A Class Cohesion Metric Focusing on Cohesive-Part Size

Hirosa AMAN†, Member, Kenji YAMASAI†, Nonmember, Hiroyuki YAMADA†, Member, and Matu-Tarow NODA†, Nonmember

SUMMARY Cohesion is an important software attribute, and it is one of significant criteria for assessing object-oriented software quality. Although several metrics for measuring cohesion have been proposed, there is an aspect which has not been supported by those existing metrics, that is “cohesive-part size.” This paper proposes a new metric focusing on “cohesive-part size,” and evaluates it in both of qualitative and quantitative ways, with a mathematical framework and an experiment measuring some Java classes, respectively. Through those evaluations, the proposed metric is showed to be a reasonable metric, and not redundant one. It can collaborate with other existing metrics in measuring class cohesion, and will contribute to more accurate measurement.

key words: object-oriented software, metrics, cohesion, mathematical framework, correlation analysis

1. Introduction

Cohesion is one of important attributes for software module, representing the degree to which its components are functionally connected within the module [1]–[3]. This attribute could be applied to object-oriented software, where a class corresponds to a module, and its attributes and methods correspond to the module components. In general, high level cohesion may lead to high maintainability, reusability and reliability [4], [5], so that class cohesion would be one of important criteria for assessing object-oriented software quality.

For measuring class cohesion, several metrics have been proposed, such as “Lack of Cohesion in Methods (LCOM)” [6]–[10], “Information flow-based Cohesion (ICH)” [11], “Tight Class Cohesion (TCC)” and “Loose Class Cohesion (LCC)” [12]. These metrics are mainly based on “the number of sets of connected methods,” or “the density of method-connections/attribute-accesses” in a class. However there is an aspect of cohesion which has not been supported by the above metrics. That is “size of cohesive-part” in a class, in other words, “extent of associations among methods” through attribute-accesses and/or method-invocations in a class. In order to support such lacking aspect, and to contribute more accurate measurement of cohesion, this paper proposes a new cohesion metric focusing on cohesive-part size, and evaluates it in both of qualitative and quantitative ways.

The rest of this paper is organized as follows. Section 2 briefly describes the existing eight metrics for measuring class cohesion. Section 3 presents a new cohesion metric focusing on cohesive-part size, and Sect. 4 evaluates it in both of qualitative and quantitative ways. Section 5 gives our conclusions and future works.

2. Existing Class Cohesion Metrics

Several metrics are proposed in order to measure cohesion of object-classes [6]–[12]. This section presents some brief descriptions of those existing metrics. See the literature [5] for their detailed discussion.

2.1 Lack of Cohesion in Methods (LCOM)

“Lack of COhesion in Methods” (LCOM) is a well-known class cohesion metric which quantifies poorness of cohesion. The higher value represents the lower cohesion, and vice versa. LCOM has thus far been discussed by several researchers [6]–[10]. Through their works, there are the following five different types of LCOM. These metrics have studied on usefulness as predictors or indicators of maintenance effort [13], fault-proneness [14] and so on.

2.1.1 LCOM by Chidamber and Kemerer

The original LCOM is defined by Chidamber and Kemerer [6]. Their approach is based on whether the each pair of methods are sharing an attribute or not in a class. Those attribute-sharing relationships mean that different methods are accessing to the same attribute. Chidamber and Kemerer define LCOM as the number of method-pairs which are not sharing any attributes. LCOM value represents a weakness of functional connections among methods in a class. For the sake of convenience, we will call the metric “LCOM1.”

Here is a formal definition of LCOM1.

Definition 1 (LCOM1):
Given a class C. Let M be the set of all methods in C, and A be the set of all attributes in C. LCOM1 value of C is defined as $LCOM1(C) = |P|$, where

$$P = \{ \{m_1, m_2\} \mid m_1, m_2 \in M, \exists a \in A \text{ s.t. both of } m_1 \text{ and } m_2 \text{ access to } a \}$$.
In the literature [7], Chidamber and Kemerer propose another definition of LCOM. We will call it “LCOM2.” LCOM2 takes into account the following two different factors: (1) the number of method-pairs which are “not sharing” any attributes, and (2) the number of method-pairs which are “sharing” an attribute. The above two factors have complementary meanings each other. The former is a weakening factor for cohesion, while the latter is a strengthening factor for one. LCOM2 combines those factors.

The following is a formal definition of LCOM2.

Definition 2 (LCOM2):
Given a class C. Let M be the set of all methods in C, and A be the set of all attributes in C. LCOM2 value of C is defined as follows:

\[ LCOM2(C) = \begin{cases} 
|P| - |Q|, & \text{if } |P| > |Q|; \\
0, & \text{otherwise.}
\end{cases} \]

where P is the set described in Def. 1, and

\[ Q = \{ (m_1, m_2) | m_1, m_2 \in M, \exists a \in A \text{ s.t. both of } m_1 \text{ and } m_2 \text{ access to } a \}. \]

2.1.2 LCOM by Hitz and Montazeri

Hitz and Montazeri propose a different LCOM [8] using the graph theory. Their approach is also based on the above attribute-sharing relationships, and those relationships are represented by an undirected graph. Now a class corresponds to an undirected graph, where each vertex represents each method, and each edge means each attribute-sharing relationship. Hitz and Montazeri define LCOM as the number of “connected components” [15] in the undirected graph. We will call it “LCOM3.” LCOM3 value indicates the number of disjoint sets of methods in terms of the attribute-sharing relationship.

Here is a formal definition of LCOM3.

Definition 3 (LCOM3):
Given a class C. Let M be the set of all methods in C, and A be the set of all attributes in C. Consider an undirected graph \( G_C = (V, E) \) with \( V = M \) and

\[ E = \{ (m_1, m_2) | m_1, m_2 \in M, \exists a \in A \text{ s.t. both of } m_1 \text{ and } m_2 \text{ access to } a \}. \]

LCOM3 value of C is defined as \( LCOM3(C) = k(G_C) \), where \( k(G_C) \) is the number of connected components in \( G_C \). \( \square \)

Hitz and Montazeri also discuss some effects of “access-methods” upon LCOM values [9]. Access-methods are methods to provide reading/writing-accesses to attributes. These methods are sometimes called “accessors” or “getters and setters.” When two or more methods access to an attribute via “access-methods”, we have no attribute-sharing relationships, because the above attribute-sharing relationships are based on “direct” accesses to attributes by methods. Hitz and Montazeri propose another LCOM taking into account such accesses via “access-methods.” We will call it “LCOM4.”

The following is a formal definition of LCOM4.

Definition 4 (LCOM4):
Given a class C. Let M be the set of all methods in C, and A be the set of all attributes in C. Consider an undirected graph \( G_C = (V, E) \) with \( V = M \) and

\[ E = \{ (m_1, m_2) | m_1, m_2 \in M, \\
(\exists a \in A \text{ s.t. both of } m_1 \text{ and } m_2 \text{ access to } a) \\
or (m_1 \text{ invokes } m_2) \}. \]

LCOM4 value of C is defined as \( LCOM4(C) = k(G_C) \), where \( k(G_C) \) is the number of connected components in \( G_C \). \( \square \)

2.1.3 LCOM by Henderson-Sellers

Henderson-Sellers proposes LCOM in a different way [10]. We will call it “LCOM5.” LCOM5 value is characterized by the following two cases: (i) if each method accesses to “all” attributes, the LCOM5 value is 0; (ii) if each method accesses to “only one” attribute, the LCOM5 value is 1. The case (i) means the most cohesive class, while the case (ii) means a less cohesive one. LCOM5 represents cohesion-levels using these two cases as benchmarks.

Here is a formal definition of LCOM5.

Definition 5 (LCOM5):
Given a class C. Let M be the set of all methods in C, and A be the set of all attributes in C. For each \( a \in A \), let \( \mu(a) \) be the number of methods accessing to the attribute \( a \).

LCOM5 value of C is defined as follows:

\[ LCOM5(C) = \frac{1}{|M| - 1} \left( |M| - \frac{1}{|A|} \sum_{a \in A} \mu(a) \right). \]

\( \square \)

2.2 Information Flow Based Cohesion (ICH)

Lee et al. propose a cohesion metric based on information flows between methods [11]. The approach focuses on the number of method-invocations weighted by the number of their parameters. Using the number of method parameters, the metric takes into account a strength of link between methods. The metric is called “Information flow based Cohesion (ICH).”

The following is a formal definition of ICH.

Definition 6 (ICH):
Given a class C. Let M be the set of all methods in C. For \( m_1, m_2 \in M \), let \( \nu(m_1, m_2) \) be the number of invocations of \( m_2 \) by \( m_1 \), and \( \pi(m_2) \) be the number of \( m_2 \)'s parameters.

\[^{\dagger}\text{A connected component is a maximal sub graph in which all vertexes are reachable each other.}\]
ICH value of \( C \) is defined as follows:

\[
ICH(C) = \sum_{m_1 \in M} \sum_{m_2 \in M} (1 + \pi(m_2)) \cdot \nu(m_1, m_2).
\]

### 2.3 Tight Class Cohesion (TCC) and Loose Class Cohesion (LCC)

Bieman and Kang propose a couple of class cohesion metrics which are called “Tight Class Cohesion (TCC)” and “Loose Class Cohesion (LCC)” [12]. TCC represents a density of attribute-sharing relationships between public methods in a class. LCC represents a density of extended attribute-sharing relationships between public methods, where those extended relationships are constructed by the transitive closure of the above attribute-sharing relationships. These metrics have studied on usefulness as indicators of reusability for object classes [12].

We present definitions of TCC and LCC.

**Definition 7** (TCC, LCC):

Given a class \( C \). Let \( M_p \) be the set of all public methods in \( C \), and \( A \) be the set of all attributes in \( C \).

When a method \( m \) uses an attribute \( a \) in the method body, we say “\( m \) directly accesses to \( a \)”.

If a method \( m \) invokes another method \( m_1 \), \( m_1 \) invokes \( m_2 \), \ldots, \( m_{n-1} \) invokes \( m_n \) (\( n \geq 1 \)) and \( m_n \) directly accesses to an attribute \( a \), then we say “\( m \) indirectly accesses to \( a \)”.

Now consider the following two sets:

\[
T = \{ (m, m') \mid m, m' \in M_p, \exists a \in A \text{ s.t. } m \text{ and } m' \text{ directly or indirectly access to } a \},
\]

and

\[
L = \{ (m, m') \mid \exists m_i \in M_p \text{ s.t. } m = m_1, m' = m_k, k > 1, (m_j, m_{j+1}) \in T (j = 1, \ldots, k - 1) \}.
\]

**TCC and LCC values** of \( C \) are defined as follows:

\[
TCC(C) = \frac{|T|}{\left\lfloor \frac{|M_p|}{2} \right\rfloor}, \quad LCC(C) = \frac{|T| + |L|}{\frac{|M_p|}{2}},
\]

where \( \left\lfloor \frac{|M_p|}{2} \right\rfloor = |M_p|(|M_p| - 1)/2 \).

### 3. A New Class Cohesion Metric

#### 3.1 Motivation

We have introduced eight metrics: \( LCOM1\sim5 \), \( ICH \), \( TCC \) and \( LCC \). Table 1 shows a summary of them. \( LCOM1\sim4 \) are based on the numbers of pairs/sets of methods connected by attribute-sharing relationships. \( LCOM5 \) presents a density of accesses to attributes by methods. \( ICH \) shows the number of method-invocations weighted by the number of method parameters. \( TCC \) and \( LCC \) are based on densities of attribute-sharing relationships among methods. However, they do not consider “extents” of attribute-sharing relationships among methods, in other words, “sizes of cohesive-parts” in a class. Since the higher cohesive class would have the larger cohesive-parts, such aspect is also one of the cohesion aspects to be measured. This section proposes a cohesion metric focusing on cohesive-part size.

#### 3.2 Preliminaries

Preliminary to the development of our metric, we define several underlying notions.

At first, we define a mathematical relation between methods through method-invocations.

**Definition 8** (binary relation on methods):

Given a class. Let \( M \) be the set of all methods in the class.

We define a binary relation \( S \) as follows:

\[
S = \{ (m_1, m_2) \mid m_1, m_2 \in M, \ m_1 \text{ invokes } m_2 \}.
\]

Now we can obtain the reflective transitive closure \( S^* \) as follows:

\[
S^* = \left\{ (m_1, m_2) \mid m_1, m_2 \in M, \ (m_1 = m_2) \lor \left( m_1 S^n m_2 \right) \right\},
\]

where \( S^n = S \circ S (n \geq 2) \), and \( S^1 = S \); “\( \circ \)” indicates the composition of relations [16].

\( S^* \) represents directly or indirectly method-invocations.

Now we define a relationship between a method and an attribute.
 Definition 9 (accesses to attributes by methods):
Given a class. Let $M$ be the set of all methods, and $A$ be the set of all attributes, in the class. For any $m \in M$, $a \in A$, we define a predicate $ac$ as follows:

\[
ac(m, a) \overset{\text{def}}{\iff} \exists m' \in M \text{ s.t. } (m \text{ accesses } a) \land (m' \text{ accesses } a'),
\]

where an “access” means a direct access to an attribute by a method.

The predicate $ac$ considers not only direct accesses to attributes but also indirect accesses to ones via access-methods. Using the predicate, we introduce a graph model.

 Definition 10 (association-graph):
Given a class. Let $M$ be the set of all methods, and $A$ be the set of all attributes, in the class. We define the association-graph as an undirected graph $G_a = (V, E)$, where $V = M$ and $E = \{ m_1, m_2 \mid m_1, m_2 \in M, m_1 \neq m_2, \exists a \in A \text{ s.t. } ac(m_1, a) \land ac(m_2, a) \}$. (1)

When two or more methods access to one attribute, those methods seem to share the attribute. The association-graph $G_a$ represents those attribute-sharing relationships between methods in a class. In $G_a$, if there is a path from a method to another method, those methods share an attribute directly or indirectly. Those relationships correspond to the reachabilities in the graph.

 Definition 11 (set of reachable methods):
Given a class. Let $M$ be the set of all methods, and $A$ be the set of all attributes, in the class. Consider the association-graph $G_a = (V, E)$, where $V = M$ and $E$ is described in Eq. (1).

For each method $m_i \in M (i = 1, \ldots, |M|)$, define the set of reachable methods, $R_a(m_i)$, as follows:

\[
R_a(m_i) = \{ m_j \mid \exists m_1, \ldots, m_p \in M \text{ s.t. } (m_i = m_1) \land (m_j = m_p) \land \{\exists m_k \land m_k \neq m_i \land \exists t \in \{1, \ldots, p - 1\} \} \}.
\]

$R_a(m_i)$ is the set of all methods which are reachable by $m_i$ in $G_a$. The methods belonging to $R_a(m_i)$ seem to be associated with $m_i$ through their attribute-sharing relationships. In other words, those methods are in the “chain of links” where methods are directly or indirectly linked through attribute-sharing relationships. So the methods are functionally connected by the chain of links, and we should not decompose the set into two or more smaller sets. $R_a(m_i)$ seems to form a cohesive-part of the class including $m_i$, and $|R_a(m_i)|$ is involved in the extent of attribute-sharing relationships.

3.3 Definition

Using the association-graph and the sets of reachable methods, we propose a metric for measuring class cohesion, which is called “Association-Extent based Class Cohesion (AECC).”

 Definition 12 (AECC):
Given a class $C$. Let $M$ be the set of all methods, and $A$ be the set of all attributes, in $C$. Consider the association-graph $G_a = (V, E)$ for $C$, where $V = M$ and $E$ is in Eq. (1). For each method $m \in M$, let $R_a(m)$ be the set of all methods which are reachable by $m$ in $G_a$ (see Eq. (2)).

Define Association-Extent based Class Cohesion (AECC) value of $C$ as follows:

\[
AECC(C) = \max_{m \in M} \left[ \frac{|R_a(m)|}{|M| - 1} \right],
\]

(3)

3.4 Meanings of the Proposed Metric

For each method $m$, $|R_a(m)|/(|M| - 1)$ denotes the percentage of methods to be reachable by $m$ in the association-graph; “−1” in the numerator represents excluding $m$ itself from the percentage calculation. $AECC$ is the maximum percentage for all methods in the class. Since $R_a(m)$ forms a cohesive-part in which all methods are directly or indirectly linked through attribute-sharing relationships, $|R_a(m)|/(|M| - 1)$ represents the relative size of the cohesive-part including $m$. In other words, $|R_a(m)|/(|M| - 1)$ denotes the extent of attribute-sharing relationships. That is, $AECC$ quantifies the maximum extent of attribute-sharing relationships among methods in the class. Since the higher cohesive class would have the larger cohesive-parts, $AECC$ represents an aspect of class cohesion.

Although $AECC$ is similar to $TCC$ and $LCC$, $AECC$ is essentially different from them. $TCC$ and $LCC$ calculate “densities” of attribute-sharing relationships among methods. However, $AECC$ represents the maximum “extent” of attribute-sharing relationships. While the “density” is based on the number of attribute-sharing relationships, the “extent” corresponds to the connectivity among methods via attribute-sharing relationships.

3.5 Examples

We now present a simple example of calculating $AECC$ value. Consider a class shown in Fig. 1, which has seven methods $M = \{m_1, \ldots, m_7\}$ and four attributes $A = \{a_1, \ldots, a_4\}$. Fig. 2 shows its structure$^1$.

Here the following ten predicates are true:

\[ac(m_1, a_1), ac(m_1, a_2), ac(m_2, a_1), ac(m_3, a_2).\]

\[\text{For the sake of simplicity, we omit the existence of the super class \texttt{java.lang.Object}.}\]

\[\text{Footnote 1:}
\]

\[\text{\texttt{For the sake of simplicity, we omit the existence of the super class \texttt{java.lang.Object}.}}\]
Table 2 Metric values of the classes in Fig. 2 and Fig. 4.

<table>
<thead>
<tr>
<th>metric</th>
<th>Fig. 2</th>
<th>Fig. 4</th>
<th>Fig. 2 → Fig. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCOM1</td>
<td>17</td>
<td>17</td>
<td>no change</td>
</tr>
<tr>
<td>LCOM2</td>
<td>13</td>
<td>13</td>
<td>no change</td>
</tr>
<tr>
<td>LCOM3</td>
<td>3</td>
<td>3</td>
<td>no change</td>
</tr>
<tr>
<td>LCOM4</td>
<td>2</td>
<td>1</td>
<td>up</td>
</tr>
<tr>
<td>LCOM5</td>
<td>0.833</td>
<td>0.833</td>
<td>no change</td>
</tr>
<tr>
<td>ICH</td>
<td>3</td>
<td>3</td>
<td>no change</td>
</tr>
<tr>
<td>TCC</td>
<td>0.429</td>
<td>0.333</td>
<td>down</td>
</tr>
<tr>
<td>LCC</td>
<td>0.429</td>
<td>1.0</td>
<td>up</td>
</tr>
<tr>
<td>AECC</td>
<td>0.5</td>
<td>1.0</td>
<td>up</td>
</tr>
</tbody>
</table>
4. Evaluations

This section presents evaluations of our AECC in the both of qualitative and quantitative ways. The qualitative evaluation will describe that AECC satisfies mathematical necessary conditions of cohesion metrics, based on a mathematical framework. The quantitative evaluation will show that AECC gives the metric values without depending on the other existing metrics described in Sect. 2.

4.1 Qualitative Evaluation

4.1.1 Mathematical Framework

Briand, Morasca and Basili propose a mathematical framework (BMB framework) including some properties to be satisfied by several types of software metrics [17]. The supported types of metrics are “size,” “length,” “complexity,” “coupling” and “cohesion.” Note that BMB framework provides necessary conditions of software metrics, because the framework does not include the all properties to be satisfied by those metrics. We will use it for checking some mathematical necessary conditions of metrics.

In BMB framework, a software is represented by a graph model in which vertexes are corresponding to the components, and edges are corresponding to coupling relationships between the components. BMB framework suggests the following four properties to be satisfied by cohesion metrics. For the sake of convenience, we will write the cohesion of a class C as “cohe(C).”

**Property 1:**

For any class C, $\text{cohe}(C) \in [0, \text{max}]$, where max is a positive constant number.

**Property 2:**

Let $G = (V, E)$ be the graph model of a class C, where V and E are the vertex set and the edge set, respectively. Then $E = \emptyset \Rightarrow \text{cohe}(C) = 0$.

**Property 3:**

Consider two classes C and C’ whose models are $G = (V, E)$ and $G’ = (V, E’)$, respectively. Then $E \subseteq E’ \Rightarrow \text{cohe}(C) \leq \text{cohe}(C’)$.

**Property 4:**

Consider two classes C₁ and C₂ whose models are $G₁ = (V₁, E₁)$ and $G₂ = (V₂, E₂)$, respectively. Let $C₁₂$ be a class whose model is $G₁₂ = (V₁ \cup V₂, E₁ \cup E₂)$, i.e., $C₁₂$ is composed of $C₁$ and $C₂$. Then

$$\forall [u, v] \in E₁ \cup E₂ \ [u \in V₁ \cap V₂ \iff v \in V₁ \cap V₂]$$

$$\Rightarrow \max \{ \text{cohe}(C₁), \text{cohe}(C₂) \} \geq \text{cohe}(C₁₂).$$

Property 1 represents a nonnegativity and a normalization. The nonnegativity is a basic property to be satisfied by any of mathematical measures [18]. The normalization brings us meaningful comparisons between different metrics.

Property 2 is also a basic property of mathematical measures: the measure of empty set is null. In our context, if there is no relationships among components in a class, then the class cohesion is null.

Property 3 corresponds to a monotonicity. When some relationships between components are added in a class,
cohesion-level does not go down.

Property 4 considers a situation combining a class with another class, where they have no common component. Such situation means that two classes cohabit, but they have no relationship each other. Property 4 says that such combined class does not have a greater cohesion than the maximum of the original classes’ cohesion.

4.1.2 Results

AECC satisfies the above four properties (see Appendix for their proofs), and be a metric holding necessary conditions of cohesion metric. We will consider AECC to be one of reasonable class cohesion metrics, and compare it with the other existing metrics in the following section.

4.2 Quantitative Evaluation

We have seen AECC holds the above necessary conditions of cohesion metric. We next present some measurement data of practical object classes, and evaluate AECC using those data, especially we will show that AECC measures classes without depending on the other existing metrics, in other words, it is not redundant metric.

Table 4 presents metric values of Java classes which are selected from “Sun J2SE SDK version 1.3.1” randomly.

Unfortunately any sufficient conditions of cohesion metric have never been found nor proposed. Since we have no sufficient condition, it is difficult to present an objective and reliable discussion on the validity of AECC as a cohesion metric, even if we use many experimental data. (While highly reusable classes such as String, StringBuffer and so on, seem to have relative high value of AECC, they can not be any grounds for the validity as cohesion metric; Rather they would be materials for investigating a usefulness of AECC.) So we have checked AECC by mathematical properties of cohesion metric in the above. As a quantitative evaluation of AECC, we analyze dependencies between metrics using experimental data shown in Table 4, and we show AECC is not redundant metric. For example, we now see three classes shown in the head of Table 4. Although the model of AECC is similar to the models of TCC and LCC, those experimental data show a different tendency between AECC and TCC/LCC (e.g., see Fig. 7). In order to show such actual differences in tendency between metrics, we use the following correlation analysis.

4.2.1 Correlation Coefficient

When some different software metrics measure a software attribute, they have to capture different aspects of the software attribute independently. In other words, metrics should not depend on each other for providing their measurements. If there is a strong correlation [19] between two metrics, one of them is a redundant metric. We can explore such relationships between metrics by calculating their correlation coef-
ficients.

The following is a brief description of calculating correlation coefficients for verifying metrics.

Given two metrics and \( N \) sample data (software). Measure each of sample data using those metrics, then obtain 2-dimensional vector \( x_i = (x_{i1}, x_{i2}) \) for each data \( (i = 1, \ldots, N) \), where \( x_{i1} \) and \( x_{i2} \) are the metric values for the \( i \)-th sample data, respectively. Now the correlation coefficient \( r \) is calculated as follows:

\[
r = \frac{\sum_{i=1}^{N} (x_{i1} - \bar{x}_1)(x_{i2} - \bar{x}_2)}{\sqrt{\sum_{i=1}^{N} (x_{i1} - \bar{x}_1)^2 \sum_{i=1}^{N} (x_{i2} - \bar{x}_2)^2}},
\]

where \( \bar{x}_1 = (\sum_{i=1}^{N} x_{i1})/N \) and \( \bar{x}_2 = (\sum_{i=1}^{N} x_{i2})/N \).

The higher value of \(|r|\) means the stronger correlation between two metrics. Now prepare a threshold value \( \tau \). If \(|r| \geq \tau \), we will consider those two metrics are dependent on each other, and one of them is a redundant metric.

4.2.2 Experiments

We perform the following experiments in order to show that \( AECC \) measures cohesion without depending on the other existing metrics (see Sect. 2), in other words, \( AECC \) is not redundant metric.

Prepare a set of sample data (Java classes). Measure cohesion using the above eight metrics and \( AECC \). For each pair of \( AECC \) and the others, (i.e., for the following eight pairs: \{\( AECC, LCOM1 \), \{\( AECC, LCOM2 \), \{\( AECC, LCOM3 \), \{\( AECC, LCOM4 \), \{\( AECC, LCOM5 \), \{\( AECC, ICH \), \{\( AECC, TCC \), \{\( AECC, LCC \), \} calculate their correlation coefficients.

We now use \( \tau = 0.8 \) [19]. If there is a correlation coefficient is greater than or equal to \( \tau \), we will regard \( AECC \) as a redundant metric. Otherwise, we will consider that \( AECC \) can provide its measurements without depending on any of the other existing metrics.

Now our set of sample data (Java classes) is shown in Table 4.

4.2.3 Results

Tables 5 shows the correlation coefficients calculated in the above experiment\(^* \). From this table, we have no metric whose correlation coefficient with \( AECC \) is grater than or equal to \( \tau(=0.8) \), i.e., no metric who has a strong correlation with \( AECC \):

- \( LCOM1 \sim LCOM5 \) and \( ICH \) have some low correlation coefficients (\(|r| = 0.085906 \sim 0.259502) \).
- \( TCC \) and \( LCC \) have some middle-level correlations (\(|r| = 0.695496, 0.697812) \), but not strong correlations. The reason they have such correlations is because their underlying model is similar to \( AECC \)'s model: The models of \( TCC \) and \( LCC \) focus on “densities” of attribute-sharing relationships while \( AECC \)'s model focuses on “extents” of those relationships.

Thus \( AECC \) is not redundant metric, and will be an important one. It measures an aspect of cohesion which is not supported by the other existing eight metrics. \( AECC \) can collaborate with those existing metrics in measuring class cohesion, and will contribute to more accurate measurement.

5. Conclusion and Future Work

A class cohesion metric, “Association-Extent based Class Cohesion (AECC),” has been proposed in this paper. \( AECC \) measures class cohesion using the maximum size of cohesive-parts, which is an aspect has not been supported by the other existing metrics. \( AECC \) has been evaluated in both of qualitative and quantitative ways: it has been showed that

- \( AECC \) satisfies mathematical conditions of cohesion metrics described in \( BMB \) framework,
- \( AECC \) presents cohesion values without depending on any of the other existing metrics: \( LCOM1 \sim 5, ICH, TCC \) and \( LCC \).

Therefore, \( AECC \) is a reasonable class cohesion metric, and not redundant metric. It can collaborate with the other existing metrics in measuring class cohesion, and will contribute to more accurate measurement.

Our future works include investigations into

(1) practical usefulness of \( AECC \) and other metrics, such as predictors of software maintenance cost using class cohesion metrics,
(2) relationships between reusability of software components and their cohesion, and so on.

\(^*\)For the lack of space, we omit the metric values for each of sample data.
Acknowledgement

The authors would like to thank the anonymous reviewers for their thoughtful comments and helpful suggestions. This research has been supported by the Kayamori Foundation of Information Science Advancement.

References


Appendix: Proofs

The followings are the proofs of that AECC satisfies the four properties of cohesion metrics described in Sect. 4.1.1 [20]. In these proofs, we will consider the association-graph to be the graph model of a class.

Proof for Property 1:

Given a class $C$. Let $M$ be the set of all methods in $C$.
(i) Case $|M|=1$:
From Eq. (3), we obtain $AECC(C)=0$.
(ii) Case $|M|>1$:
For each $m \in M$, let $R_a(m)$ be the set of methods reachable by $m$ (see Eq. (2)). From the definition, we have $0 \leq |R_a(m)| \leq |M| - 1$, for each $m \in M$. Thus, from Eq. (3), we obtain $0 \leq AECC(C) \leq 1$.

Therefore, we can get $AECC(C) \in [0, 1]$, for any class $C$.

Proof for Property 2:

Given a class $C$. Let $M$ be the set of all methods in $C$, and $G_a=(V,E)$ be the association-graph of $C$.
(i) Case $|M|=1$:
From Eq. (3), we obtain $AECC(C)=0$.
(ii) Case $|M|>1$:
We assume $E=\emptyset$. For each $m \in M$, let $R_a(m)$ be the set of methods reachable by $m$ (see Eq. (2)). Since $E=\emptyset$, we have $|R_a(m)|=0$ for each $m \in M$. Thus, we obtain $AECC(C)=0$.

Therefore, we can get $AECC(C)=0$ for any class $C$ such that $E=\emptyset$.

Proof for Property 3:

Given a class $C$. Let $M$ be the set of all methods in $C$.
(i) Case $|M|=1$:
Since we have only one vertex (method) in the graph, we could not add any edges to the association-graph of $C$, and each vertex is not allowed having a self-loop (an edge from a vertex to the same vertex). Thus, the class $C'$, described in the property 3, must be the same as $C$. Therefore, we can obtain $AECC(C)=AECC(C')$.
(ii) Case $|M|>1$:
Let $G_a(C)=(V,E)$ be the association-graph of $C$. Now we can consider a class $C'$ whose association-graph is $G'_a(C')=(V,E')$ where $E \subseteq E'$. For each $m \in M$, let $R_a(m)$ and $R'_a(m)$ be the sets of reachable methods by $m$ in $G_a$ and $G'_a$, respectively (see Eq. (2)).

Since $E \subseteq E'$, we obtain $R_a(m) \subseteq R'_a(m)$, for each $m \in M$. Thus, we have $|R_a(m)| \leq |R'_a(m)|$, and we get

$$\max_{m \in M} |R_a(m)| \leq \max_{m \in M} |R'_a(m)|.$$

i.e., $AECC(C) \leq AECC(C')$, for each $m \in M$.

Therefore, we can get $AECC(C) \leq AECC(C')$, for all pairs of classes such that $E \subseteq E'$.

Proof for Property 4:
Given a pair of classes \( C_1 \) and \( C_2 \), and consider a class \( C_{12} \) composed of \( C_1 \) and \( C_2 \). Let \( M_1, M_2 \) and \( M_{12} \) (= \( M_1 \cup M_2 \)) be the sets of all methods in \( C_1, C_2 \) and \( C_{12} \), respectively. Let \( G_{a_1} = (V_1, E_1), G_{a_2} = (V_2, E_2) \) and \( G_{a_{12}} = (V_1 \cup V_2, E_1 \cup E_2) \) be the association-graphs of \( C_1, C_2 \) and \( C_{12} \), respectively. For each \( m_1 \in M_1 \), let \( R_{a_1}(m_1) \) be the set of reachable methods by \( m_1 \) in \( G_{a_1} \) (see Eq. (2)). Similarly, let \( R_{a_2}(m_2) \) and \( R_{a_{12}}(m_{12}) \) be the sets of reachable methods by \( m_2 \in M_2 \) and \( m_{12} \in M_{12} \), respectively.

(i) Case \( |M_1| = |M_2| = 1 \):
Since \( M_{12} = M_1 \cup M_2 \), we have \( |M_{12}| = 1 \). From Eq. (3), we obtain \( AECC(C_1) = AECC(C_2) = AECC(C_{12}) = 0 \). Thus, we get

\[
\max \{ AECC(C_1), AECC(C_2) \} = AECC(C_{12}).
\]

(ii) Case \( |M_1| > 1 \) or \( |M_2| > 1 \):
This case could be \( |M_1| + |M_2| \geq |M_{12}| \), and we put \( |M_1| + |M_2| - |M_{12}| = p \geq 0 \).
Let \( m_{1,i} (i = 1, \ldots, |M_1|), m_{2,j} (j = 1, \ldots, |M_2|) \) and \( m_{12,k} (k = 1, \ldots, |M_{12}|) \) be the methods in \( C_1, C_2 \) and \( C_{12} \), respectively. Now we put the relations among \( m_{1,i}, m_{2,j} \) and \( m_{12,k} \) as follows, without loss of generality:

\[
m_{12,k} = \begin{cases} m_{1,k} , & (k = 1, \ldots, |M_1| - p), \\ m_{2,k} - |M_1| + p , & \text{otherwise}, \end{cases}
\]

where \( m_{2,k} - |M_1| + p \) = \( m_{1,k} \) (\( k = |M_1| - p, \ldots, |M_1| \)).

From the assumption of the property 4, we have \( \forall \{u, v\} \in E_1 \cup E_2 \ [u \in V_1 \cap V_2 \iff v \in V_1 \cap V_2] \). That is, \( G_{a_{12}} \) has no path from a method in \( C_1 \) to one in \( C_2 \), vice versa. Thus we obtain the following equation:

\[
R_{a_{12}}(m_{12,k}) = \begin{cases} R_{a_1}(m_{1,k}) , & (k = 1, \ldots, |M_1| - p), \\ R_{a_2}(m_{2,k} - |M_1| + p) , & \text{otherwise}. \end{cases}
\]

Since \( |M_1|, |M_2| \leq |M_{12}| \), we obtain

\[
\frac{|R_{a_{12}}(m_{12,k})|}{|M_{12}| - 1} \leq \frac{|R_{a_1}(m_{1,k})|}{|M_1| - 1}, \quad (k = 1, \ldots, |M_1|),
\]

and

\[
\frac{|R_{a_{12}}(m_{12,k})|}{|M_{12}| - 1} \leq \frac{|R_{a_2}(m_{2,j})|}{|M_2| - 1}, \quad (k = |M_1| - p + 1, \ldots, |M_{12}|; j = k - |M_1| + p).
\]

Thereby we get

\[
\max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_{12}}(m_{12,k})|}{|M_{12}| - 1} \right] \leq \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_1}(m_{1,k})|}{|M_1| - 1} \right] = AECC(C_1),
\]

and

\[
\max_{k=|M_1|-p+1, \ldots, |M_{12}|} \left[ \frac{|R_{a_{12}}(m_{12,k})|}{|M_{12}| - 1} \right] \leq \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_2}(m_{2,j})|}{|M_2| - 1} \right] = AECC(C_2).
\]

Therefore, we get

\[
AECC(C_{12}) = \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_{12}}(m_{12,k})|}{|M_{12}| - 1} \right] = \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_1}(m_{1,k})|}{|M_1| - 1} \right],
\]

\[
= \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_1}(m_{1,k})|}{|M_1| - 1} \right] = \max_{k=1, \ldots, |M_1|} \left[ \frac{|R_{a_2}(m_{2,j})|}{|M_2| - 1} \right],
\]

\[
\leq \max \{ AECC(C_1), AECC(C_2) \},
\]

for all classes such that \( \forall \{u, v\} \in E_1 \cup E_2 \ [u \in V_1 \cap V_2 \iff v \in V_1 \cap V_2] \). \qed
Matu-Tarow Noda received the Dr. of Science from Osaka City University, Osaka, Japan, in 1969. Since 1971, he has been with Ehime University. He is currently a professor of the Faculty of Engineering, Ehime University. He also is a director of Center of Information Technology, Ehime University (CITE). His primary research interests are in algorithm developments of symbolic-numeric combined computation, and Internet security in mobile environments. He received the Best Paper Award from IPSJ in 1992. He is the president of Japan Society for Symbolic and Algebraic Computation (JSSAC). He is a member of IPSJ, ACM, Japan Society for Industrial and Applied Mathematics (JSIAM), and International Association for Mathematics and Computers in Simulation (IMACS).