International Journal of Engineering and Science Invention (IJESI)
ISSN (Online): 2319-6734, ISSN (Print): 2319-6726
www.ijesi.org ||Volume 15 Issue 4 April 2026 || PP 36-39

SICC: A Structural Interaction-Based Class Cohesion Metric

Levani Mtivlishvili[1]
[1](*Faculty of Informatics and Control Systems, Georgian Technical University, Tbilisi, Georgia)*

ABSTRACT: *Class cohesion is one of the most important internal quality characteristics in object-oriented programming systems. Various metrics have long been used to evaluate software quality; however, none fully captures all aspects that define cohesion. Existing metrics often fail when relationships between methods involve indirect connections rather than simple attribute sharing. This paper introduces a new cohesion metric, SICC (Structural Interaction Class Cohesion), which incorporates both direct and indirect relationships between methods and attributes, considers dependency depth, and excludes unused attributes. The proposed metric extends the Jaccard Index with depth-based weighting and penalty coefficients. Experimental evaluation on a sample class demonstrates that SICC provides a more accurate and comprehensive assessment of cohesion compared to traditional metrics.*

KEYWORDS: *Class cohesion, SICC, Jaccard index, object-oriented metrics, software quality*

Date of Submission: 08-04-2026 Date of acceptance: 20-04-2026

I. Introduction

Class cohesion is a key internal quality attribute in object-oriented systems. Numerous metrics such as LCOM, TCC, LCC, Class Cohesion, and CAMC have been proposed; however, none provide a complete model that captures all essential aspects of cohesion. Existing metrics often produce inaccurate results when relationships between class elements involve indirect connections. These limitations can be categorized as follows:

- Metrics based on attribute usage;
- Metrics based on direct and indirect attribute usage;
- Metrics based on method parameter types.

In real-world systems, methods are often tightly interconnected to accomplish a task, yet this connection is not always reflected through shared attributes. For example, method m1 may call method m2, which uses attribute a1. Thus, m1 has an indirect relationship with a1, which should contribute to cohesion. Ignoring such relationships leads to incomplete evaluation. Another drawback of traditional metrics is that they consider unused attributes, leading to "class pollution" and overly pessimistic cohesion assessments. Therefore, a new metric is needed that considers:

- Direct and indirect relationships between methods and attributes;
- Dependencies among methods;
- Depth and strength of relationships.

II. A Structural Interaction-Based Class Cohesion Metric

Class cohesion is one of the most important internal quality characteristics in object-oriented programming systems. Therefore, various metrics have long been used in the process of evaluating software quality. However, despite numerous studies and widely used metrics developed over the years, such as LCOM, TCC, LCC, Class Cohesion, CAMC, and others, there is still no model that comprehensively considers all essential aspects that truly define class cohesion [1, 2, 3, 4, 5].

Existing metrics often produce inaccurate results in cases where internal class relationships are expressed not only through shared attributes but also through indirect connections between methods and inter-method relationships [6, 7]. The limitations of existing metrics can be categorized as follows:

- Metrics based on attribute usage;
- Metrics based on direct and indirect attribute usage;
- Metrics based on method parameter types.

Despite these approaches, real-world projects often contain classes where relationships between methods are more complex. Methods may be tightly connected to accomplish a single task, but this connection is not reflected solely through shared attributes. For example, method m1 may directly call method m2, which in turn uses attribute a1. In this case, the relationship between m1 and a1 is indirect, yet it can still be considered strongly

cohesive. Ignoring such hidden relationships prevents a complete evaluation of cohesion and may misrepresent the strength or weakness of a class's cohesion.

One significant drawback of most traditional cohesion metrics is that they consider class attributes that are not used by any method. This situation is often referred to as class pollution, as such attributes formally increase structural complexity but do not contribute to functional behavior. As a result, including these attributes leads to an overly pessimistic evaluation of the class's structural integrity. Given these limitations, there is a clear need for a new cohesion metric that takes into account:

- Direct and indirect influence of methods on attributes;
- Interdependencies between methods;
- The depth and quality of relationships.

Such a metric would allow for a more accurate evaluation of class cohesion and improve the analysis process. Traditionally, an indirect relationship is defined as a connection where a method is linked to an attribute through another method that shares that attribute. However, traditional metrics do not capture method calls. We introduce a new concept of indirect connection based on method invocation.

According to class C (Fig. 1), method m1 has an indirect connection with attribute a3 because m1 calls m2, which has a direct connection with a3.

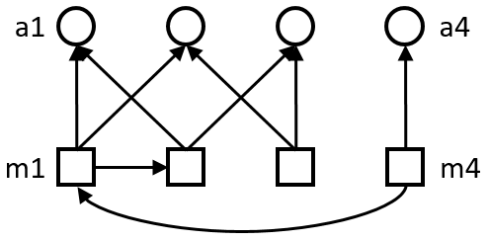

Figure 1. Graphical representation of class C

The depth of a direct connection is 0, since the method directly accesses the attribute. Each indirect connection based on method calls increases the depth by one. Method m4 has:

- a direct (depth 0) connection with attribute a4;
- first-level indirect connections with attributes a1 and a2 through m1;
- a second-level indirect connection with attribute a3 through m1 and m2 (m4 -> m1 -> m2).

Thus, for class C (Fig. 1), method-attribute relationships are:

- $A_{m1} = \{A_1^{D_0}, A_2^{D_0}, A_3^{D_1}\}$
- $A_{m2} = \{A_1^{D_0}, A_3^{D_0}\}$
- $A_{m3} = \{A_2^{D_0}, A_3^{D_0}\}$
- $A_{m4} = \{A_1^{D_1}, A_2^{D_1} A_3^{D_2}, A_4^{D_0}\}$

In set similarity theory, one important measure is the **Jaccard Index**, which is used to evaluate similarity between two sets. It is defined as the ratio of the intersection of elements to the union of unique elements [8]. Bieman and Kang first applied the Jaccard Index to measure method similarity in their Class Cohesion metric [9]. According to the Jaccard Index, similarity between methods m1 and m2 is:

$$JaccardIndex(m_1, m_2) = \frac{A_{m1} \cap A_{m2}}{A_{m1} \cup A_{m2}}$$

However, the Jaccard Index assumes equal weight for all elements. In our case, elements may have different weights due to varying depths. Therefore, the similarity formula must be modified to incorporate depth analysis. The new class cohesion formula is:

$$SICC = \frac{\sum Similarity(i,j)}{n \cdot \frac{n-1}{2}}$$

where:

- n is the number of methods;
- Similarity(i, j) is a modified Jaccard-based similarity measure that considers dependency depth and introduces penalty coefficients.

The similarity is calculated as:

$$Similarity(i,j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \cdot \prod_{k=0}^{depth} \left(1 - \frac{|A_i^{D_k} \cup A_j^{D_k}|}{|A_i \cup A_j|} \cdot \frac{k}{1+k}\right)$$

Where:

- $\boldsymbol{A_i}, \boldsymbol{A_j}$ are attribute sets used by method m_i, m_j;
- $|\boldsymbol{A_i} \cap \boldsymbol{A_j}|$ is the number of shared attributes;
- $|\boldsymbol{A_i} \cup \boldsymbol{A_j}|$ is the number of unique attributes;
- depth is the maximum depth of attribute connections;
- $\boldsymbol{A^{D_k}}$ represents attributes at depth D_k.

For methods m1 and m2, depth = 1, thus:

$$Similarity(m1, m2) = \frac{|A_{m1} \cap A_{m2}|}{|A_{m1} \cup A_{m2}|} \cdot \left(1 - \frac{\left|A_{m1}^{D_0} \cup A_{m2}^{D_0}\right|}{|A_{m1} \cup A_{m2}|} \cdot \frac{0}{1+0}\right) \cdot \left(1 - \frac{\left|A_{m1}^{D_1} \cup A_{m2}^{D_1}\right|}{|A_{m1} \cup A_{m2}|} \cdot \frac{k}{1+k}\right)$$
$$= \frac{2}{3} \cdot 1 \cdot \left(1 - \frac{1}{3} \cdot \frac{1}{2}\right) \approx 0.5556$$

Pairwise similarities for class C are shown in Table 1:

Table 1. Similarity matrix of methods in Class C based on the SICC metric

Method Pair	Similarity
m1,m2	0.5556
m1,m3	0.5556
m1,m4	0.3906
m2,m3	0.3333
m2,m4	0.3125
m3,m4	0.3125

Thus, for class C:

$$SICC = \frac{\sum Similarity(i,j)}{n \cdot \frac{n-1}{2}} \approx 0.41$$

Although SICC accurately describes class cohesion, it requires constructing a dependency graph. For simplification, if only first-level depth is considered, the similarity formula becomes:

$$Similarity(i,j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \cdot \left(1 - \frac{|A_i^{D_1} \cup A_j^{D_1}|}{2|A_i \cup A_j|}\right)$$

Boundary values:

- Lower bound (~0): low cohesion - methods rarely share attributes directly or indirectly;
- Upper bound (~1): high cohesion - most or all methods share attributes directly or indirectly.

Advantages:

- Evaluates method similarity based on shared attributes;
- Describes inter-method relationships;
- Does not require detailed knowledge of the class.

Disadvantages:

- Requires construction of a dependency graph.

III. Conclusion

The SICC metric is a powerful tool for analyzing internal class relationships. It describes how tightly class methods are connected through shared attributes or method calls. Traditional cohesion metrics alone often do not provide sufficient results and are typically used in combination.

SICC represents a significant evolution in cohesion evaluation. It combines Jaccard-based similarity, ignores unused attributes, incorporates depth analysis, and accounts for indirect relationships. As a result, it surpasses classical cohesion metrics by capturing both direct and indirect connections and providing a more accurate and comprehensive analysis of class cohesion.

References

[1] Bieman, J. M., & Kang, B.-K. (1995). Cohesion and reuse in an object-oriented system. In Proceedings of the 1995 ACM Symposium on Software Reusability (pp. 259–262). ACM.

[2] Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6), 476–493.

[3] Fernández, L., & Peña, R. (2006). Understanding class cohesion metric (SCOM). In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06). IEEE.

[4] Henderson-Sellers, B. (1996). Object-oriented metrics: Measures of complexity. Prentice Hall PTR.

[5] Henderson-Sellers, B., Constantine, L. L., & Graham, I. M. (1996). Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design). Object Oriented Systems, 3, 143–158.

[6] Hitz, M., & Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems. In Proceedings of the International Symposium on Applied Corporate Computing.
[7] Li, W., & Henry, S. (1993). Object-oriented metrics that predict maintainability. Journal of Systems and Software, 23, 111–122.
[8] Tan, P.-N., Steinbach, M., & Kumar, V. (2018). Introduction to data mining (2nd ed.). Pearson.
[9] Bonja, C., & Kidanmariam, E. (2006). Metrics for class cohesion and similarity between methods. In Proceedings of the 44th Annual Southeast Regional Conference (pp. 91–95). ACM