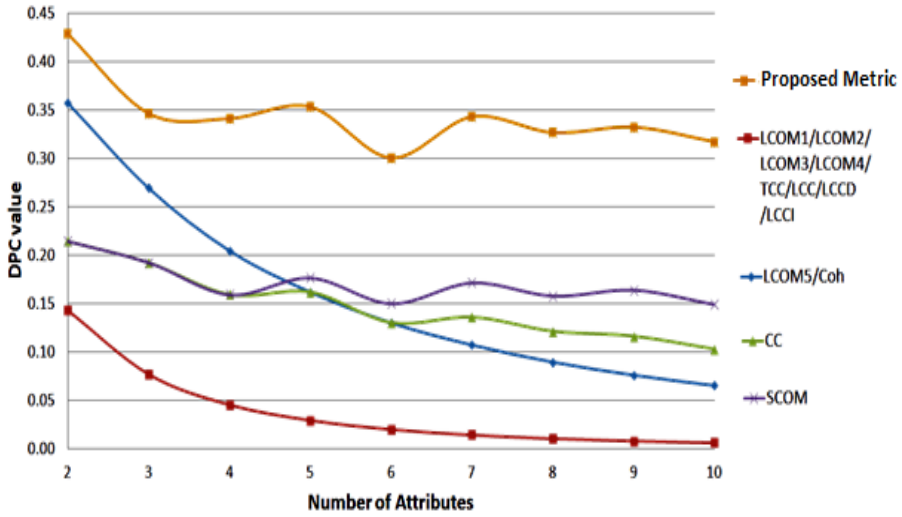

Figure 7. The change in DPC values as the number of attributes changes (Number of methods=2)

Additionally, most of the metrics in case of higher number of attributes and methods provides huge number of possible cases. In this context, most of the competitive metrics provide DPC with very small values (less than 0.005) though our proposed metric has significant difference in DPC values. This is because our proposed metric takes into consideration the different types of connectivity between class elements as illustrated in section 2.

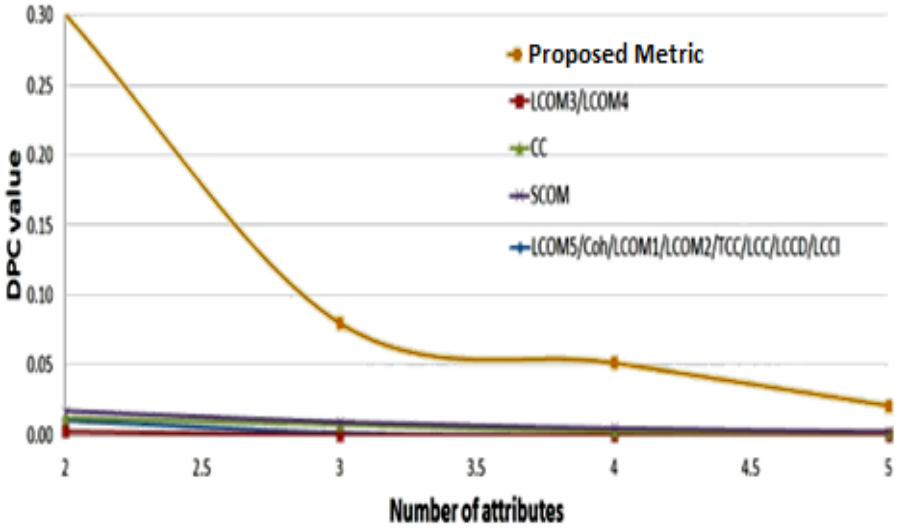

Fig. 8. The change in DPC values as the number of attributes changes (Number of methods=5)

4.1.Metric evaluation using open-source projects

As illustrated in Section 3, various object-oriented cohesion metrics are applied on several open-source Java projects and then the DPC value is calculated per unique model (certain number of attributes and methods). Finally, the average DPC value is calculated for each cohesion metric per project.

Figure 9 summarizes the obtained results; it could be concluded from the experimental results the following:

1. The existing attribute-method metrics (such as *LCOM5* and *Coh* metrics) concern only on direct connectivity between attributes and methods and are not considering the indirect connectivity between attributes and methods, and methods with each other. This explains the motivation for proposing this novel metric which has the ability to give higher discrimination power than existing attribute-method metrics.
2. The existing method-method metrics are concerned with limited types of connectivity between methods and do not include all connectivity types between methods as illustrated in section 2. For example, the *CC* and *SCOM* metrics include the density of the method-method connectivity measurement that explains their higher DPC value than other metrics that are not considering the connection density. However, these metrics (*CC* and *SCOM*) are not including in their assessments the indirect connection between methods which leads to lower DPC values than our proposed metric as verified by the experimental results.

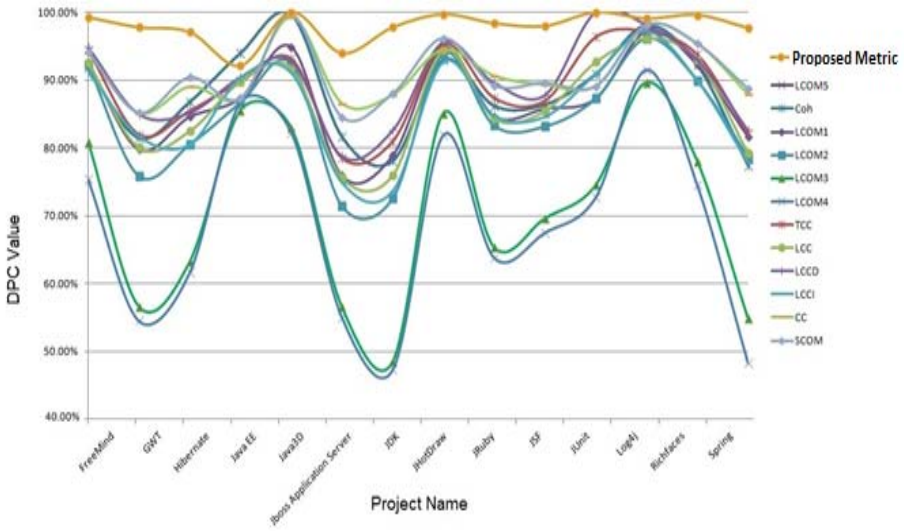

Fig. 9. DPC value for several open-source Java projects

5. Conclusion

In this paper, novel class cohesion metric has been proposed. As demonstrated, the proposed metric is considering not only the different intersections and connections between class elements in the assessment criterion. An empirical evaluation of the proposed metric was performed through not only synthetic class designs but also on large scale of open-source Java projects Experimental results show that the proposed metric is more robustness and has the highest discrimination power over the other existing cohesion metrics, and it can be used as a good indicator to predict fault-proneness of object-oriented classes.

6. References

[1] J. Al Dallal and L. C. Briand, An Object-Oriented High-Level Design-Based Class Cohesion Metric, Simula Research Laboratory, 2009.

[2] L. Badri and M. Badri, "A Proposal of a New Class Cohesion Criterion: An Empirical Study", Journal Of Object Technology, Vol. 3, No. 4, 2004, pp. 145-159.

[3]Bansiya, J. and Davis, C. G., "A hierarchical model for object-oriented design quality assessment", IEEE Transactions on Software Engineering, vol. 28, no. 1, January 2002, pp. 4-17.

[4]Briand, L. C., Wüst, J., Daly, J. W., and Porter, V. D., "Exploring the relationship between design measures and software quality in object-oriented systems", Journal of System and Software, vol. 51, no. 3, May 2000, pp. 245-273.

[5]El-Emam, K., "Object-Oriented Metrics: A Review of Theory and Practice", in Advances in software engineering, Springer-Verlag, New York, 2002, pp. 23-50.

[6]Quah, T.-S. and Thwin, M. M. T., "Application of neural networks for software quality prediction using object-oriented metrics", in Proceedings of International Conference on Software Maintenance, September 22-26 2003, pp. 116-125.

[7]Brito e Abreu, F. and Goulao, M., "Coupling and cohesion as modularization drivers: are we being overpersuaded?" in Proceedings of 5th European Conference on Software Maintenance and Reengineering, 2001, pp. 47-57.

[8]Maletic, J. I. and Marcus, A., "Supporting Program Comprehension Using Semantic and Structural Information", in Proceedings of 23rd International Conference on Software Engineering, Toronto, Canada, May 12-19 2001, pp. 103-112.

[9]Etzkorn, L. H. and Davis, C. G., "Automatically Identifying Reusable OO Legacy Code", IEEE Computer, vol. 30, no. 10, October 1997, pp. 66-72.

[10] Lee, J. K., Jung, S. J., Kim, S. D., Jang, W. H., and Ham, D. H., "Component identification method with coupling and cohesion", in Proceedings of Eighth Asia-Pacific Software Engineering Conference, December 2001, pp. 79-86.

[11] S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design", In Proceedings of Conference on Object-Oriented Programming Systems, Languages and Applications, 1991, pp. 197-211.

[12] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol. 20, No. 6, 1994, pp. 476-493.

[13] W. Li. and S. Henry, "Maintenance Metrics for the Object Oriented Paradigm", In Proceedings of 1st Int. Software Metrics Symp., 1993.

[14] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object Oriented Systems", In Proceedings of the International Symposium on Applied Corporate Computing, 1995.

[15] J. M. Bieman and B. Kang, Cohesion and Reuse in an Object-Oriented System, In Proceedings of the 1995 Symposium on Software reusability, 1995, pp. 259-262.

[16] C. Bonja and E. Kidanmariam, "Metrics for Class Cohesion and Similarity Between Methods", In Proceedings of the 44th Annual Southeast Regional Conference, 2006, pp. 91-95.

[17] R. P. Fernández, "A Sensitive Metric of Class Cohesion", International Journal of Information Theories & Applications, Vol. 13, 2006, pp. 82–91.

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

[19] L. C. Briand, J. W. Daly, and J. K. Wüst, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", Empirical Software Engineering, Vol. 3, No. 1, 1998, pp. 65-117.

[20] 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.

[21] Etzkorn, L. and Delugach, H., "Towards a semantic metrics suite for object-oriented design", in Proceedings of 34th International Conference on Technology of ObjectOriented Languages and Systems, July 30 2000, pp. 71 - 80.

[22] A1 Allen, E. B., Khoshgoftaar, T. M., and Chen, Y., "Measuring coupling and cohesion of software modules: an information-theory approach", in Proc. of 7th International Software Metrics Symposium, April 4-6 2001, pp. 124-134

[23] Meyers, T. M. and Binkley, D., "Slice-based cohesion metrics and software intervention", in Proceedings of 11th Working Conference on Reverse Engineering (WCRE'04), Nov. 8-12 2004, pp. 256-265.

[24] Montes de Oca, C. and Carver, D. L., "Identification of data cohesive subsystems using data mining techniques", in Proceedings of International Conference on Software Maintenance (ICSM'98), November 1998, pp. p. 16-23.

[25] Kramer, S. and Kaindl, H., "Coupling and cohesion metrics for knowledge-based systems using frames and rules", ACM Transactions on Software Engineering and Methodology, vol. 13, no. 3, July 2004, pp. 332-358.

[26] Cho, E. S., Kim, C. J., Kim, D. D., and Rhew, S. Y., "Static and dynamic metrics for effective object clustering", in Proceedings of Asia Pacific International Conference on Software Engineering, 1998, pp. 78 - 85.
[27] S. M. Ibrahim, S. A. Salem, M. A. Ismail, and M. Eladawy, "Identification of Nominated Classes for Software Refactoring Using Object-Oriented Cohesion Metrics", International Journal of Computer Science Issues, Vol. 9, No. 2, 2012, pp. 68-76.
[28] J. Al Dallal, "Measuring the Discriminative Power of Object-Oriented Class Cohesion Metrics", IEEE Trans. Software Eng., Vol. 37, No. 6, 2011, pp. 788-804.
[29] http://code.google.com/p/google-web-toolkit/
[30] http://www.oracle.com/technetwork/java/javase/downloads
[31] http://jruby.org/download
[32] https://github.com/jbossas/jboss-as
[33] https://github.com/KentBeck/jun

الملخص

يعتبر ترابط الفصيلة أحد أهم سبل تقييم جودة البرمجيات .لسوء الحظ معظم مقاييس الترابط التي تم تطويرها لا تنظر في التقاطعات المختلفة بين عناصر الفصيلة الواحدة في قياس الترابط. هذا البحث يقدم مقياس ترابط جديد يأخذ في الاعتبار التقاطعات المختلفة للترابط. تم إختبار المقياس المقترح على فصائل أكثر من 35ك من 16مشروعا مفتوح المصدر. تظهرالنتائج التجريبية أن مقياس الترابط المقترح يحقق أعلى قوة تمييز جنبا إلى جنب بفارق كبير مقارنة مع غيره من مقاييس الترابط الـتنافسية الـمعروفة للجميع. لذلك، ينصح بإستخدام المقياس المقترح لـتقييم جودة تصميم البرمجيات.