Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Improved binary similarity measures for software modularization*

Rashid NASEEM[†1], Mustafa Bin Mat DERIS[1], Onaiza MAQBOOL[2], Jing-peng LI[3], Sara SHAHZAD[4], Habib SHAH[5]

([1]*Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja 86400, Malaysia*)

([2]*Department of Computer Science, Quaid-i-Azam University, Islamabad 45320, Pakistan*)

([3]*Division of Computer Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK*)

([4]*Department of Computer Science, University of Peshawar, Peshawar 25120, Pakistan*)

([5]*Faculty of Computer and Information Systems, Islamic University Madina, Madina POBox 170, KSA*)

[†]E-mail: rnsqau@gmail.com

Received Oct. 30, 2015; Revision accepted Apr. 12, 2016; Crosschecked Aug. 4, 2017

Abstract: Various binary similarity measures have been employed in clustering approaches to make homogeneous groups of similar entities in the data. These similarity measures are mostly based only on the presence or absence of features. Binary similarity measures have also been explored with different clustering approaches (e.g., agglomerative hierarchical clustering) for software modularization to make software systems understandable and manageable. Each similarity measure has its own strengths and weaknesses which improve and deteriorate the clustering results, respectively. We highlight the strengths of some well-known existing binary similarity measures for software modularization. Furthermore, based on these existing similarity measures, we introduce several improved new binary similarity measures. Proofs of the correctness with illustration and a series of experiments are presented to evaluate the effectiveness of our new binary similarity measures.

Key words: Binary similarity measure; Binary features; Combination of measures; Software modularization
http://dx.doi.org/10.1631/FITEE.1500373 **CLC number:** TP311

1 Introduction

Clustering is an approach that makes clusters of similar entities in the data. Entities in a cluster are similar to each other (based on characteristics or features) while they are distinct from entities in other clusters. In the software domain, an important application of clustering is to modularize a software system or to recover the module architecture or components of the software systems by clustering the software entities, e.g., functions, files, or classes, in the source code. Recovery is very important when no up-to-date documentation of a software system is available (Shtern and Tzerpos, 2014). Besides clustering, other approaches have also been used for software modularization, e.g., supervised clustering (Hall *et al.*, 2012), optimization techniques (Praditwong *et al.*, 2011), role-based recovery (Dugerdil and Jossi, 2008), graph-based techniques (Bittencourt and Guerrero, 2009), association-based approaches (Vasconcelos and Werner, 2007), spectral method (Xanthos and Goodwin, 2006), rough set theory (Jahnke, 2004), concept analysis (Tonella, 2001), and visualization tools (Synytskyy *et al.*, 2005).

* Project supported by the Office of Research, Innovation, Commercialization and Consultancy (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia (No. U063)

ORCID: Rashid NASEEM, http://orcid.org/0000-0002-4952-8100

A key activity in software clustering consists of gathering the entities from source code of software systems into meaningful and independent modules. The process of software clustering usually starts with the selection of entities and their features by parsing the source code of software systems. Then entities are organized into cohesive clusters by employing a particular clustering algorithm (Mitchell and Mancoridis, 2006).

Agglomerative hierarchical clustering (AHC) algorithms have been widely used by researchers to cluster software systems (Wiggerts, 1997; Anquetil and Lethbridge, 1999; Mitchell, 2006; Maqbool and Babri, 2007; Patel *et al.*, 2009; Shtern and Tzerpos, 2010; Muhammad *et al.*, 2012). AHC comprises two main factors, a similarity measure to find the association between two entities and a linkage method to update the similarity values between entities in each iteration. However, selection of a similarity measure is an important factor in AHC (Jackson *et al.*, 1989; Cui and Chae, 2011), which has a major influence on the clustering results (Naseem *et al.*, 2010; Shtern and Tzerpos, 2012).

There exist a large number of binary similarity measures (Cheetham and Hazel, 1969; Seung-Seok *et al.*, 2010). Nevertheless, for software modularization, the comparative studies have reported that Jaccard (JC) binary similarity measure produces better clustering results (Davey and Burd, 2000; Tzerpos and Holt, 2000; Lung *et al.*, 2004; Shtern and Tzerpos, 2012). In our previous study (Naseem *et al.*, 2010), we proposed a new binary similarity measure, called JaccardNM (JNM), which can overcome some deficiencies of the JC binary similarity measure. We also examined the Russell&Rao (RR) binary similarity measure for software modularization for the first time, and found that it can generate better results as compared to JC and JNM binary similarity measures for some of the test software systems. In Naseem *et al.* (2013), we proposed the COUSM (cooperative only update similarity matrix) clustering algorithm, which combines two similarity measures in a single clustering process based on AHC.

In this paper, we explore the integration of the existing binary similarity measures for AHC algorithms using linkage methods (e.g., complete linkage (CL), single linkage (SL), and weighted average linkage (WL) methods). For example, we select the JC similarity measure, which produces a relatively large number of clusters (Saeed *et al.*, 2003; Maqbool and Babri, 2004), and the JNM binary similarity measure, which takes a smaller number of arbitrary decisions (Naseem *et al.*, 2010) during the clustering process. During the clustering process, creating a large number of clusters means that a clustering approach may create compact clusters, hence improving the quality of clustering results (Maqbool and Babri, 2007). Arbitrary decision is the arbitrary clustering of two entities when there exist more than two equally similar entities; hence, arbitrary decisions create problems and reduce the quality of clustering results (Maqbool and Babri, 2007; Naseem *et al.*, 2010). This analysis leads us to introduce better binary similarity measures by combining the JC and JNM measures.

We focus mainly on the identification of the strengths of the existing binary similarity measures. Moreover, the improved similarity measures are based on the integration of JC, JNM, and RR similarity measures. While in our previous studies (Naseem *et al.*, 2010; 2011), the main focus was to explore the deficiencies (i.e., creating a large number of equal similarity values and giving no importance to a pair of entities sharing a large number of features) of some well-known binary similarity measures and then to solve these deficiencies by adding the total proportion of features to the denominator in the JC similarity measure.

The contributions of this paper can be summarized as follows:

1. We analyze the Jaccard and JaccardNM (Naseem *et al.*, 2010) similarity measures for binary features and compare their strengths.

2. We integrate the strengths of existing binary similarity measures to form new binary similarity measures for software clustering.

3. We introduce four improved binary similarity measures that yield more effective solutions than the existing binary similarity measures. They are

(1) JCJNM: add the JC and JNM binary similarity measures;

(2) JCRR: add the JC and RR binary similarity measures;

(3) JNMRR: add the JNM and RR binary similarity measures;

(4) JCJNMRR: add the JC, JNM, and RR binary similarity measures.

4. We provide additional evidence to support the

correctness of the new binary similarity measures.

5. We conduct an experimental study which presents external and internal evaluations of the proposed and existing similarity measures.

6. For all evaluation criteria, on average, our new similarity measures outperform the existing similarity measures.

2 Software modularization using AHC

Clustering algorithms can be broadly categorized into hierarchical and partitional. As stated in Section 1, AHC has been commonly used for software modularization. AHC considers each entity to be a singleton cluster and groups the two most similar clusters at every step. At the end, it makes one large cluster, which contains all the entities, as shown in Algorithm 1.

Algorithm 1 Agglomerative hierarchical clustering

Input: Feature matrix, $\boldsymbol{F}$

Output: Hierarchy of clusters (dendrogram)

1: Create a similarity matrix by calculating similarity using a **Similarity Measure** between each pair of entities.
2: **repeat**
3: Group the most similar (singleton) clusters into one cluster (using the maximum value of similarity in the similarity matrix).
4: Update the similarity matrix by recalculating similarity using a **Linkage Method** between a newly formed cluster and existing (singleton) clusters.
5: **until** the required number of clusters or a single large cluster is formed.

Partitional clustering produces flat clusters with no hierarchy, and it requires prior knowledge of the number of clusters. In the software domain, partitional clustering has also been used (Lakhotia, 1997; Kanellopoulos *et al.*, 2007; Shah *et al.*, 2013); however, there are some advantages of using AHC. For example, AHC does not require prior information about the number of clusters. Moreover, Wiggerts (1997) stated that the process of AHC is very similar to the approach of reverse engineering where the architecture of a software system is recovered in a bottom-up fashion. AHC provides different levels of abstraction and can be useful for end users to select the desired number of clusters when the modularization results are meaningful to them (Lutellier *et al.*, 2015). Since a maintainer may not have the knowledge of the number of clusters in advance, viewing the architecture at different abstraction levels facilitates understanding. Techniques have also been proposed to select an appropriate abstraction level; for example, Chong *et al.* (2013) proposed a dendrogram cutting approach for this purpose.

When AHC is used for software modularization, the first step that occurs is the selection of the entities to be clustered where each entity is described by different features. The steps are presented in detail in the following subsections.

2.1 Selection of entities and features

Selecting the entities and features associated with entities depends on the type of the software system and the desired architecture (e.g., layered/module architectures) to be recovered. For software modularization, researchers have used different types of entities, for example, methods (Saeed *et al.*, 2003), classes (Bauer and Trifu, 2004), and files (Anquetil and Lethbridge, 1999; Andritsos and Tzerpos, 2005). Researchers have also used different types of features to describe the entities such as global variables used by an entity (Muhammad *et al.*, 2012) and procedure calls (Andritsos and Tzerpos, 2005). Features are based on the relationships between entities, for example, containment and inheritance. Features may be in the binary or non-binary format. A binary feature represents the presence or absence of a relationship between two entities, while non-binary features are weighted features using different weighting schemes, for example, absolute or relative (Cui and Chae, 2011), to demonstrate the strength of the relationship between entities. Binary features are widely used in software modularization (Wiggerts, 1997; Mitchell and Mancoridis, 2006; Cui and Chae, 2011).

To apply AHC, a software system must be parsed to extract the selected entities and features associated with entities. This process results in a feature matrix of size $N \times P$, where N is the total number of entities and P the total number of features. Each entity in the feature matrix has a feature vector $\boldsymbol{f}_i = [f_1, f_2, \ldots, f_P]$. More generally, $\boldsymbol{F}$ presents a general feature matrix, which takes values from $\{0,1\}^P$; in other words, $\boldsymbol{F} = \{0,1\}^P$, where '1' means the presence of a feature and '0' otherwise. AHC takes $\boldsymbol{F}$ as the input, as shown in Algorithm 1.

Table 1 shows an example feature matrix $\boldsymbol{F}$ of a very small imaginary software system, which contains five entities (E_1–E_5) and seven binary features (f_1–f_7). In Table 1, for example, f_1 is present in entities E_1, E_2, and E_3 while absent in entities E_4 and E_5.

Table 1 An example feature matrix ($\boldsymbol{F}$)

Entity	f_1	f_2	f_3	f_4	f_5	f_6	f_7
E_1	1	1	0	0	0	0	0
E_2	1	1	0	0	0	0	0
E_3	1	0	1	1	0	0	0
E_4	0	0	1	1	1	0	0
E_5	0	0	0	0	0	1	0

2.2 Selection of similarity measure

The first step of the AHC process is to calculate the similarity between each pair of entities to obtain a similarity matrix by using a similarity measure, as shown in step 1 of Algorithm 1. Following are some well-known binary similarity measures for software modularization:

$$\text{Jaccard (JC)} = \frac{a}{a+b+c}, \tag{1}$$

$$\text{JaccardNM (JNM)} = \frac{a}{2(a+b+c)+d}, \tag{2}$$

$$\text{Russell\&Rao (RR)} = \frac{a}{a+b+c+d}. \tag{3}$$

All the existing binary similarity measures are expressed as functions of the following four quantities associated with the pair of entities $(E_i,\ E_j)$, $\forall E_i, E_j \in \boldsymbol{F}$ (Lesot *et al.*, 2009):

1. The number of features common to both entities, denoted by a;

2. The number of features present in E_i, but not in E_j, denoted by b;

3. The number of features present in E_j, but not in E_i, denoted by c;

4. The number of features absent in both entities, denoted by d.

Note that $a+b+c+d$ is a constant value and is equal to the total number of features P; $a+b=0$ occurs only when E_i has the feature vector $\boldsymbol{f}_i = [0,0,\ldots,0]$. Likewise, $a+c=0$ shows that E_j has feature vector $\boldsymbol{f}_j = [0,0,\ldots,0]$.

Definition 1 A binary similarity measure, SM, is a function whose domain is $\{0,1\}^P$, and whose range is $\mathbb{R}^+$; that is, SM: $\{0,1\}^P \to \mathbb{R}^+$ (Veal, 2011), with the following properties:

(P1) Positivity: $\text{SM}(E_i,\ E_j) \geq 0, \forall E_i, E_j \in \boldsymbol{F}$;

(P2) Symmetry: $\text{SM}(E_i,\ E_j) = \text{SM}(E_j,\ E_i)$, $\forall E_i, E_j \in \boldsymbol{F}$;

(P3) Maximality: $\text{SM}(E_i,\ E_i) \geq \text{SM}(E_i,\ E_j)$, $\forall E_i, E_j \in \boldsymbol{F}$.

To illustrate the calculation, for instance, of the JC measure as defined in Eq. (1), Table 2 gives the similarity matrix of the feature matrix, as shown in Table 1. The similarity between E_1 and E_2 is calculated using the quantities defined by a, b, c, and d, and in this case $a=2$, $b=0$, $c=0$, and $d=5$. Putting all these values in the JC similarity measure, we obtain similarity value '1' (shown in Table 2). Likewise, similarity values are calculated for each pair of entities and are presented in Table 2. Now, AHC will group the most similar entities in Table 2, according to step 2 in Algorithm 1. E_1 and E_2 have the highest similarity value, and thus AHC groups these entities in a single cluster (E_1E_2). A new cluster is therefore formed, and AHC will update the similarity values of E_1E_2 and all other (singleton) clusters, that is, E_3, E_4, and E_5. To update these similarity values, different linkage methods can be used, which will be described in the next subsection.

Table 2 Similarity matrix derived from the matrix in Table 1 by using the JC similarity measure

Entity	E_1	E_2	E_3	E_4	E_5
E_1					
E_2	1				
E_3	0.25	0.25			
E_4	0	0	0.5		
E_5	0	0	0	0.2	

2.3 Selection of the linkage method

When a new cluster is formed, the similarities between the new and existing clusters are updated using a linkage method, as shown in step 3 of Algorithm 1. There exist a number of methods, which update similarities differently. However, in this study, we discuss only those methods that are widely used for software modularization. They are listed below, where (E_iE_j) represents a new cluster and E_k represents an existing singleton cluster:

1. Complete linkage:

$\text{CL}(E_iE_j, E_k) = \min\{\text{sim}(E_i, E_k), \text{sim}(E_j, E_k)\}$;

2. Single linkage:

$\text{SL}(E_iE_j,\ E_k) = \max\{\text{sim}(E_i,\ E_k), \text{sim}(E_j,\ E_k)\}$;

3. Weighted average linkage:
$\mathrm{WL}(E_iE_j,\ E_k) = 0.5(\mathrm{sim}(E_i,\ E_k) + \mathrm{sim}(E_j,\ E_k))$.

In the illustrative example, we update similarity values between a new cluster (E_1E_2) and existing singleton clusters using the CL method. The updated similarity matrix is shown in Table 3. For example, the CL method returns the minimum similarity value between E_1 and E_3 (0.25) and E_2 and E_3 (0.25). These two returned values are the same (if there was a minimum, then that value would be selected). Therefore, AHC selects this similarity value as the new similarity between (E_1E_2) and E_3 (Table 3). Similarly, all similarity values are updated between (E_1E_2) and E_4, and between (E_1E_2) and E_5.

Table 3 Updated similarity matrix from the values in Table 2 using the complete linkage (CL) method

Entity	E_1E_2	E_3	E_4	E_5
E_1E_2				
E_3	0.25			
E_4	0	0.5		
E_5	0	0	0.2	

AHC repeats steps 2 and 3 until all entities are merged in one large cluster, or the desired number of clusters are obtained. At the end, AHC results in a hierarchy of clusters, also known as dendrogram, which is shown for the current example in Fig. 1. The obtained hierarchy is then evaluated to assess the quality of the automatically formed clusters, and the performance of similarity measures and methods.

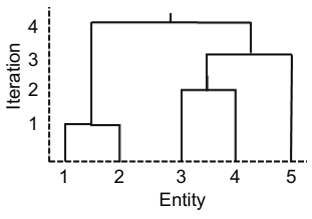

Fig. 1 Hierarchy created using the JC measure and CL method

2.4 Assessment of the results

Assessment of the clustering results is usually carried out using two approaches: external or internal assessment. The external assessment approach finds the association between automated results (decomposition) and the authoritative decomposition prepared by a human expert (e.g., original developer of the test software system). This approach is also known as authoritativeness. The automated decomposition should resemble the authoritative decomposition as much as possible (Wu *et al.*, 2005). To find the authoritativeness, different measures may be used, such as precision, recall (Sartipi and Kontogiannis, 2003), MoJo, and MoJoFM. Here, the widely used MoJoFM (Wen and Tzerpos, 2004) is discussed. MoJoFM is the updated version of MoJo (Tzerpos and Holt, 1999; Wen and Tzerpos, 2003), which calculates the move and join operations to convert the automated decomposition (M) into authoritative decomposition (N):

$$\mathrm{MoJoFM}(M,N) = \left(1 - \frac{\mathrm{mno}(M,N)}{\max\left\{\mathrm{mno}(\forall M,N)\right\}}\right) \times 100\%, \quad (4)$$

where $\mathrm{mno}(M,N)$ is the minimum number of 'move' and 'join' operations required to translate M into N and $\max\left\{\mathrm{mno}(\forall M,N)\right\}$ is the maximum of $\mathrm{mno}(\forall M,N)$. MoJoFM produces the percentage of the similarity between two decompositions. A higher percentage shows greater correspondence between the two decompositions and hence better results, while a lower percentage indicates that the decompositions are different.

Internal assessment is to evaluate the quality of the internal characteristics of the clusters in automated decomposition. There exist a number of measures to evaluate the cluster quality internally, for example, the number of arbitrary decisions (Wang *et al.*, 2010), the number of clusters (Wang *et al.*, 2010), size of clusters (extremity) (Glorie *et al.*, 2009), modularization quality (Praditwong, 2011), and coupling and cohesion (Cui and Chae, 2011). In this study, we intend to use the number of arbitrary decisions and number of clusters. Arbitrary decisions are taken by AHC when there exist more than one maximum similarity value in the similarity matrix during iteration. Thus, the decision of selecting the maximum value is arbitrary, since more than one pair of entities are equally similar. A large number of arbitrary decisions shows poor characteristic of the clusters, while a small number of arbitrary decisions means good characteristic, in terms of authoritativeness (Naseem *et al.*, 2013). The number of clusters is another internal

assessment criterion that is used to evaluate cluster quality. If an AHC algorithm produces a large number of clusters during the clustering process, it means that clusters are compact and have good quality (Maqbool and Babri, 2007). A small number of clusters during clustering means that they are large in size and less compact, and hence are considered having poor quality.

3 New similarity measures

As discussed in Section 1, we define new similarity measures that have the combined strengths of existing similarity measures JC, JNM, and RR defined in Eqs. (1)–(3), respectively. These three similarity measures have shown better results for software modularization as compared to other measures (Cui and Chae, 2011). To highlight the strengths of these existing measures, we first present a small example case study, and then define our new similarity measures.

3.1 An example case study

To illustrate the strengths of existing similarity measures, we take a small example of an imaginary feature matrix from Naseem *et al.* (2013). The example feature matrix is shown in Table 4, with eight entities (E_1–E_8) and 13 features (f_1–f_{13}). Using the feature matrix shown in Table 4, we illustrate the strengths of the JC and JNM similarity measures. We use the CL method in AHC, using JC with the CL and JNM with the CL on the feature matrix in Table 4.

3.1.1 JC with the CL clustering process

First, we illustrate the JC measure with the CL method. The first step of AHC is to create the similarity matrix using a similarity measure. After applying the JC measure to the feature matrix in Table 4, we obtain the similarity matrix (Table 5). In the first iteration of AHC, a maximum similarity value from the similarity matrix (Table 5) is selected to make a new cluster or update a cluster. So, AHC searches for a maximum similarity value in Table 5 but it finds maximum similarity value '1' twice (i.e., for (E_1E_2) and (E_3E_4)). AHC may select either, but we enforce AHC to select the last occurring value, that is, similarity value of (E_3E_4) (Table 6).

The CL method is used to update the similarity values between the new cluster (E_3E_4) and all existing singleton clusters, and the updated similarity matrix is shown in Table 6. In the second iteration, AHC searches again for the maximum value in the updated similarity matrix (Table 6). This time it makes (E_1E_2) as a new cluster and updates its similarity value with all other existing clusters (Table 7). In iterations 3 and 4, it makes clusters

Table 4 An example feature matrix

Entity	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}
E_1	0	0	0	0	0	1	1	1	1	1	1	1	1
E_2	0	0	0	0	0	1	1	1	1	1	1	1	1
E_3	1	1	0	0	0	0	0	0	0	0	0	0	0
E_4	1	1	0	0	0	0	0	0	0	0	0	0	0
E_5	1	1	1	1	1	0	0	0	0	0	0	0	0
E_6	1	1	1	1	0	0	0	0	0	0	0	0	0
E_7	0	0	1	1	1	1	1	0	0	1	0	1	0
E_8	0	0	1	1	1	1	0	1	1	0	1	0	0

Table 5 Similarity matrix of Table 4 using the Jaccard (JC) similarity measure

Entity	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
E_1								
E_2	**1**							
E_3	0	0						
E_4	0	0	**1**					
E_5	0	0	0.4	0.4				
E_6	0	0	0.5	0.5	0.8			
E_7	0.363	0.363	0	0	0.333	0.222		
E_8	0.363	0.363	0	0	0.333	0.222	0.4	

Table 6 Iteration 1: updated similarity matrix of Table 5 using the CL method

Entity	E_1	E_2	(E_3E_4)	E_5	E_6	E_7	E_8
E_1							
E_2	**1**						
(E_3E_4)	0	0					
E_5	0	0	0.4				
E_6	0	0	0.5	0.8			
E_7	0.363	0.363	0	0.333	0.222		
E_8	0.363	0.363	0	0.333	0.222	0.4	

Table 7 Iteration 2: updated similarity matrix of Table 6 using the CL method

Entity	(E_1E_2)	(E_3E_4)	E_5	E_6	E_7	E_8
(E_1E_2)						
(E_3E_4)	0					
E_5	0	0.4				
E_6	0	0.5	**0.8**			
E_7	0.363	0	0.333	0.222		
E_8	0.363	0	0.33	0.222	0.4	

of (E_5E_6) and (E_7E_8), as shown in Tables 8 and 9, respectively. From Table 8, it can be seen that there are two maximum values (0.4); hence, AHC may select either again. As stated earlier, AHC will select a value that occurs later; therefore, it makes a cluster (E_7E_8). In the remaining iterations, AHC makes clusters of $((E_3E_4)(E_5E_6))$, $((E_1E_2)(E_7E_8))$, and $(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$, as shown in Tables 10–12.

Table 8 Iteration 3: updated similarity matrix of Table 7 using the CL method

Entity	(E_1E_2)	(E_3E_4)	(E_5E_6)	E_7	E_8
(E_1E_2)					
(E_3E_4)	0				
(E_5E_6)	0	**0.4**			
E_7	0.363	0	0.222		
E_8	0.363	0	0.222	**0.4**	

Table 9 Iteration 4: updated similarity matrix of Table 8 using the CL method

Entity	(E_1E_2)	(E_3E_4)	(E_5E_6)	(E_7E_8)
(E_1E_2)				
(E_3E_4)	0			
(E_5E_6)	0	**0.4**		
(E_7E_8)	0.363	0	0.2	

Table 10 Iteration 5: updated similarity matrix of Table 9 using the CL method

Entity	(E_1E_2)	$((E_3E_4)(E_5E_6))$	(E_7E_8)
(E_1E_2)			
$((E_3E_4)(E_5E_6))$	0		
(E_7E_8)	**0.363**	0	

Table 11 Iteration 6: updated similarity matrix of Table 10 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	$((E_3E_4)(E_5E_6))$
$((E_1E_2)(E_7E_8))$		
$((E_3E_4)(E_5E_6))$	0	

Table 12 Iteration 7: updated similarity matrix of Table 11 using the CL method

Entity	$(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$
$(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$	

3.1.2 JNM with the CL clustering process

Now we apply the JNM measure on the feature matrix given in Table 4, and obtain a similarity matrix that can be seen in Table 13. The process for making clusters is the same as discussed in Section 3.1.1. As per the AHC, the first cluster formed is (E_1E_2), second is (E_5E_6), third is (E_7E_8), fourth is $((E_1E_2)(E_7E_8))$, fifth is (E_3E_4), sixth is $((E_3E_4)(E_5E_6))$, and the last is $(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$. The similarity matrices during iterations, from the first iteration to the seventh $(n-1)$ iteration, are given in Tables 14–20. In each iteration, the CL method is used to update the similarity between the newly formed and existing (singleton) clusters.

Table 13 Similarity matrix of the feature matrix in Table 4 using the JNM similarity measure

Entity	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
E_1								
E_2	**0.381**							
E_3	0	0						
E_4	0	0	0.133					
E_5	0	0	0.111	0.111				
E_6	0	0	0.118	0.118	0.222			
E_7	0.166	0.166	0	0	0.136	0.09		
E_8	0.166	0.166	0	0	0.136	0.09	0.173	

Table 14 Iteration 1: updated similarity matrix of Table 13 using the CL method

Entity	(E_1E_2)	E_3	E_4	E_5	E_6	E_7	E_8
(E_1E_2)							
E_3	0						
E_4	0	0.133					
E_5	0	0.111	0.111				
E_6	0	0.118	0.118	**0.222**			
E_7	0.166	0	0	0.136	0.09		
E_8	0.166	0	0	0.136	0.09	0.173	

Table 15 Iteration 2: updated similarity matrix of Table 14 using the CL method

Entity	(E_1E_2)	E_3	E_4	(E_5E_6)	E_7	E_8
(E_1E_2)						
E_3	0					
E_4	0	0.133				
(E_5E_6)	0	0.111	0.111			
E_7	0.166	0	0	0.09		
E_8	0.166	0	0	0.09	**0.173**	

Table 16 Iteration 3: updated similarity matrix of Table 15 using the CL method

Entity	(E_1E_2)	E_3	E_4	(E_5E_6)	(E_7E_8)
(E_1E_2)					
E_3	0				
E_4	0	0.133			
(E_5E_6)	0	0.111	0.111		
(E_7E_8)	**0.166**	0	0	0.09	

Table 17 Iteration 4: updated similarity matrix of Table 16 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	E_3	E_4	(E_5E_6)
$((E_1E_2)(E_7E_8))$				
E_3	0			
E_4	0	**0.133**		
(E_5E_6)	0	0.111	0.111	

Table 18 Iteration 5: updated similarity matrix of Table 17 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	(E_3E_4)	(E_5E_6)
$((E_1E_2)(E_7E_8))$			
(E_3E_4)	0		
(E_5E_6)	0	**0.111**	

Table 19 Iteration 6: updated similarity matrix of Table 18 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	$((E_3E_4)(E_5E_6))$
$((E_1E_2)(E_7E_8))$		
$((E_3E_4)(E_5E_6))$	**0**	

Table 20 Iteration 7: updated similarity matrix of Table 19 using the CL method

Entity	$(((E_1E_2)(E_7E_8))$ $((E_3E_4)(E_5E_6)))$
$(((E_1E_2)(E_7E_8))$ $((E_3E_4)(E_5E_6)))$	

3.1.3 Discussion on the results of the JC and JNM measures

In Sections 3.1.1 and 3.1.2, we can observe that the JC measure results in more clusters during clustering as compared to the JNM measure. Fig. 2 shows the maximum number of clusters achieved and the total number of arbitrary decisions made during the clustering process for both measures. It can be seen from Fig. 2 that the JNM measure produces a smaller number of arbitrary decisions as compared to the JC measure. JNM produces results as expected because the main intuition of introducing this measure is to reduce the number of arbitrary decisions (Naseem *et al.*, 2010). Hence, from these results we can easily conclude that the JC measure has the strength to create more clusters during the clustering process, while the JNM measure has the strength to reduce the number of arbitrary decisions.

It has been shown that if a clustering approach results in a large number of clusters during the clustering process, it increases the cohesiveness of the clusters and hence increases the authoritativeness of the results (Maqbool and Babri, 2007). Second, in our previous study, we showed that if a clustering approach reduces the number of arbitrary decisions, it would create more authoritative results (Naseem *et al.*, 2013). This analysis leads us to define the new measures that may have the characteristics of these existing similarity measures. It would be useful to integrate the existing similarity measures and come up with the new measures to increase the number of clusters and reduce the number of arbitrary decisions during the clustering process.

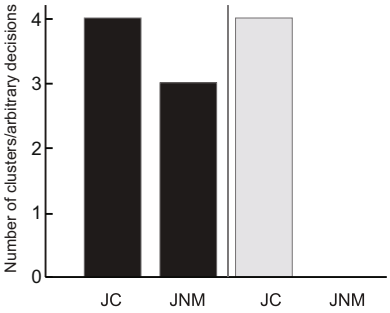

Fig. 2 The number of clusters (for black bar) and the number of arbitrary decisions (for grey bar) created by JC and JNM

3.2 New binary similarity measures

According to the aforementioned discussion, the JC measure has the strength of creating a large number of clusters during the clustering process, while the JNM measure has the strength of creating clusters with a small number of arbitrary decisions. We also consider the RR measure because for some case studies, it has produced better clustering results for software modularization (Naseem *et al.*, 2010). RR reduces the number of arbitrary decisions that JC creates when, for example, the values of a among the entities are different and $b + c = 0$. To combine the strengths of these existing similarity measures, the add operation is used to combine the existing similarity measures. We introduce four new similarity measures: JCJNM, JCRR, JNMRR, and ICJNMRR.

3.2.1 Addition of the JC and JNM measures: 'JCJNM' similarity measure

The strengths of similarity measures shown in Section 3.1 can be combined by adding the JC and JNM similarity measures to obtain the 'JCJNM' binary similarity measure. Our new measure 'JCJNM'

is defined as follows:

$$\text{JCJNM} = \text{JC} + \text{JNM} = \frac{a}{s} + \frac{a}{2s+d} = \frac{a(3s+d)}{s(2s+d)}, \tag{5}$$

where $s = a + b + c$.

3.2.2 An example case study and JCJNM measure

To demonstrate the strength of our new measure, we now apply the JCJNM similarity measure to the example feature matrix shown in Table 4. The corresponding similarity matrix using the JCJNM similarity measure is shown in Table 21. The CL method is used to update the similarity matrix during the clustering process. We can see from Table 21 that JCJNM prioritizes the similarity values between the pair of entities (E_1E_2) and (E_3E_4), as done by JNM in the similarity matrix given in Table 13. Hence, the decision to cluster the entities is no longer arbitrary. Entities E_1 and E_2 have a high value of similarity and are grouped first (Table 22). Then in the subsequent iterations (Tables 23–28), AHC makes clusters of (E_3E_4), (E_5E_6), (E_7E_8), etc. Note that in iteration 3 as given in Table 8, the JC measure creates arbitrary decisions while our new measure JCJNM does not, as shown in Table 24.

It is interesting to note that the JCJNM measure creates clusters as created by the JC measure (four clusters) and similar to the JNM measure, making no arbitrary decisions. It can be inferred that our new measure has the strength to create a large number of clusters while reducing the number of arbitrary decisions made by AHC during the clustering process. Therefore, JCJNM outperforms the existing similarity measures.

Proposition 1 Let the range of JCJNM be z (P represents the total number of features). Then we have

$$\text{JCJNM}(E_i,\ E_j) = \begin{cases} z = 1.5, & a = P, \\ z \in (0,\ 1.5), & 0 < a < P, \\ z = 0, & a = 0. \end{cases} \tag{6}$$

Proof As JCJNM is the combined function of four quantities *a, b, c*, and *d*, as shown in Eq. (5), substituting $s = a + b + c$ into Eq. (5), we have

$$\text{JCJNM}(E_i,\ E_j) = \frac{a[3(a+b+c)+d]}{(a+b+c)[2(a+b+c)+d]}. \tag{7}$$

Table 21 Similarity matrix of the feature matrix in Table 4 using the JCJNM similarity measure

Entity	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
E_1								
E_2	**1.381**							
E_3	0	0						
E_4	0	0	1.133					
E_5	0	0	0.511	0.511				
E_6	0	0	0.618	0.618	1.022			
E_7	0.526	0.526	0	0	0.466	0.31		
E_8	0.526	0.526	0	0	0.466	0.31	0.573	

Table 22 Iteration 1: updated similarity matrix of Table 21 using the CL method

Entity	(E_1E_2)	E_3	E_4	E_5	E_6	E_7	E_8
(E_1E_2)							
E_3	0						
E_4	0	**1.133**					
E_5	0	0.511	0.511				
E_6	0	0.618	0.618	1.022			
E_7	0.526	0	0	0.466	0.31		
E_8	0.526	0	0	0.466	0.31	0.573	

Table 23 Iteration 2: updated similarity matrix of Table 22 using the CL method

Entity	(E_1E_2)	(E_3E_4)	E_5	E_6	E_7	E_8
(E_1E_2)						
(E_3E_4)	0					
E_5	0	0.511				
E_6	0	0.618	**1.022**			
E_7	0.526	0	0.466	0.31		
E_8	0.526	0	0.466	0.31	0.573	

If all features are present in feature vectors of E_i and E_j, i.e., $b = c = d = 0$ and $a = P$, then the above equation reduces to

$$\text{JCJNM}(E_i,\ E_j) = \frac{a \cdot (3a)}{a \cdot (2a)} = 1.5. \tag{8}$$

Therefore, the maximum similarity value that JCJNM can produce is 1.5.

Now, if no common feature is present in feature vectors of E_i and E_j, i.e., $a = 0$ and $b + c + d \geq 0$, then Eq. (7) reduces to

$$\text{JCJNM}(E_i,\ E_j) = \frac{0[3(0+b+c)+d]}{(0+b+c)[2(0+b+c)+d]} = 0. \tag{9}$$

Thus, the minimum similarity value that JCJNM can produce is 0.

Lastly, if there exist some common and absent features in feature vectors of E_i and E_j, i.e., if $a = x$ and $b = c = d = y$, where $x,\ y > 0$, then Eq. (7)

Table 24 Iteration 3: updated similarity matrix of Table 23 using the CL method

Entity	(E_1E_2)	(E_3E_4)	(E_5E_6)	E_7	E_8
(E_1E_2)					
(E_3E_4)	0				
(E_5E_6)	0	0.511			
E_7	0.526	0	0.31		
E_8	0.526	0	0.31	**0.573**	

Table 25 Iteration 4: updated similarity matrix of Table 24 using the CL method

Entity	(E_1E_2)	(E_3E_4)	(E_5E_6)	(E_7E_8)
(E_1E_2)				
(E_3E_4)	0			
(E_5E_6)	0	0.511		
(E_7E_8)	**0.526**	0	0.31	

Table 26 Iteration 5: updated similarity matrix of Table 25 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	(E_3E_4)	(E_5E_6)
$((E_1E_2)(E_7E_8))$			
(E_3E_4)	0		
(E_5E_6)	0	**0.511**	

Table 27 Iteration 6: updated similarity matrix of Table 26 using the CL method

Entity	$((E_1E_2)(E_7E_8))$	$((E_3E_4)(E_5E_6))$
$((E_1E_2)(E_7E_8))$		
$((E_3E_4)(E_5E_6))$	0	

Table 28 Iteration 7: updated similarity matrix of Table 27 using the CL method

Entity	$(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$
$(((E_1E_2)(E_7E_8))((E_3E_4)(E_5E_6)))$	

reduces to

$$\mathrm{JCJNM}(E_i,\ E_j)=\frac{x[3(x+y+y)+y]}{(x+y+y)[2(x+y+y)+y]}. \tag{10}$$

The above equation simplifies to

$$\mathrm{JCJNM}(E_i,\ E_j)=\frac{3x^2+7xy}{2x^2+9xy+10y^2}. \tag{11}$$

Eq. (11) results in a value between 0 and 1.5, if $x,\ y>0,\ \forall E_i, E_j\in \boldsymbol{F}$.

Proposition 2 JCJNM satisfies Definition 1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^P$ and the range is $\mathbb{R}^+$.

Proof JCJNM is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated using only binary values in a feature vector of entities as defined in Section 2.2. Hence, the domain of the JCJNM measure is $\{0,1\}^P$. Meanwhile, JCJNM results in a real value, that is, $z\in\mathbb{R}^+$, as proved in Proposition 1.

Proposition 3 JCJNM fulfills the properties of positivity and symmetry.

Proof First, let us show the positivity. It has been shown in the proof of Proposition 1 that JCJNM creates a similarity value equal to or greater than 0; that is, $\mathrm{JCJNM}(E_i,\ E_j)\rightarrow\mathbb{R}^+,\ \forall E_i, E_j\in\boldsymbol{F}$.

Next, let us show the symmetry. JCJNM is the combined function of four quantities a, b, c, and d, and all these quantities are symmetric. So, it is obvious that

$$\mathrm{JCJNM}(E_i,\ E_j)=\mathrm{JCJNM}(E_j,\ E_i). \tag{12}$$

Proposition 4 JCJNM fulfills the maximality property of a similarity measure.

Proof Suppose $b+c=x$ and x is a positive number. Then Eq. (7) becomes

$$\mathrm{JCJNM}(E_i,\ E_j)=\frac{a[3(a+x)+d]}{(a+x)[2(a+x)+d]}. \tag{13}$$

The above equation simplifies to

$$\mathrm{JCJNM}(E_i,\ E_j)=\frac{a(3a+d+3x)}{(a+x)(2a+d+2x)}. \tag{14}$$

To calculate the similarity of an entity with itself, i.e., $\mathrm{JCJNM}(E_i,\ E_i)$, it is sure that $x=0$ and $a, d\geq 0$. Using these quantities, Eq. (14) reduces to

$$\mathrm{JCJNM}(E_i,\ E_i)=\frac{a(3a+d)}{a(2a+d)}. \tag{15}$$

Therefore, using Eqs. (14) and (15), $\forall E_i, E_j\in\boldsymbol{F}$, the following association will always be true for $a+d\geq 1$ and x, where $a+d+x=P$:

$$\frac{a(3a+d)}{a(2a+d)}\begin{cases}>\dfrac{a(3a+d+3x)}{(a+x)(2a+d+2x)}, & x>0,\\ \geq\dfrac{a(3a+d+3x)}{(a+x)(2a+d+2x)}, & x\geq 0.\end{cases}$$

Note that we have two associations, equality and maximality. Equality: for instance, let $x=0$, then Eq. (14) becomes equal to Eq. (15). Maximality: let $a=d=x=1$. So, $a+d+x=P=3$. Then the association between Eqs. (14) and (15) becomes

$$\frac{3+1}{2+1}>\frac{3+1+3}{(1+1)(2+1+2)}.$$

3.2.3 Addition of the JC and RR measures: 'JCRR' similarity measure

The second new similarity measure that we derive is 'JCRR'. This measure adds the RR similarity measure to the JC similarity measure. The derivation of JCRR is given as

$$\text{JCRR} = \text{JC} + \text{RR} = \frac{a}{s} + \frac{a}{s+d} = \frac{a(2s+d)}{s(s+d)}. \quad (16)$$

For proofs of the correctness of the JCRR binary similarity measure, please refer to Appendix A.

3.2.4 Combination of the JNM and RR measures: 'JNMRR' similarity measure

The third combination of the existing similarity measures is JNM and RR. We add these two similarity measures and the derived similarity measure is called 'JNMRR', defined as follows:

$$\begin{aligned}\text{JNMRR} &= \text{JNM} + \text{RR} \\ &= \frac{a}{2s+d} + \frac{a}{s+d} = \frac{a(3s+2d)}{(2s+d)(s+d)}.\end{aligned} \quad (17)$$

For proofs of the correctness of the JNMRR binary similarity measure, please refer to Appendix B.

3.2.5 Addition of the JC, JNM, and RR measures: 'JCJNMRR' similarity measure

Finally, we add all the three existing measures and come up with a new measure 'JCJNMRR'. This measure combines JC with JNM and RR:

$$\begin{aligned}\text{JCJNMRR} &= \text{JC} + \text{JNM} + \text{RR} \\ &= \frac{a}{s} + \frac{a}{2s+d} + \frac{a}{s+d} \\ &= \frac{a(5s^2+5sd+d^2)}{s(2s^2+3sd+d^2)}.\end{aligned} \quad (18)$$

For proofs of the correctness of the JCJNMRR binary similarity measure, please refer to Appendix C.

4 Experimental setup

AHC produces different results due to the biases of the different linkage methods and similarity measures (Cui and Chae, 2011). We have performed a number of experiments by employing well-known basic linkage methods using existing and our new similarity measures. In this section, we discuss the test systems used for experimental purposes and the clustering process setup including the selection of assessment criteria.

4.1 Test systems

We have used eight test systems that are developed using C, C++, and Java languages to conduct the experiments. The test systems have been used in previous research (Muhammad *et al.*, 2012; Siddique and Maqbool, 2012; Naseem *et al.*, 2013). All of these test systems vary in their source code sizes and application domains. Table 29 presents the details of the test software systems.

Table 29 Details of the test software systems

ID	Test system	Number of classes	Lines of code	Entity
1	DDA	90	82 877	Class
2	FES	47	10 402	Class
3	Mozilla	1202 files	400 000	File
4	PEDS	41	16 360	Class
5	PLC	69	51 768	Class
6	PLP	72	50 661	Class
7	SAVT	97	27 311	Class
8	Weka	331	100 000	Class

We used two open source and six proprietary test software systems. The open source test software systems are: (1) Mozilla, an open source web browser, which was developed in C and C++ programming languages. For experiments, we used Mozilla (ftp://ftp.mozilla.org/pub/mozilla.org/mozilla/releases/mozilla1.3/src/) version 1.3 released in March 2003. This test system is taken from Siddique and Maqbool (2012). (2) Weka, another open source software system developed in Java programming language, which is a well-known data mining software system used for data pre-processing, clustering, regression, classification, association rules, and visualization. We used Weka (http://perun.pmf.uns.ac.rs/radovanovic/dmsem/cd/install/Weka/doc/html/Weka%203.4.5.htm) version 3.4, taken from Siddique and Maqbool (2012), for experimental purposes.

The proprietary test software systems used for experiments were developed in Visual C++ programming language. They are: (1) DDA, software to design the document composition and layout; (2) FES, a fact extractor software system to

extract the facts of software systems; (3) PEDS, a power dispatch problem solver using conventional and evolutionary computing techniques; (4) PLC, a printer language converter software system used to convert the intermediate data structure to a well-known printer language; (5) PLP, a parser software system, which is used to parse a well-known printer language; (6) SAVT, a statistical and analysis visualization tool. These software systems are proprietary and are currently operational. We obtained the extracted feature matrix from Muhammad *et al.* (2012).

4.2 Entities and features

Mozilla's dataset is taken from Siddique and Maqbool (2012), who considered files as entities because a .c or .cpp file contains both functions with or without classes. Hence, in this study, files are considered entities for Mozilla, and file calling is used as a feature. The total number of file calling features is 258. For the Weka test system, which is purely developed in Java, we consider classes to be entities (Siddique and Maqbool, 2012). Since classes are considered to be the basic building blocks of object oriented languages, Siddique and Maqbool (2012) selected functions invoked, user-defined types, and global variables as features for classes.

For the proprietary software systems, Muhammad *et al.* (2012) considered classes to be entities. We select nine indirect features for these entities (Table 30), since indirect features give better results as compared to direct features (Muhammad *et al.*, 2012). We consider various types of test software systems that are developed in different programming languages, with different types of entities and features, because we want to see whether our newly proposed similarity measures are applicable to these different artifacts to make good clusters for software modularization.

4.3 Clustering strategies

In this study, we consider existing similarity measures and linkage methods that have produced better results for software modularization (Davey and Burd, 2000; Muhammad *et al.*, 2012). To conduct the experiments, we categorize clustering actors into different clustering stratagems as shown in Table 31. These stratagems are composed of three existing similarity measures and four new similarity measures, using three well-known basic linkage methods. The details of these stratagems are given in Table 31.

Table 30 Indirect features of the classes in proprietary test systems used for experiments

Feature type	Number of features					
	DDA	FES	PEDS	PLC	PLP	SAVT
Same inheritance hierarchy	98	166	70	26	64	986
Same class containment	58	56	12	58	144	1032
Same class in methods	476	384	76	162	672	1900
Same generic class	59	91	6	465	98	49
Same generic parameter	0	4	0	0	0	0
Same file	136	42	36	1812	826	264
Same folder	2456	0	0	0	0	0
Same macro access	0	0	12	0	2	0
Same global access	918	0	0	0	268	0
Total	4201	743	212	2523	2074	4231

4.4 Assessment criteria

To assess the output of the stratagems given in Section 4.3, we consider external as well as internal

Table 31 Clustering stratagems

Serial number	SM	Abbr.	Linkage method	Abbr.
1	JCJNM	JCJNM	Complete	CL
2	JCRR	JCRR	Complete	CL
3	JNMRR	JNMRR	Complete	CL
4	JCJNMRR	JCJNMRR	Complete	CL
5	JCJNM	JCJNM	Single	SL
6	JCRR	JCRR	Single	SL
7	JNMRR	JNMRR	Single	SL
8	JCJNMRR	JCJNMRR	Single	SL
9	JCJNM	JCJNM	Weighted average	WL
10	JCRR	JCRR	Weighted average	WL
11	JNMRR	JNMRR	Weighted average	WL
12	JCJNMRR	JCJNMRR	Weighted average	WL
13	Jaccard	JC	Complete	CL
14	JaccardNM	JNM	Complete	CL
15	Russell&Rao	RR	Complete	CL
16	Jaccard	JC	Single	SL
17	JaccardNM	JNM	Single	SL
18	Russell&Rao	RR	Single	SL
19	Jaccard	JC	Weighted average	WL
20	JaccardNM	JNM	Weighted average	WL
21	Russell&Rao	RR	Weighted average	WL

assessment. External assessment is the approach in which expert decomposition is required to evaluate the automated results, also known as authoritativeness. As AHC produces results at each iteration, the following question arises: which iteration's result should be evaluated? To answer the question, researchers used an external criterion to evaluate the results of each iteration, and then presented the maximum or average value of the criterion (Wen and Tzerpos, 2004; Maqbool and Babri, 2007; Muhammad *et al.*, 2012). We report the results by selecting the maximum MoJoFM value out of all values obtained from each iteration. To show the strengths and weaknesses of the proposed and existing measures, we also evaluate the experimental results using internal assessment criteria, that is, the number of arbitrary decisions (Naseem *et al.*, 2013) and the number of clusters produced during clustering (Maqbool and Babri, 2007).

4.5 Expert decomposition

Since we evaluate our results externally (using MoJoFM), it is important to have reliable expert decompositions, with which to compare our clustering results. For the proprietary software systems, expert decompositions are developed by personnel having design and development experience in the software industry. They have six to seven years of experience in developing software systems using C++. Some of the experts are the original developers of the software systems. We provided the source code and class listing to all the experts and asked them to develop a decomposition of the given system. The experts were not provided with any details about clustering algorithms, and what relationships between entities were used during clustering. A summary of relevant statistics of the experts is presented in Table 32.

Table 32 Personnel statistics

Personnel	System	Experience in years
Actual designer	DDA	7
Experienced in C++	FES	6
Actual designer	PEDS	7
Maintainer	PLC	7
Actual designer	PLP	6
Experienced in C++	SAVT	7

The process of preparing expert decomposition has been described in detail in Muhammad *et al.* (2012) and Naseem *et al.* (2013). These expert decompositions have been previously used in different studies (Muhammad *et al.*, 2012; Naseem *et al.*, 2013).

For Mozilla, we used the expert decomposition used in Siddique and Maqbool (2012). They have taken this decomposition from Xia and Tzerpos (2005). For Weka, we used the expert decomposition given in Patel *et al.* (2009). This expert decomposition was provided by the original designers of the Weka software system. The expert decompositions for these systems have been used for modularization experiments earlier (Andreopoulos *et al.*, 2005; Patel *et al.*, 2009; Siddique and Maqbool, 2012; Hussain *et al.*, 2015).

5 Assessment of new similarity measures

In this section, we present the experimental results using the number of arbitrary decisions, the number of clusters, and authoritativeness using the MoJoFM measure.

5.1 Number of arbitrary decisions

The overall results for all similarity measures are reported in Table 33. This table lists the average number of arbitrary decisions that are made by the AHC using different similarity measures in each iteration.

It can be seen from Table 33 that our proposed similarity measures have reduced the number of arbitrary decisions for each method. It is interesting that the JCJNM, JCRR, and JCJNMRR measures in most cases produce the same number of arbitrary decisions and also smaller ones than the existing JC and RR measures. JNMRR and JNM measures produce the same results except for in two cases, where the difference is minor. The same results may be due to the fact that both similarity measures (JNM and RR) have the ability to count all features, that is, a, b, c, and d. Therefore, adding RR may have no additional effect on the similarity values of JNM to reduce the number of arbitrary decisions.

Though our proposed measures produce a small number of arbitrary decisions, the existing JNM measure and our new JNMRR measure have the lowest number of arbitrary decisions. It is interesting to note that for the SL method, the JNMRR and JNM measures produce a smaller number of

Table 33 Experimental results using arbitrary decisions for all similarity measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average*	Average**
CL	JCJNM	2.06	10.28	253.19	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	**129.69**
	JCRR	2.06	10.28	253.20	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	
	JNMRR	**1.89**	**10.26**	253.17	**9.80**	37.63	9.25	**18.89**	**695.13**	**129.50**	
	JCJNMRR	2.07	10.28	253.19	**9.80**	**37.59**	**9.24**	20.47	695.42	129.76	
	JC	4.34	10.43	**245.27**	10.95	72.10	10.72	30.19	700.39	135.55	139.86
	JNM	**1.89**	**10.26**	253.17	**9.80**	37.63	9.25	**18.89**	**695.13**	**129.50**	
	RR	33.79	11.93	296.01	14.78	50.79	27.39	31.52	769.93	154.52	
SL	JCJNM	1.55	3.00	4.07	**1.13**	**35.90**	**1.85**	**10.91**	374.69	54.14	**54.03**
	JCRR	1.55	3.00	4.08	**1.13**	**35.90**	**1.85**	**10.91**	374.69	54.14	
	JNMRR	**1.22**	**2.98**	**3.03**	**1.13**	**35.90**	**1.85**	**10.91**	**372.77**	**53.72**	
	JCJNMRR	1.55	3.00	4.05	**1.13**	**35.90**	1.86	**10.91**	374.69	54.14	
	JC	3.93	3.20	8.68	2.25	70.47	3.30	20.98	384.05	62.11	60.90
	JNM	**1.22**	**2.98**	**3.03**	**1.13**	**35.90**	**1.85**	**10.91**	**372.77**	**53.72**	
	RR	8.38	4.70	17.49	8.20	53.16	20.59	22.32	400.20	66.88	
WL	JCJNM	1.15	2.98	2.36	0.98	(**35.81**)	1.80	(**10.68**)	371.36	53.39	(**53.37**)
	JCRR	1.15	2.98	2.37	0.98	(**35.81**)	1.80	(**10.68**)	371.36	53.39	
	JNMRR	(**0.97**)	(**2.91**)	(**2.23**)	(**0.90**)	35.85	(**1.79**)	(**10.68**)	(**371.21**)	(**53.32**)	
	JCJNMRR	1.15	2.98	2.36	0.98	(**35.81**)	(**1.79**)	(**10.68**)	371.36	53.39	
	JC	3.06	3.07	4.43	1.93	70.28	3.20	20.30	374.15	60.05	58.55
	JNM	0.99	(**2.91**)	(**2.23**)	(**0.90**)	35.85	(**1.79**)	10.69	(**371.21**)	(**53.32**)	
	RR	4.97	3.72	7.42	3.73	48.71	19.61	16.48	393.58	62.28	

* represents the average value for each similarity measure; ** represents the average value for a new or an existing measure. Bold case values represent the better values for a certain system/method, and the bold ones enclosed in parentheses indicate the best values

arbitrary decisions for all the test systems as compared to other methods.

It can be easily observed from the second last column of Table 33 that, as expected, our proposed similarity measures produced a smaller number of arbitrary decisions on average, as compared to existing similarity measures (see the last column of Table 33).

To ease the analysis, we select the average values from the second last column of Table 33, and summarize them in Table 34. Table 34 shows the average values of each similarity measure for linkage methods. It can be seen from the second last column of Table 34 that for all methods on average, JNMRR and JNM produced smaller numbers of arbitrary decisions. Meanwhile, the last column shows the average values for all methods of our new and existing similarity measures separately. As can be seen, our new similarity measures on average produced 79.03 arbitrary decisions during the clustering process, which is fewer than that produced by the existing similarity measures (86.44).

For further analysis, box-plots are used in Fig. 3 to illustrate the number of arbitrary decisions made during the clustering process by AHC using different similarity measures. A box-plot shows the variation

Table 34 Average numbers of arbitrary decisions for all similarity measures

Measure	Number of arbitrary decisions				
	CL	SL	WL	Average*	Average**
JCJNM	129.76	54.14	53.39	79.09	**79.03**
JCRR	129.76	54.14	53.39	79.09	
JNMRR	129.50	53.72	53.32	**78.85**	
JCJNMRR	129.76	54.14	53.39	79.09	
JC	135.55	62.11	60.05	85.90	86.44
JNM	129.50	53.72	53.32	**78.85**	
RR	154.52	66.88	62.28	94.56	

* represents the average value for each similarity measure; ** represents the average value for a new or an existing measure. Bold numbers represent the best values in the column

in the number of arbitrary decisions by indicating the quartiles and also highlights the outliers for each measure. For clarity, points are presented alongside a box-plot, representing the number of iterations in which many arbitrary decisions are made (i.e., the number of times that number of arbitrary decisions arises during clustering). Thus, the density of the points shows which arbitrary decision value is mostly observed during the clustering process. Due to limited space, only the results of CL and SL methods employed on DDA, FES, Mozilla, PEDS, and PLC test software systems are presented.

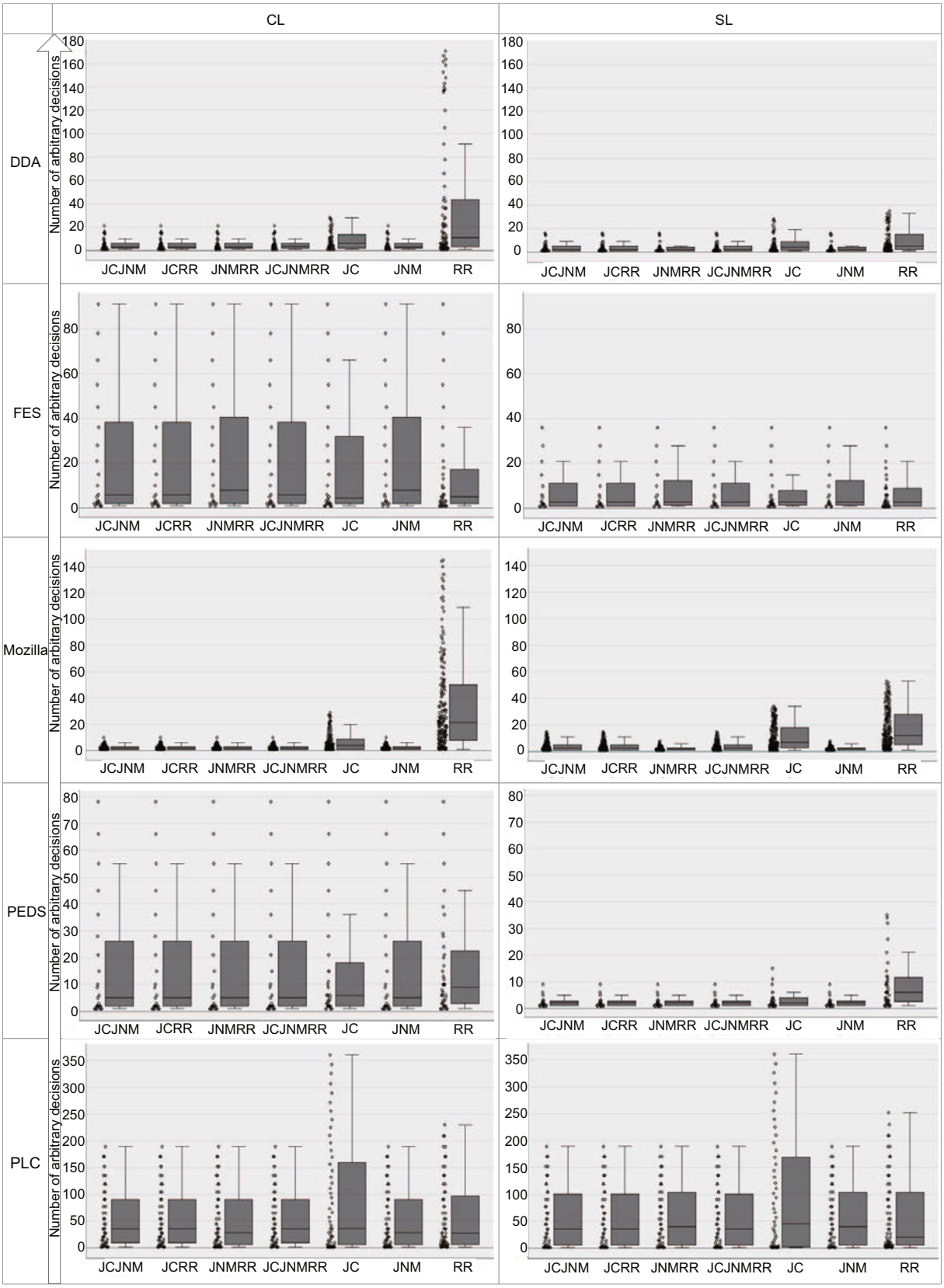

Fig. 3 The number of arbitrary decisions made during the clustering process by all binary similarity measures using the CL and SL methods for different test software systems

As shown in Fig. 3, JC and RR produce a larger number of arbitrary decisions as compared to JCJNM, JCRR, JNMRR, JCJNMRR, and JNM in general. This is apparent from the height of the box. Although in some cases the height of the box is lower for JC and RR (e.g., for FES using SL), there is a higher average in Table 33 for these measures. This indicates that arbitrary decisions are being made in a large number of iterations.

We also list some important statistics for our new and existing measures. Table 35 presents, for the arbitrary decisions, the minimum value (min), maximum value (max), mean, standard deviation (std), mode, and percentage of the count of occurrences of mode values. The min value for new measures is 78.85, which is equal to the min value for existing measures. However, the max value for new measures (79.09) is smaller than that for the existing measures. It indicates that new measures create the maximum value, which is very near to the minimum value created by both the existing and new measures. It can be seen that the std for our new similarity measures is 0.11, which is much smaller than that for existing similarity measures. This clearly indicates that our new similarity measures produce very compact results as compared to existing similarity measures. It can also be observed that 75% of the results produced by our proposed measures are the same, while the existing measures created varied results and hence found no mode.

Table 35 Statistics of arbitrary decisions

Measure type	Min	Max	Mean	Std	Mode	Percentage
New	78.85	79.09	79.03	0.11	79.09	75%
Existing	78.85	94.56	86.44	6.42	N/A	0%

5.2 Number of clusters

The number of clusters during the clustering process shows how compact the produced clusters are. A high number of clusters created during the clustering process indicates that the created clusters are highly compact, while a low number of clusters indicates that the created clusters are non-cohesive (Maqbool and Babri, 2007; Wang *et al.*, 2010). Hence, a high number of clusters indicates the usefulness of the AHC approach.

Table 36 shows the maximum number of non-singleton clusters created by AHC during all iterations. It can be seen from Table 36 that the number of clusters created by AHC using our new similarity measures is higher than that created by existing similarity measures. It can be seen that for all test software systems, JCJNM, JCRR, and JC measures created a large number of clusters using the CL, SL, and WL methods except for six cases, that is, CL applied on PLC, SL applied on DDA, Mozilla, and SAVT, and WL applied on Mozilla and Weka. For the SL method, JCJNM, JCRR, and JCJNMRR measures produced a large number of clusters for all test systems except one (SAVT). Again, note that JNMRR and JNM measures resulted in the same number of clusters for all test software systems and linkage methods. It can also be seen that for Mozilla and Weka software systems, our new measures substantially increased the number of clusters similar to the JC measure. Note that the new measures integrating the JC similarity measure have achieved an equally large number of clusters as the JC measure. This is because our new measures (JCJNM, JCRR, and JCJNMRR) integrate the JC measure, which results in a large number of clusters. JNMRR does not integrate the JC measure; therefore, results are the same as those of the JNM measure.

As can be seen from the second last column of Table 36, on average, for the CL method, the JCJNM, JCRR, JCJNMRR, and JC measures produced better results; that is, they created larger numbers of clusters. The JCJNM and JCRR measures with the SL method and the JCJNMRR measure with the WL method resulted in a large number of clusters. It can be seen from the last column of Table 36 that, for each linkage method our new similarity measures outperformed the existing ones.

To summarize the results, we listed the values given in the second last column of Table 36 in Table 37. Table 37 presents the numbers of clusters of similarity measures for linkage methods. This table clearly indicates how well a measure performs as compared to other contesting measures. This table also shows the average of each type, which indicates the average of all new measures and all existing measures separately. It can be seen that the new measures outperform the existing ones by creating a larger number of clusters.

For further analysis, a violin plot is shown in Fig. 4, which shows the numbers of clusters created during the clustering process by different similarity

Table 36 Experimental results using the number of clusters produced during clustering for all similarity measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average*	Average**
CL	JCJNM	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	(**25.75**)
	JCRR	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	
	JNMRR	21	9	64	**12**	(**11**)	11	16	35	22.38	
	JCJNMRR	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	
	JC	(**23**)	(**10**)	(**75**)	**12**	10	(**12**)	(**18**)	(**55**)	(**26.88**)	22.29
	JNM	21	9	64	**12**	(**11**)	11	16	35	22.38	
	RR	17	8	49	10	10	9	14	24	17.63	
SL	JCJNM	**17**	**8**	**49**	**8**	**6**	**8**	12	**31**	**17.38**	**14.72**
	JCRR	**17**	**8**	**49**	**8**	**6**	**8**	12	**31**	**17.38**	
	JNMRR	7	4	11	5	5	4	7	12	6.88	
	JCJNMRR	**17**	**8**	48	**8**	**6**	**8**	12	**31**	17.25	
	JC	16	**8**	48	**8**	**6**	**8**	**13**	**31**	17.25	10.08
	JNM	7	4	11	5	5	4	7	12	6.88	
	RR	7	4	8	5	4	4	5	12	6.13	
WL	JCJNM	**20**	**9**	68	**12**	**9**	**11**	**17**	**47**	24.13	**22.22**
	JCRR	**20**	**9**	68	**12**	**9**	**11**	**17**	**47**	24.13	
	JNMRR	15	7	45	8	7	10	13	26	16.38	
	JCJNMRR	**20**	**9**	**69**	**12**	**9**	**11**	**17**	**47**	**24.25**	
	JC	**20**	**9**	**69**	**12**	**9**	**11**	**17**	46	24.13	
	JNM	15	7	45	8	7	10	13	26	16.38	18.54
	RR	15	7	41	8	8	9	11	22	15.13	

* represents the average value for each similarity measure; ** represents the average value for a new or an existing measure. Bold case values represent the better values for a certain system/method, and the bold ones enclosed in parentheses indicate the best values

measures using CL and SL on various test software systems. The clustering process started by forming very small-sized clusters (size two). The increase in the height of the violin graph means that a large number of clusters are created during the clustering process. The width of the violin shows the value distribution over the observed number of clusters. It can be easily observed that our new similarity measures and JC (in most of the cases) resulted in a larger number of clusters. It can also be seen that JNM, RR, and JNMRR measures produced a relatively small number of clusters during the clustering process. This is especially true for the SL measure, where the number of clusters during clustering is much lower for these measures.

To support the claim that our proposed measures perform better, Table 38 shows the statistics of Table 37. All the statistical data (min, max, mean, std, mode, and percentage) are derived from the second last column of Table 37. The min value of new measures is 15.21, which is greater than the min value of existing measures (12.96). The max value of the new measures is slightly greater than that of the existing measures. It can be seen that the std value of our proposed measures (3.28) is smaller than that of the existing measures, which indicates that the results of our new measures are closer to the mean value and also closer to each other. This table also

Table 37 Average numbers of clusters produced during clustering for all similarity measures

Measure	Number of clusters			Average*	Average**
	CL	SL	WL		
JCJNM	**26.88**	**17.38**	24.13	**22.79**	**20.90**
JCRR	**26.88**	**17.38**	24.13	**22.79**	
JNMRR	22.38	6.88	16.38	15.21	
JCJNMRR	**26.88**	17.25	**24.25**	**22.79**	
JC	**26.88**	17.25	24.13	22.75	16.97
JCNM	22.38	6.88	16.38	15.21	
RR	17.63	6.13	15.13	12.96	

* represents the average value for each similarity measure; ** represents the average value for a new or an existing measure. Bold numbers represent the best values in the column

Table 38 Statistics of the number of clusters based on Table 37

Measure type	Min	Max	Mean	Std	Mode	Percentage
New	15.21	22.79	20.90	3.28	22.79	75%
Existing	12.96	22.75	16.97	4.19	N/A	0%

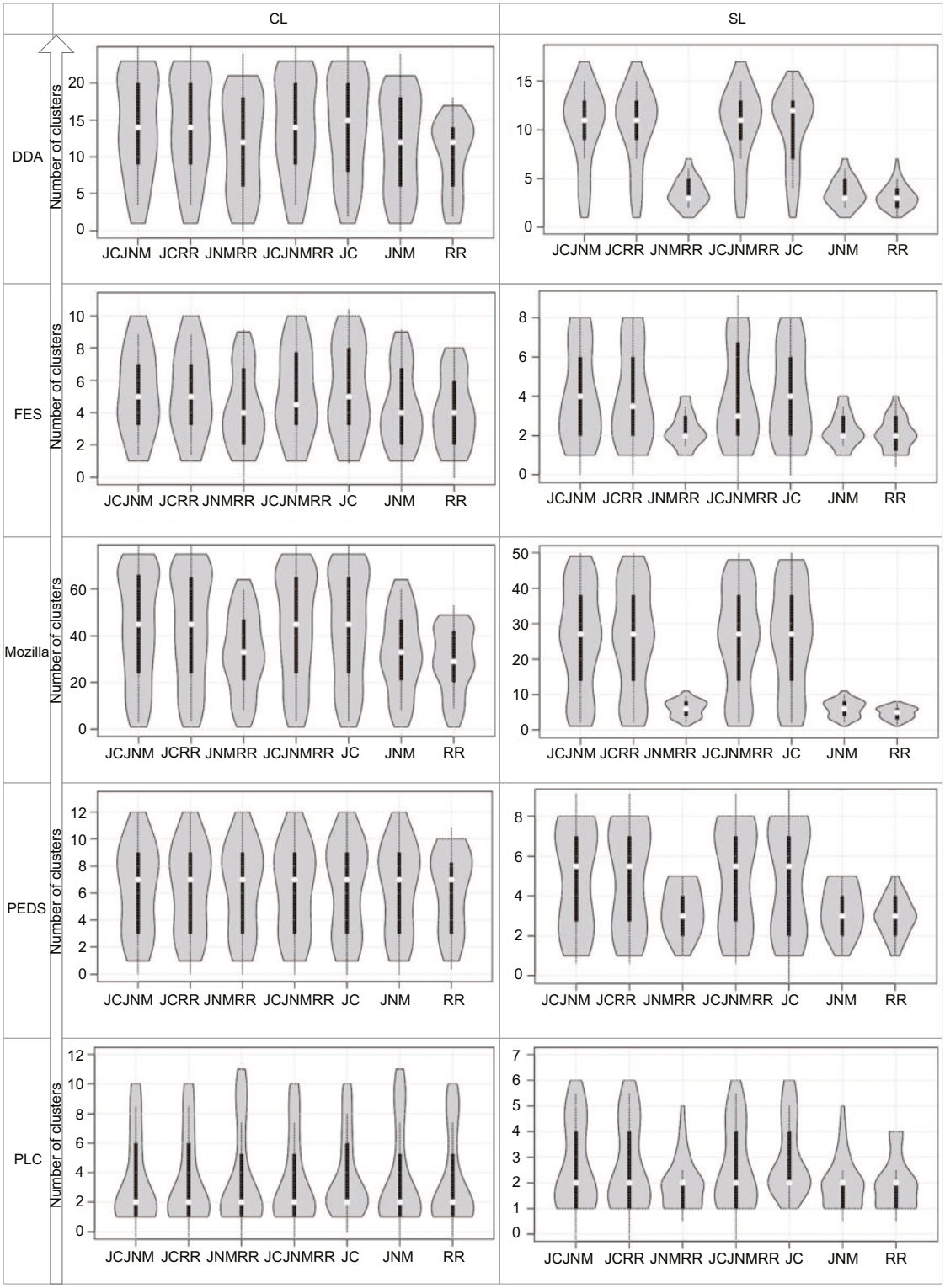

Fig. 4 The number of clusters created by all binary similarity measures during the clustering process using the CL and SL methods for different test software systems

shows the mode and percentage of the occurrence of the mode value. The mode of new measures is 22.79, which occurs three times in the second last column of Table 37. The percentage value 75% means that 75% values are the same for the JCJNM, JCRR, and JCJNMRR similarity measures.

5.3 Authoritativeness

The automated result is required to approximate the decomposition prepared by a human expert (an authority). For this purpose we use MoJoFM to compare the automated results with expert decompositions. The MoJoFM values for the series of experiments are given in Table 39. This table shows the maximum MoJoFM values selected during the iterations of the clustering process for all similarity measures and test software systems. The average values for each similarity measure are shown in the second last column of Table 39, while the last column presents the average for new measures and existing measures, based on the average values given in the second last column.

It can be seen from Table 39 that in most cases our new measures outperform the existing ones. As shown in Sections 5.1 and 5.2, our new similarity measures result in a smaller number of arbitrary decisions and a larger number of clusters. Thus, reducing the number of arbitrary decisions and increasing the number of clusters improve the authoritativeness of the automated results (Maqbool and Babri, 2007; Naseem *et al.*, 2013).

It can be easily analyzed that our new measures produce better results than the existing measures except for one test software system, that is, the PLC software system where the result of the JNM measure using the CL method is better. As can be seen from Table 39, for the DDA software system the JNMRR measure using the CL method results in the highest MoJoFM value. For the FES software system, the JCJNM, JCRR, and JCJNMRR measures using the CL method produce better results as compared to all other stratagems. For the Mozilla test system, the JNMRR similarity measure using the CL method gives the highest MoJoFM value. For the PEDS software system, our new and existing measures perform equally well. PLC is the only test system for which the existing JNM measure using the CL method results in the highest MoJoFM value. For the PLP software system, all the existing measures using the CL method give the lowest value, while for the SAVT test software system, the JNMRR measure using the

Table 39 MoJoFM results for all similarity measures

Method	Measure	DDA	FES	Mozilla	PEDS	PLC	PLP	SAVT	Weka	Average*	Average**
CL	JCJNM	56.25	**45.00**	63.89	**57.14**	61.54	(**65.67**)	**65.93**	**30.45**	55.73	(**55.94**)
	JCRR	56.25	**45.00**	63.89	**57.14**	61.54	**65.67**	**65.93**	**30.45**	55.73	
	JNMRR	(**60.00**)	**45.00**	**64.68**	**57.14**	61.54	**65.67**	(**67.03**)	30.13	(**56.40**)	
	JCJNMRR	57.50	**45.00**	63.89	**57.14**	61.54	**65.67**	**65.93**	**30.45**	55.89	
	JC	56.25	43.00	63.00	**57.14**	61.00	51.00	54.00	**30.45**	51.98	52.45
	JNM	56.25	43.00	64.00	**57.14**	(**65.00**)	60.00	54.00	30.13	53.69	
	RR	53.75	38.00	62.00	**57.14**	64.00	55.00	58.00	25.64	51.69	
SL	JCJNM	**53.75**	(**47.50**)	**46.03**	**57.14**	63.08	**59.70**	**58.24**	22.12	50.95	**47.97**
	JCRR	**53.75**	(**47.50**)	**46.03**	**57.14**	63.08	**59.70**	**58.24**	22.12	50.95	
	JNMRR	25.00	32.50	35.71	54.29	60.00	38.81	47.25	17.31	38.86	
	JCJNMRR	**53.75**	(**47.50**)	**46.03**	**57.14**	**64.62**	**59.70**	**58.24**	22.12	**51.14**	
	JC	**53.75**	35.00	46.00	54.29	55.00	28.00	32.00	**23.08**	40.89	36.16
	JNM	25.00	43.00	36.00	54.29	42.00	28.00	32.00	17.31	34.70	
	RR	23.75	33.00	36.00	51.43	42.00	28.00	32.00	16.99	32.90	
WL	JCJNM	**56.25**	42.50	61.11	**57.14**	**63.08**	**59.70**	(**67.03**)	(**32.05**)	**54.86**	**53.95**
	JCRR	**56.25**	42.50	61.11	**57.14**	**63.08**	**59.70**	(**67.03**)	(**32.05**)	**54.86**	
	JNMRR	50.00	40.00	56.35	54.29	**63.08**	58.21	63.74	24.68	51.29	
	JCJNMRR	**56.25**	42.50	**62.30**	**57.14**	**63.08**	**59.70**	(**67.03**)	30.45	54.81	
	JC	**56.25**	38.00	61.00	**57.14**	56.00	46.00	48.00	31.09	49.19	49.40
	JNM	50.00	35.00	61.00	54.29	55.00	55.00	53.00	24.68	48.50	
	RR	52.50	**45.00**	62.00	54.29	61.00	55.00	51.00	23.40	50.52	

*: the average value for each similarity measure; **: the average value for a new or an existing measure. Bold case values represent the better values for a certain system/method, and the bold ones enclosed in parentheses indicate the best values

CL method, and JCJNM, JCRR, and JCJNMRR measures using the WL method, result in the highest MoJoFM values (67.03%). Lastly, for the Weka software system, the JCJNM and JCRR measures using the WL method produce the highest MoJoFM values as compared to other measures using the CL and SL methods.

As can be seen from the second last column in Table 39, on average our new measures outperform the existing measures. The last column of Table 39 shows the average values of the new and existing similarity measures. As can be seen, our new measures outperform all existing measures using all linkage methods on all test software systems. Meanwhile, the CL method results in the highest MoJoFM value. Hence, we can infer that the CL method produces a better result as compared to the SL method, while the WL method falls in between these two.

To show the results more precisely, Table 40 lists the average values from the second last column of Table 39 for each similarity measure. As shown in Table 40, for the CL method, the JNMRR measure results in a better MoJoFM value as compared to other similarity measures. For the SL method, the JCJNMRR measure yields a better result, while for the WL method, JCJNM and JCRR produce better results. As can be seen from the second last column of Table 40, on average, JCJNMRR gives a better result as compared to all other similarity measures. Table 40 also provides good evidence to say that our new measures produce better results as compared to existing measures for all linkage methods.

Table 41 lists different statistics based on values in the second last column of Table 40. It can be seen that min, max, and mean values for the proposed new measures are higher than their counterparts of the existing measures. This indicates that in all cases, our new measures produce better results. These values indicate the superiority of our new measures over existing measures. Standard deviation (std) values for both types of measures indicate that each type has one value that is out of the range. Meanwhile, JCJNM and JCRR measures result in exactly the same values, which can be seen in Tables 40 and 41 (column mode). The percentage value 50% means that 50% values are the same for JCJNM and JCRR similarity measures.

Table 40 Average MoJoFM results for all similarity measures

Measure	MoJoFM			Average*	Average**
	CL	SL	WL		
JCJNM	55.73	50.95	**54.86**	53.85	**52.62**
JCRR	55.73	50.95	**54.86**	53.85	
JNMRR	**56.40**	38.86	51.29	48.85	
JCJNMRR	55.89	**51.14**	54.81	**53.94**	
JC	51.98	40.89	49.19	47.35	46.01
JNM	53.69	34.70	48.50	45.63	
RR	51.69	32.90	50.52	45.04	

* represents the average value for each similarity measure; ** represents the average value for a new or an existing measure. Bold numbers represent the best values in the column

Table 41 Statistics for new and existing measures based on Table 40

Measure type	Min	Max	Mean	Std	Mode	Percentage
New	48.85	53.94	52.62	2.18	53.85	50%
Existing	45.04	47.35	46.01	0.98	N/A	0%

5.3.1 Significance of authoritativeness

To show the significance of authoritativeness results, T-test has been conducted to show if there is a significant difference between the MoJoFM values of new measures and the existing measures. In this case, null hypothesis H_o is as follows: There is no difference between the MoJoFM values of new measures and the existing measures.

T-test (with df=7) has been conducted between the new and existing measures using the CL and SL methods. The T-test results are given in Table 42. This table clearly indicates that if $t > 2.365$, new measures have significantly performed better at the

Table 42 *T*-test values between the MoJoFM results of the new and existing measures using the CL and SL methods

New measure	Existing measure	T-test value	
		CL	SL
JCJNM	JC	1.90	2.41
	JNM	1.30	4.10
	RR	2.79	5.10
JCRR	JC	1.90	2.41
	JNM	1.30	4.10
	RR	2.79	5.10
JNMRR	JC	2.24	−0.45
	JNM	1.64	1.31
	RR	3.19	2.39
JCJNMRR	JC	2.01	2.46
	JNM	1.42	4.11
	RR	2.93	5.12

95% confidence level; at this level of confidence, all new measures are significantly better than RR using the CL method, and JCJNM, JCRR, and JCJNMRR are better than all existing measures using SL. Moreover, JNMRR is significantly better than RR using the SL method. At the 90% confidence level ($t > 1.895$), all new measures are significantly better than JC using the CL method. At the 80% confidence level ($t > 1.415$), JNMRR and JCJNMRR are significantly better than JNM using the CL method. Finally, JCJNM and JCRR are significantly better than JNM using the CL method at the 70% confidence level ($t > 1.12$). Thus, the new measures have produced significantly better results at different confidence levels; hence, the null hypothesis H_o is rejected.

5.4 Summary of results

Considering the overall performance of the similarity measures, our new similarity measures (JCJNM, JCRR, JNMRR, and JCJNMRR) outperform the existing similarity measures (JC, JNM, and RR) using the CL, SL, and WL methods with respect to authoritativeness, the number of clusters, and the number of arbitrary decisions. It can be noticed that for most of the cases the JCJNM, JCRR, and JCJNMRR measures yield better results. However, on average JNMRR using the CL method results in the highest authoritativeness value (Table 39). The JCJNM and JCRR measures used with the WL method achieve the second position, and the JCJNMRR measure using the SL method is on the third position. On average, the JCJNMRR measure gives better authoritativeness for all methods as compared to all other measures, which can be seen in Table 40.

Experimental results in terms of the number of clusters reveal that for all test software systems, the JCJNM, JCRR, JCJNMRR, and JC measures using the CL method yield better results, as can be seen in Table 36. However, for all methods, the JCJNM measure performs better as shown in Table 37.

As can be seen from Table 33, the JNMRR measure using the WL method creates the lowest arbitrary decisions for all test software systems, as compared to other stratagems. For all methods, the JNMRR and JNM measures produce a smaller number of arbitrary decisions, as shown in Table 34.

As compared to the SL and WL methods, the CL method yields better results for authoritativeness for all the test software systems, as shown in Table 39. The WL method results in a smaller number of arbitrary decisions, as presented in Table 33. The CL method creates a larger number of clusters during clustering as compared to the WL and SL methods, which can be seen in Table 36.

5.5 Threats to validity

The selection of test systems may pose a threat to a study, since the test system characteristics may influence the results. To reduce this threat, we selected open or closed source software systems of different sizes (numbers of source lines) having different application domains. We used eight test systems developed in well-known programming languages, that is, C, C++, and Java. However, more experiments may be conducted on large-size software systems developed in other programming languages.

To evaluate the results, we used MoJoFM as the external assessment criterion, and used the number of clusters and the number of arbitrary decisions as the internal assessment criteria. As MoJoFM requires an expert decomposition prepared by a human, a threat may exist during the preparation of expert decomposition due to human biases. However, we have tried to reduce this threat by selecting experienced software personnel as experts, who were also involved in the development of the proprietary software systems. For Mozilla, expert decomposition was prepared by Godfrey and Lee (2000) and then updated for Mozilla 1.3 by Xia and Tzerpos (2005); for Weka, we used the decomposition prepared by the actual designers of the systems (Patel *et al.*, 2009). Moreover, we selected software systems that have been used in previous studies, with their expert decompositions (Andreopoulos *et al.*, 2005; Patel *et al.*, 2009; Muhammad *et al.*, 2012; Siddique and Maqbool, 2012; Hussain *et al.*, 2015). Since our main goal was to integrate the strengths of existing measures, our choices of internal assessment criteria are based on that goal. We are aware that there exist a number of other internal assessment criteria that can also be used to investigate the results, for example, coupling and cohesion.

6 Conclusions

We have presented the improved binary similarity measures, namely, JCJNM, JCRR, JNMRR, and

JCJNMRR, for clustering software systems for modularization. These measures integrate the strengths of the following existing binary similarity measures: Jaccard (JC), JaccardNM (JNM), and Russell&Rao (RR). An example of the existing and new similarity measures has been presented to show how our new measures integrate strengths of the existing similarity measures. We have presented empirical results to compare the results of the existing and new similarity measures and showed the strengths of our new measures. Experimental results have been obtained using eight real-world, different test software systems which are developed in different programming languages and belong to different domain applications.

The empirical results indicate that our new binary similarity measures can produce better authoritative results than the existing similarity measures. Our new measures can reduce the number of arbitrary decisions and increase the number of clusters during the clustering process. The proposed similarity measures can be applied to test software systems developed in any programming language, since they depend only on the binary feature vector representation of data.

It is possible to extend our integration approach to form new similarity and distance measures using existing similarity and distance measures for other clustering application domains.

References

Andreopoulos, B., An, A.J., Tzerpos, V., *et al.*, 2005. Multiple layer clustering of large software systems. Proc. 12th Working Conf. on Reverse Engineering, p.79-88. https://doi.org/10.1109/wcre.2005.24

Andritsos, P., Tzerpos, V., 2005. Information-theoretic software clustering. *IEEE Trans. Softw. Eng.*, **31**(2): 150-165. https://doi.org/10.1109/tse.2005.25

Anquetil, N., Lethbridge, T.C., 1999. Experiments with clustering as a software remodularization method. Proc. 6th Working Conf. on Reverse Engineering, p.235-255. https://doi.org/10.1109/wcre.1999.806964

Bauer, M., Trifu, M., 2004. Architecture-aware adaptive clustering of OO systems. Proc. 8th European Conf. on Software Maintenance and Reengineering, p.3-14. https://doi.org/10.1109/csmr.2004.1281401

Bittencourt, R.A., Guerrero, D.D.S., 2009. Comparison of graph clustering algorithms for recovering software architecture module views. Proc. 13th European Conf. on Software Maintenance and Reengineering, p.251-254. https://doi.org/10.1109/csmr.2009.28

Cheetham, A.H., Hazel, J.E., 1969. Binary (presence-absence) similarity coefficents. *J. Paleontol.*, **43**(5): 1130-1136.

Chong, C.Y., Lee, S.P., Ling, T.C., 2013. Efficient software clustering technique using an adaptive and preventive dendrogram cutting approach. *Inform. Softw. Technol.*, **55**(11):1994-2012. https://doi.org/10.1016/j.infsof.2013.07.002

Cui, J.F., Chae, H.S., 2011. Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems. *Inform. Softw. Technol.*, **53**(6): 601-614. https://doi.org/10.1016/j.infsof.2011.01.006

Davey, J., Burd, E., 2000. Evaluating the suitability of data clustering for software remodularisation. Proc. 7th Working Conf. on Reverse Engineering, p.268-276. https://doi.org/10.1109/wcre.2000.891478

Dugerdil, P., Jossi, S., 2008. Reverse-architecting legacy software based on roles: an industrial experiment. *Commun. Comput. Inform. Sci.*, **22**:114-127. https://doi.org/10.1007/978-3-540-88655-6_9

Glorie, M., Zaidman, A., van Deursen, A., *et al.*, 2009. Splitting a large software repository for easing future software evolution—an industrial experience report. *J. Softw. Mainten. Evol. Res. Pract.*, **21**(2):113-141. https://doi.org/10.1002/smr.401

Godfrey, M.W., Lee, E.H., 2000. Secrets from the monster: extracting Mozilla's software architecture. Proc. Int. Symp. on Constructing Software Engineering Tools, p.1-10.

Hall, M., Walkinshaw, N., McMinn, P., 2012. Supervised software modularisation. Proc. 28th IEEE Int. Conf. on Software Maintenance, p.472-481. https://doi.org/10.1109/icsm.2012.6405309

Hussain, I., Khanum, A., Abbasi, A.Q., *et al.*, 2015. A novel approach for software architecture recovery using particle swarm optimization. *Int. Arab. J. Inform. Technol.*, **12**(1):1-10.

Jackson, D.A., Somers, K.M., Harvey, H.H., 1989. Similarity coefficients: measures of co-occurrence and association or simply measures of occurrence. *Am. Nat.*, **133**(3):436-453. https://doi.org/10.1086/284927

Jahnke, J.H., 2004. Reverse engineering software architecture using rough clusters. Proc. IEEE Annual Meeting of the Fuzzy Information, p.4-9. https://doi.org/10.1109/nafips.2004.1336239

Kanellopoulos, Y., Antonellis, P., Tjortjis, C., *et al.*, 2007. K-attractors: a clustering algorithm for software measurement data analysis. Proc. 19th IEEE Int. Conf. on Tools with Artificial Intelligence, p.358-365. https://doi.org/10.1109/ictai.2007.31

Lakhotia, A., 1997. A unified framework for expressing software subsystem classification techniques. *J. Syst. Softw.*, **36**(3):211-231. https://doi.org/10.1016/0164-1212(95)00098-4

Lesot, M.J., Rifqi, M., Benhadda, H., 2009. Similarity measures for binary and numerical data: a survey. *Int. J. Knowl. Eng. Soft Data Parad.*, **1**(1):63. https://doi.org/10.1504/ijkesdp.2009.021985

Lung, C.H., Zaman, M., Nandi, A., 2004. Applications of clustering techniques to software partitioning, recovery and restructuring. *J. Syst. Softw.*, **73**(2):227-244. https://doi.org/10.1016/s0164-1212(03)00234-6

Lutellier, T., Chollak, D., Garcia, J., *et al.*, 2015. Comparing software architecture recovery techniques using accurate dependencies. Proc. 37th IEEE Int. Conf. on Software Engineering, p.69-78. https://doi.org/10.1109/icse.2015.136

Maqbool, O., Babri, H., 2004. The weighted combined algorithm: a linkage algorithm for software clustering. Proc. 8th European Conf. on Software Maintenance and Reengineering, p.15-24. https://doi.org/10.1109/csmr.2004.1281402

Maqbool, O., Babri, H., 2007. Hierarchical clustering for software architecture recovery. *IEEE Trans. Softw. Eng.*, **33**(11):759-780. https://doi.org/10.1109/tse.2007.70732

Mitchell, B.S., 2006. Clustering Software Systems to Identify Subsystem Structures. Technical Report, Department of Mathematics and Computer Science, Drexel University, USA.

Mitchell, B.S., Mancoridis, S., 2006. On the automatic modularization of software systems using the Bunch tool. *IEEE Trans. Softw. Eng.*, **32**(3):193-208. https://doi.org/10.1109/tse.2006.31

Muhammad, S., Maqbool, O., Abbasi, A.Q., 2012. Evaluating relationship categories for clustering object-oriented software systems. *IET Softw.*, **6**(3):260-274. https://doi.org/10.1049/iet-sen.2011.0061

Naseem, R., Maqbool, O., Muhammad, S., 2010. An improved similarity measure for binary features in software clustering. Proc. 2nd Int. Conf. on Computational Intelligence, Modelling and Simulation, p.111-116. https://doi.org/10.1109/cimsim.2010.34

Naseem, R., Maqbool, O., Muhammad, S., 2011. Improved similarity measures for software clustering. Proc. 15th European Conf. on Software Maintenance and Reengineering, p.45-54. https://doi.org/10.1109/csmr.2011.9

Naseem, R., Maqbool, O., Muhammad, S., 2013. Cooperative clustering for software modularization. *J. Syst. Softw.*, **86**(8):2045-2062. https://doi.org/10.1016/j.jss.2013.03.080

Patel, C., Hamou-Lhadj, A., Rilling, J., 2009. Software clustering using dynamic analysis and static dependencies. Proc. 13th European Conf. on Software Maintenance and Reengineering, p.27-36. https://doi.org/10.1109/csmr.2009.62

Praditwong, K., 2011. Solving software module clustering problem by evolutionary algorithms. Proc. 8th Int. Joint Conf. on Computer Science and Software Engineering, p.154-159. https://doi.org/10.1109/jcsse.2011.5930112

Praditwong, K., Harman, M., Yao, X., 2011. Software module clustering as a multi-objective search problem. *IEEE Trans. Softw. Eng.*, **37**(2):264-282. https://doi.org/10.1109/tse.2010.26

Saeed, M., Maqbool, O., Babri, H., *et al.*, 2003. Software clustering techniques and the use of combined algorithm. Proc. 7th European Conf. on Software Maintenance and Reengineering, p.301-306. https://doi.org/10.1109/csmr.2003.1192438

Sartipi, K., Kontogiannis, K., 2003. On modeling software architecture recovery as graph matching. Proc. Int. Conf. on Software Maintenance, p.224-234. https://doi.org/10.1109/icsm.2003.1235425

Seung-Seok, C., Cha, S.H., Tappert, C.C., 2010. A survey of binary similarity and distance measures. *J. Syst. Cybern. Inform.*, **8**(1):43-48.

Shah, Z., Naseem, R., Orgun, M., *et al.*, 2013. Software clustering using automated feature subset selection. Proc. Int. Conf. on Advanced Data Mining and Applications, p.47-58. https://doi.org/10.1007/978-3-642-53917-6_5

Shtern, M., Tzerpos, V., 2010. On the comparability of software clustering algorithms. Proc. IEEE 18th Int. Conf. on Program Comprehension, p.64-67. https://doi.org/10.1109/icpc.2010.25

Shtern, M., Tzerpos, V., 2012. Clustering methodologies for software engineering. *Adv. Softw. Eng.*, **2012**: 792024.1-792024.18. https://doi.org/10.1155/2012/792024

Shtern, M., Tzerpos, V., 2014. Methods for selecting and improving software clustering algorithms. *Softw. Pract. Exp.*, **44**(1):33-46. https://doi.org/10.1002/spe.2147

Siddique, F., Maqbool, O., 2012. Enhancing comprehensibility of software clustering results. *IET Softw.*, **6**(4):283. https://doi.org/10.1049/iet-sen.2012.0027

Synytskyy, N., Holt, R.C., Davis, I., 2005. Browsing software architectures with LSEdit. Proc. 13th Int. Workshop on Program Comprehension, p.176-178. https://doi.org/10.1109/wpc.2005.11

Tonella, P., 2001. Concept analysis for module restructuring. *IEEE Trans. Softw. Eng.*, **27**(4):351-363. https://doi.org/10.1109/32.917524

Tzerpos, V., Holt, R.C., 1999. MoJo: a distance metric for software clusterings. Proc. 6th Working Conf. on Reverse Engineering, p.187-193. https://doi.org/10.1109/wcre.1999.806959

Tzerpos, V., Holt, R.C., 2000. On the stability of software clustering algorithms. Proc. 8th Int. Workshop on Program Comprehension, p.211-218. https://doi.org/10.1109/wpc.2000.852495

Vasconcelos, A., Werner, C., 2007. Architecture recovery and evaluation aiming at program understanding and reuse. Proc. Int. Conf. on the Quality of Software Architectures, p.72-89. https://doi.org/10.1007/978-3-540-77619-2_5

Veal, B.W.G., 2011. Binary Similarity Measures and Their Applications in Machine Learning. PhD Thesis, London School of Economics, London, UK.

Wang, Y., Liu, P., Guo, H., *et al.*, 2010. Improved hierarchical clustering algorithm for software architecture recovery. Proc. Int. Conf. on Intelligent Computing and Cognitive Informatics, p.247-250. https://doi.org/10.1109/icicci.2010.45

Wen, Z., Tzerpos, V., 2003. An optimal algorithm for MoJo distance. Proc. 11th IEEE Int. Workshop on Program Comprehension, p.227-235. https://doi.org/10.1109/wpc.2003.1199206

Wen, Z., Tzerpos, V., 2004. An effectiveness measure for software clustering algorithms. Proc. 12th IEEE Int. Workshop on Program Comprehension, p.194-203. https://doi.org/10.1109/wpc.2004.1311061

Wiggerts, T.A., 1997. Using clustering algorithms in legacy systems remodularization. Proc. 4th Working Conf. on Reverse Engineering, p.33-43. https://doi.org/10.1109/wcre.1997.624574

Wu, J., Hassan, A.E., Holt, R.C., 2005. Comparison of clustering algorithms in the context of software evolution. Proc. 21st IEEE Int. Conf. on Software Maintenance, p.525-535. https://doi.org/10.1109/icsm.2005.31

Xanthos, S., Goodwin, N., 2006. Clustering object-oriented software systems using spectral graph partitioning. *Urbana*, **51**(1):1-5.

Xia, C., Tzerpos, V., 2005. Software clustering based on dynamic dependencies. Proc. 9th European Conf. on Software Maintenance and Reengineering, p.124-133. https://doi.org/10.1109/csmr.2005.49

Appendix A: Propositions for JCRR

Proposition A1 Let the range of JCRR be z. Then we have

$$\mathrm{JCRR}(E_i,\ E_j)=\begin{cases}z=2, & a=P,\\ z\in(0,\ 2), & 0<a<P,\\ z=0, & a=0.\end{cases} \tag{A1}$$

Proof As JCRR is the combined function of four quantities a, b, c, and d, as shown in Eq. (16), substituting $s=a+b+c$ into Eq. (16), we have

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{a[2(a+b+c)+d]}{(a+b+c)[(a+b+c)+d]}. \tag{A2}$$

If all features are present in feature vectors of E_i and E_j, i.e., $b=c=d=0$ and $a=P$, then the above equation reduces to

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{a\cdot(2a)}{a\cdot a}=2. \tag{A3}$$

Therefore, the maximum similarity value that JCRR can create is 2.

Now, if no common feature is present in feature vectors of E_i and E_j, i.e., $a=0$ and $b+c+d\geq 0$, then Eq. (A2) reduces to

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{0\cdot[2(0+b+c)+d]}{(0+b+c)[(0+b+c)+d]}=0. \tag{A4}$$

Thus, the minimum similarity value that JCRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of E_i and E_j, i.e., if $a=x$ and $b=c=d=y$, where $x, y>0$, then Eq. (A2) reduces to

$$\mathrm{JCRR}(E_i,E_j)=\frac{x[2(x+y+y)+y]}{(x+y+y)[(x+y+y)+y]}. \tag{A5}$$

The above equation simplifies to

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{2x^2+5xy}{x^2+5xy+6y^2}. \tag{A6}$$

Eq. (A6) results in a value between 0 and 2, if $x, y>0, \forall E_i, E_j\in \boldsymbol{F}$.

Proposition A2 JCRR satisfies Definition 1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^P$ and the range is $\mathbb{R}^+$.

Proof JCRR is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated using only binary values in a feature vector of entities as defined in Section 2.2. Hence, the domain of JCRR measure is $\{0,1\}^P$. Meanwhile, JCRR results in a real value, that is, $z\in\mathbb{R}^+$, as proved in Proposition A1:

$$\mathrm{JCRR}(E_i,\ E_j)=\begin{cases}z>0, & a\geq 1,\\ z=0, & \text{otherwise.}\end{cases} \tag{A7}$$

Proposition A3 JCRR fulfills the properties of positivity and symmetry of a similarity measure.

Proof First, let us show the positivity. It has been shown in the proof of Proposition A1 that JCRR creates a similarity value equal to or greater than 0; that is, $\mathrm{JCRR}(E_i, E_j)\rightarrow\mathbb{R}^+, \forall E_i, E_j\in\boldsymbol{F}$.

Next, let us show the symmetry. It is obvious that

$$\mathrm{JCRR}(E_i,E_j)=\mathrm{JCRR}(E_j,E_i). \tag{A8}$$

Proposition A4 JCRR fulfills the maximality property of a similarity measure.

Proof Suppose $b+c=x$ and x is a positive number. Then Eq. (A2) becomes

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{a[2(a+x)+d]}{(a+x)[(a+x)+d]}. \tag{A9}$$

The above equation simplifies to

$$\mathrm{JCRR}(E_i,\ E_j)=\frac{a(2a+2x+d)}{a(a+d+2x)+dx+x^2}. \tag{A10}$$

To calculate the similarity of an entity with itself, i.e., $\mathrm{JCRR}(E_i,\ E_i)$, it is sure that $x=0$ and $a, d\geq 0$. Using these quantities, Eq. (A10) reduces to

$$\mathrm{JCRR}(E_i,\ E_i)=\frac{a(2a+d)}{a(a+d)}. \tag{A11}$$

Therefore, using Eqs. (A10) and (A11), $\forall E_i, E_j\in\boldsymbol{F}$, the following association will always be

true for any value of a, d, or x, where $a+d+x=P$:

$$\frac{a(2a+d)}{a(a+d)} \geq \frac{a(2a+2x+d)}{a(a+d+2x)+dx+x^2}. \quad \text{(A12)}$$

Appendix B: Propositions for JNMRR

Proposition B1 Let the range of JNMRR be z. Then we have

$$\text{JNMRR}(E_i,\ E_j) = \begin{cases} z = 1.5, & a = P, \\ z \in (0,\ 1.5), & 0 < a < P, \\ z = 0, & a = 0. \end{cases} \quad \text{(B1)}$$

Proof As JNMRR is the combined function of four quantities a, b, c, and d, as shown in Eq. (17), substituting $s = a+b+c$ into Eq. (17), we have

$$\text{JNMRR}(E_i,\ E_j) = \frac{a[3(a+b+c)+2d]}{[2(a+b+c)+d][(a+b+c)+d]}. \quad \text{(B2)}$$

If all features are present in feature vectors of E_i and E_j, that is, $b=c=d=0$, and $a=P$, then the above equation reduces to

$$\text{JNMRR}(E_i,\ E_j) = \frac{a\cdot(3a)}{2a\cdot a} = 1.5. \quad \text{(B3)}$$

Therefore, the maximum similarity value that JNMRR can create is 1.5.

Now, if no common feature is present in feature vectors of E_i and E_j, that is, $a=0$ and $b+c+d \geq 0$, then Eq. (B2) reduces to

$$\text{JNMRR}(E_i,\ E_j) = \frac{0[3(0+b+c)+2d]}{[2(0+b+c)+d][(0+b+c)+d]} = 0. \quad \text{(B4)}$$

Thus, the minimum similarity value that JNMRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of E_i and E_j, i.e., if $a=x$ and $b=c=d=y$, where $x, y > 0$, then Eq. (B2) reduces to

$$\text{JNMRR}(E_i,\ E_j) = \frac{x[3(x+y+y)+2y]}{[2(x+y+y)+y][(x+y+y)+y]}. \quad \text{(B5)}$$

The above equation simplifies to

$$\text{JNMRR}(E_i,\ E_j) = \frac{3x^2+8xy}{2x^2+11xy+15y^2}. \quad \text{(B6)}$$

Eq. (B6) results in a value between 0 and 1.5, if $x, y > 0, \forall E_i, E_j \in \boldsymbol{F}$.

Proposition B2 JNMRR satisfies Definition 1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^P$ and the range is $\mathbb{R}^+$.

Proof JNMRR is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated using only binary values in a feature vector of entities as defined in Section 2.2. Hence, the domain of JNMRR measure is $\{0,1\}^P$. Meanwhile, JNMRR results in a real value, that is, $z \in \mathbb{R}^+$, as proved in Proposition B1:

$$\text{JNMRR}(E_i,\ E_j) = \begin{cases} z > 0, & a \geq 1, \\ z = 0, & \text{otherwise.} \end{cases} \quad \text{(B7)}$$

Proposition B3 JNMRR fulfills the properties of positivity and symmetry of a similarity measure.

Proof First, let us show the positivity. It has been shown in the proof of Proposition B1 that JNMRR creates a similarity value equal to or greater than 0; that is, $\text{JNMRR}(E_i, E_j) \to \mathbb{R}^+, \forall E_i, E_j \in \boldsymbol{F}$.

Next, let us show the symmetry. It is obvious that

$$\text{JNMRR}(E_i,\ E_j) = \text{JNMRR}(E_j,\ E_i). \quad \text{(B8)}$$

Proposition B4 JNMRR fulfills the maximality property of a similarity measure.

Proof Suppose $b+c=x$ and x is a positive number. Then Eq. (B2) becomes

$$\text{JNMRR}(E_i,\ E_j) = \frac{a[3(a+x)+2d]}{[2(a+x)+d][(a+x)+d]}. \quad \text{(B9)}$$

The above equation simplifies to

$$\text{JNMRR}(E_i,\ E_j) = \frac{a(3a+3x+2d)}{a(2a+3d+4x)+d^2+3dx+2x^2}. \quad \text{(B10)}$$

To calculate the similarity of an entity with itself, i.e., $\text{JNMRR}(E_i,\ E_i)$, it is sure that $x=0$ and $a, d \geq 0$. Using these quantities, Eq. (B10) reduces to

$$\text{JNMRR}(E_i,\ E_i) = \frac{a(3a+2d)}{a(2a+3d)+d^2}. \quad \text{(B11)}$$

Therefore, using Eqs. (B10) and (B11), $\forall E_i, E_j \in \boldsymbol{F}$, the following association will always be true for any value of a, d, or x, where $a+d+x=P$:

$$\frac{a(3a+2d)}{a(2a+3d)+d^2} \geq \frac{a(3a+3x+2d)}{a(2a+3d+4x)+d^2+3dx+2x^2}. \quad \text{(B12)}$$

Appendix C: Propositions for JCJNMRR

Proposition C1 Let the range of JCJNMRR be z. Then we have

$$\mathrm{JCJNMRR}(E_i,\ E_j)=\begin{cases} z=2.5, & a=P,\\ z\in(0,\ 2.5), & 0<a<P,\\ z=0, & a=0. \end{cases} \quad \text{(C1)}$$

Proof As JCJNMRR is the combined function of four quantities a, b, c, and d, as shown in Eq. (18), substituting $s=a+b+c$ into Eq. (18), we have

$$\begin{aligned}&\mathrm{JCJNMRR}(E_i,\ E_j)\\ =&\frac{a[5(a+b+c)^2+5(a+b+c)d+d^2]}{(a+b+c)[2(a+b+c)^2+3(a+b+c)d+d^2]}.\end{aligned} \quad \text{(C2)}$$

If all features are present in feature vectors of E_i and E_j, i.e., $b=c=d=0$ and $a=P$, then the above equation reduces to

$$\mathrm{JCJNMRR}(E_i,\ E_j)=\frac{a\cdot(5a^2)}{a\cdot(2a^2)}=2.5. \quad \text{(C3)}$$

Therefore, the maximum similarity value that JCJNMRR can create is 2.5.

Now, if no common feature is present in feature vectors of E_i and E_j, i.e., $a=0$ and $b+c+d\geq 0$, then Eq. (C3) reduces to

$$\begin{aligned}&\mathrm{JCJNMRR}(E_i,\ E_j)\\ =&\frac{0[5(0+b+c)^2+5(0+b+c)d+d^2]}{(0+b+c)[2(0+b+c)^2+3(0+b+c)d+d^2]}\\ =&0.\end{aligned} \quad \text{(C4)}$$

Thus, the minimum similarity value that JCJNMRR can create is 0.

Lastly, if there exist some common and absent features in feature vectors of E_i and E_j, i.e., if $a=x$ and $b=c=d=y$, where $x, y>0$, then Eq. (C2) reduces to

$$\begin{aligned}&\mathrm{JCJNMRR}(E_i,\ E_j)\\ =&\frac{x[5(x+y+y)^2+5(x+y+y)y+y^2]}{(x+y+y)[2(x+y+y)^2+3(x+y+y)y+y^2]}.\end{aligned} \quad \text{(C5)}$$

The above equation simplifies to

$$\mathrm{JCJNMRR}(E_i,\ E_j)=\frac{5x^3+25x^2y+31xy^2}{2x^3+15x^2y+37xy^2+30y^3}. \quad \text{(C6)}$$

Eq. (C6) results in a value between 0 and 2.5, if $x, y>0, \forall E_i, E_j\in \boldsymbol{F}$.

Proposition C2 JCJNMRR satisfies Definition 1 given in Section 2.2, which states that the domain of a binary similarity measure is $\{0,1\}^P$ and the range is $\mathbb{R}^+$.

Proof JCJNMRR is the combined function of four quantities a, b, c, and d, and all these quantities can be calculated using only binary values in a feature vector of entities as defined in Section 2.2. Hence, the domain of the JCJNMRR measure is $\{0,1\}^P$. Meanwhile, JCJNMRR results in a real value, that is, $z\in\mathbb{R}^+$, as proved in Proposition C1:

$$\mathrm{JCJNMRR}(E_i,E_j)=\begin{cases} z>0, & a\geq 1,\\ z=0, & \text{otherwise.}\end{cases} \quad \text{(C7)}$$

Proposition C3 JCJNMRR fulfills the properties of positivity and symmetry of a similarity measure.

Proof First, let us show the positivity. It has been shown in the proof of Proposition C1 that JCJNMRR creates a similarity value equal to or greater than 0; that is, $\mathrm{JCJNMRR}(E_i,\ E_j)\rightarrow\mathbb{R}^+$, $\forall E_i, E_j\in\boldsymbol{F}$.

Next, let us show the symmetry. It is obvious that

$$\mathrm{JCJNMRR}(E_i,\ E_j)=\mathrm{JCJNMRR}(E_j,\ E_i). \quad \text{(C8)}$$

Proposition C4 JCJNMRR fulfills the maximality property of a similarity measure.

Proof Suppose $b+c=x$ and x is a positive number. Then Eq. (C2) becomes

$$\begin{aligned}&\mathrm{JCJNMRR}(E_i,\ E_j)\\ =&\frac{a[5(a+x)^2+5(a+x)d+d^2]}{(a+x)[2(a+x)^2+3(a+x)d+d^2]}.\end{aligned} \quad \text{(C9)}$$

To calculate the similarity of an entity with itself, i.e., $\mathrm{JCJNMRR}(E_i, E_i)$, it is sure that $x=0$ and $a, d\geq 0$. Using these quantities, Eq. (C9) reduces to

$$\mathrm{JCJNMRR}(E_i,\ E_i)=\frac{a(5a^2+5ad+d^2)}{a(2a^2+3ad+d^2)}. \quad \text{(C10)}$$

The above equation simplifies to

$$\mathrm{JCJNMRR}(E_i,\ E_i)=\frac{5a^2+5ad+d^2}{2a^2+3ad+d^2}. \quad \text{(C11)}$$

Therefore, using Eqs. (C9) and (C11), $\forall E_i, E_j\in\boldsymbol{F}$, the following association will always be true for any value of a, d, or x, where $a+d+x=P$:

$$\frac{5a^2+5ad+d^2}{2a^2+3ad+d^2}\geq\frac{a[5(a+x)^2+5(a+x)d+d^2]}{(a+x)[2(a+x)^2+3(a+x)d+d^2]}. \quad \text{(C12)}$$