SindhUniv. Res. Jour. (Sci. Ser.) Vol. 51 (03) 419-426 (2019)

http://doi.org/10.26692/sujo/2019.03.67

SINDH UNIVERSITYRESEARCH JOURNAL (SCIENCESERIES)

Parametric Based Clustering of Reusable Components

M. H. ZAFAR, M. ILYAS, S. RAZZAQ, F. MAQBOOL, W. AHMAD*, S. M. ADNAN*

Department of Computer Science and Information Technology, University of Sargodha, Sargodha, Pakistan

Received 07th December 2018 and Revised 12th July 2019

Abstract: Building new software by using existing software that has been developed by using reusability principles is known as software reuse. It results in reduction of effort and time to develop software. It also increases reliability, portability, maintainability and productivity of software product. But the problem is a lack to symmetric way to store reusable components so that retrieval of component done with less time. One of the solutions is to classify reusable components. We use clustering technique to classify reusable components because clustering results in reduction of search space by cataloguing similar objects together. In this research we propose a framework that is used to understand the process of clustering and to give a practical shape to this framework. In this framework software reusable components are provide with their associated parameters. On the basis of these parameters, software reusable components are clustered. Finally proposed clustering algorithm is evaluated by applying this algorithm on different software reusable components. Presentation of results is in the form of table and graph which shows the successful clustering of reusable components.

Keywords: Software Components, Software Clustering, Software Reusability, Software Classification.

1. INTRODUCTION

New technologies are required in software engineering to enhance the quality of software product and to reduce effort and time to develop software. Since 1968, new horizons are open when software reuse has introduced (AL-Badareen, *et al.*, 2011). For commercial and business systems, software reuse is a main development approach (Sommerville, 2007). Whole system or a part of system can be used in other systems by using software (Haiguang, 2010) Productivity, quality and reliability of software product are increased by using software reuse (AL-Badareen, *et al.*, 2011), (Singh *et al.*, 2010) (Amin *et al.*, 2010). It also minimizes the risk and ensures to deliver software product in time. It also encourages the developing team because, in current domain, they have working experience (Singaravel *et al.*, 2010) (Sandhu, *et al.*, 2010).

Despite of these benefits, there is a problem to store software reusable components in a way through which retrieval of software components becomes easy (AL-Badareen, *et al.*, 2011). So for storing and classification of software reusable components there must be an appropriate way. By adopting this way we can retrieve and access software reusable components with less effort and time. These are the major advantages of reusability (Ilyas *et al.*, 2013).

In this research we try to eliminate this problem by classifying software reusable components. We use clustering technique to classify reusable components. In clustering (Abdulfa, and Mikki 2013), data is divided into groups of objects that are similar. Every group, that is actually a cluster, comprises of similar objects within the cluster and dissimilar objects into other clusters. By using clustering technique similar objects are grouped together. As a result search space is reduced (Srinivas, *et al.*, 2013). Data that is useful, meaningful or both is divided into groups in clustering (Fokaefs, *et al.*, 2009). Clustering is also useful in making decisions, various fact finding pattern analysis, image segmentation, retrieval of documents, machine learning situations and in the field of data mining (Kaya, 2005).

A framework named as P_mBC (Parametric Based Clustering) is proposed. This framework helps us to understand the process of clustering. This framework consists of two main processes, threshold checking process and clustering process. In threshold checking process, a comparison between threshold value and existing number of clusters is made to lemmatize the maximum number of clusters according to threshold value. In clustering process, clusters are assigned to software reusable components on the basis of the parameters that are attached with components. A clustering algorithm is proposed to give practical shape to P_mBC framework. Evaluation of proposed algorithm is done by applying this algorithm on a set of software reusable components that has been collected from different websites.

++Corresponding Authors: E-mail: Husnainzafarmscs@gmail.com, Muhammad.Ilyas@uos.edu.pk Saad.Razzaq@uos.edu.pk Fahad.Maqbool@uos.edu.pk Wakeel.Ahmad@uettaxila.edu.pk Syed.Adnan@uettaxila.edu.pk
*Department of Computer Science, University of Engineering and Technology Taxila, Taxila, Pakistan

Rest of the paper compromise the following sections. Section 2 describes related work. Proposed framework and proposed clustering algorithm are given in Section 3 and Section 4 respectively. Results and discussions are explained in Section 5. Section 6 compromises the conclusion and future work.

2. RELATED WORK

(Ilyas *et al.,* 2013 suggested a framework for making the reusability organized and formal. In this framework, at each reusability level, quality earning criteria are defined to observe the requirements of reusability, extraction of reusable components, classification and integration of reusable components with new systems effectively. These criteria are applied to get the quality and satisfactory results by formalizing each activity of reusability.

(Kamalraj *et al.,* 2011) stated that when we reuse a component from a repository and then lot of time is required to search it, so it is necessary to arrange a component into an effective manner so time to extract a component reduces. (Kamalraj *et al.* 2011) propose a clustering method to extract reusable component in an effective manner to reduce time for searching component. In this method main focus is on the stability metric.

(Srinivas *et al.,* 2013) propose a clustering algorithm. Text documents, text file or software components can be classified by this algorithm. (Srinivas *et al.,* 2013) defines new similarity measure called hybrid XOR Function. By applying this XOR function a similarity matrix is developed. Input of the proposed algorithm (Srinivas *et al.,* 2013) is similarity matrix and output is clusters.

(Mamaghani and Meybodi 2009) proposed two clustering algorithms. On the basis of learning automata, first algorithm is proposed. Second is obtained by combining genetic algorithms and object migrating learning automata. These algorithms are based on approximation. Many algorithms have been proposed in this prospective like Bunch, Brain Mitchell and DAGC (Digital automatic gain control) algorithm (Parsa and Bushehrian 2005). (Bhagwan and Oberoi, 2011). A comparison of proposed algorithm is made with the above algorithms. One of the disadvantages of the previous algorithms is too stuck in local optima. The results of comparison show that second algorithm not only removes this problem but it also increases the speed of the process of searching. One more advantage is that the stability of this algorithm is very high. Software system is modeled by using MDG (Millennium Development Goals) graph in first step by the system designer. In this graph there are two things, nodes and edges. Modules of system are represented in the form of nodes and edges are used to represent relationships. After the creation of MDG graph, different clustering algorithms are applied to create partitioned MDG. The aim of this is to partition the components of system into different clusters so that the subsequent organizations are able to reduce and increase inter-connectivity and intra-connectivity respectively.

(Bhagwan and Oberior 1999) use clustering of software modules to obtain full benefits form the reusability. Selection of good quality software is done automatically on the basis of association between modules. Association is usually dependencies among modules. LOC (Line of code) and Code Clones are used for finding relationship between modules. To meet the research objective, this paper uses the HC (Hierarchical clustering) algorithm like Agglomerative method (Kamalraj *et al.,* 2011) and NHC (Non-hierarchical clustering) algorithm like K-Mean method. For this purpose this research proposes an algorithm.

(Manhas *et al.,* 2010) stated that if we use already existing components then we can reduce the cost of software development. He uses many metrics for the classification of software component. To evaluate the result, he uses Back propagation based neural networks.

3. PROPOSED METHODOLOGY

A framework P_mBC has been proposed to cluster reusable components in a systematic way. Threshold checking process and clustering process are the two processes of P_mBC framework. P_mBC framework is shown in **(Fig. 1)**. Processes of this framework are given below.

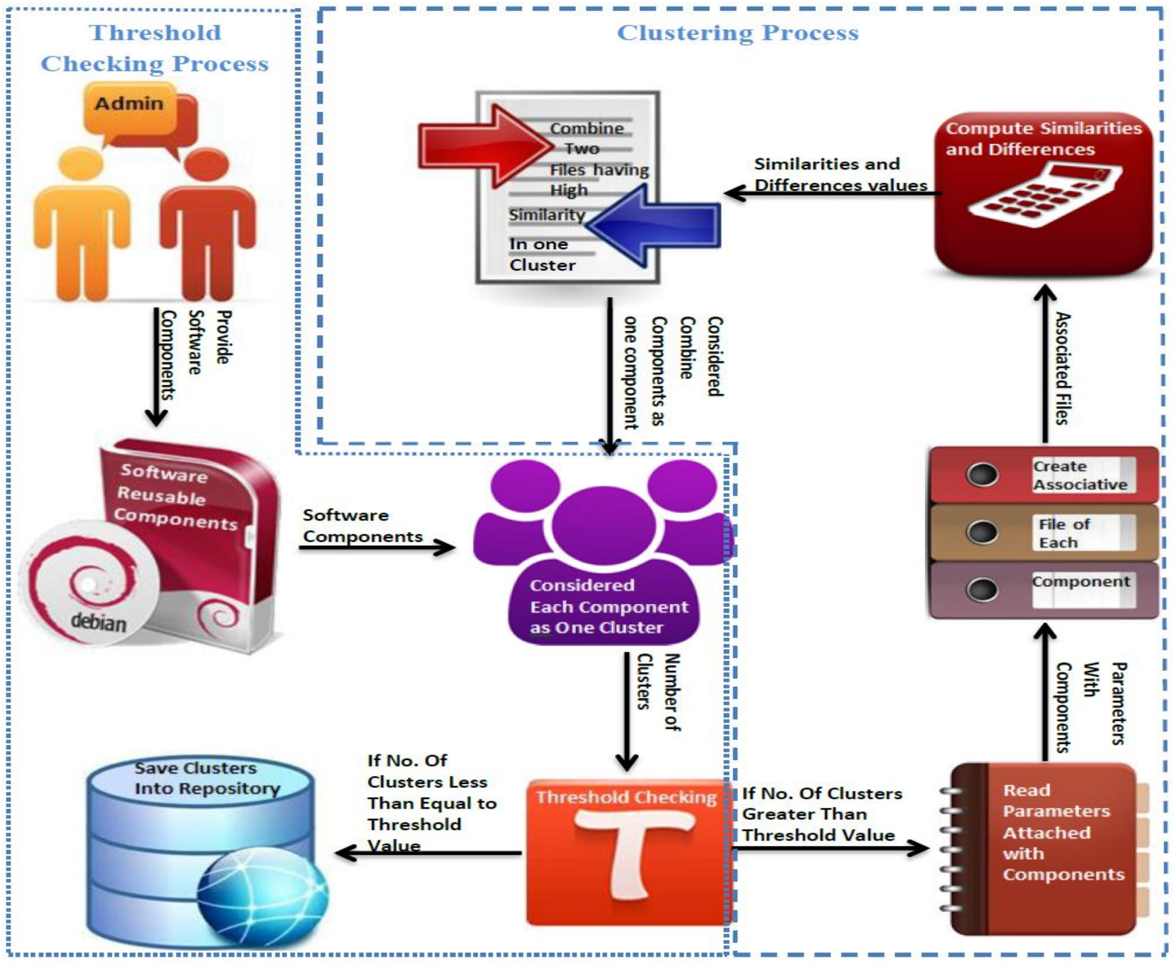
Threshold Checking Process
Clustering Process
Admin
Provide Software Components
Software Reusable Components
debian
Software Components
Considered Each Component as One Cluster
Number of Clusters
Threshold Checking
If No. Of Clusters Less Than Equal to Threshold Value
Save Clusters Into Repository
If No. Of Clusters Greater Than Threshold Value
Read Parameters Attached with Components
Parameters With Components
Create Associative File of Each Component
Associated Files
Compute Similarities and Differences
Similarities and Differences values
Combine Two Files having High Similarity In one Cluster
Considered Combine Components as one component

Fig. 1. P$_m$BC framework

3.1 Threshold Process

In this process a threshold value is provided by the user. This value acts like the value of K in k-means algorithm (Zafar, and Ilyas 2015). In k-means algorithm k denotes the total number of clusters, similarity in this framework threshold value represents total number of clusters to be built. This process starts by considering each component as one cluster. So we have total number of clusters equal to the total number of components we have. After this we compare threshold value with total number of clusters. If total number of clusters is less than or equal to threshold value then it means we already achieve total number of clusters and these clusters will saved into repository. But if total number of clusters is greater than threshold value then components will move to clustering process.

3.2 Clustering Process

This process is further divided into four processes. These are Associative File Creation, Associative File Pair Creation, Computation of Similarities and Differences and last process is Combining Files. These processes are discussed below.

3.2.1 Associative File Creation

This process starts by reading all the parameters attached with components. After reading these parameters associative files are created for each component. In these files we assign a numeric value to each value of each parameter. For each component we have four parameters. First parameter is category. For each value of category we assign a number. For example we have three values of category like game, algorithm, and AI. So we assign number 1 to game, 2 to algorithm and 3 to AI and if another value exists then we assign number 4 to that value. Second parameter is language. Similarity we assign a number to each value of language parameters. Suppose we have two values of language parameters like java and C++ then we assign number 1 to java and number 2 to C++. Third parameter is platform. Like previous two parameters we assign a number to each value of platform parameters. Forth parameter is programming technique. Values of programming technique parameter are fixed. These may be simple, Structured or OOP (Object oriented programming). Other than these values, no other value could assign to programming technique parameters, so maximum values for this parameter are 3. Number 1 is used for simple, number 2 assign to structured and number 3 represents OOP value. **(Table-1)** shows the input software components with parameters and **(Table-2)** represents their associative file.

Table 1. Input software components with parameters

Components	Category	Language	Platform	Programming Technique
A	C++ Feature	C++	Windows	Structured
B	Game	C++	Windows	Simple
C	Game	Java	Windows, Unix, Linux	OOP
D	Utilities	C	Windows, Unix, Linux	Simple
E	Algorithm	Julia	Windows, OS X, Linux	OOP

Table 2. Corresponding files of input software components

Component	Category	Language	Platform	Programming Technique
A	1	1	1	2
B	2	1	1	1
C	2	2	2	3
D	3	3	2	1
E	4	4	3	3

3.2.2 Associative File Pair Creation

After creating associative files we make the pairs of these files. Total number of pairs are computed by combination formula [17] given in equation 1. In this equation n is total number of components, $\binom{n}{k}$ is pair at a time and k = 2.

$$\binom{n}{k} = {n!}/{(k!\,(n-k)!)} \qquad (1)$$

The pairs of associative file given in table 2 are given below.

(A,B), (A,C), (A,D), (A,E), (B,C), (B,D), (B,E), (C,D), (C,E) and (D,E).

In this approach (A,B) and (B,A) have same values so we take one pair.

3.2.3 Computation of Similarities and Differences

After making the pairs of associative files we take all pairs and one by one we compute similarities and differences between a pair by using equation 2 and 3 respectively. In this process we match the parameters of two files. If the value of any parameter of one component matches with the value of that parameter of second components then we increase the value of similarity by one and if there is no match then we increase the value of difference by one. In **(Table-3)** similarities and differences are computed between the pairs that are given in section (3.2.2).

Table 3. Similarities and differences values

Pairs	Similarities	Differences
(A,B)	2	2
(A,C)	0	4
(A,D)	0	4
(A,E)	0	4
(B,C)	1	3
(B,D)	1	3
(B,E)	0	4
(C,D)	1	3
(C,E)	1	3
(D,E)	0	4

$$Sim(A,B) = \sum_{i=1}^{4} Count(i) \quad where \quad Count\,(i) = \begin{cases} 1 & when\ A_{P_i} = B_{p_i} \\ 0 & when\ A_{P_i} \neq B_{P_i} \end{cases} \qquad (2)$$

$$Diff(A,B) = 4 - Sim(A,B) \qquad (3)$$

A_{P_i} = ith parameter of Component A

B_{P_i} = ith parameter of Component B

Because each component has four parameters and similarity is measured by counting the same parameters of two components so maximum value of similarity will four. This is the reason that we subtract similarity value from four in equation 3.

3.2.4 Combining Files

After computing similarities and differences, the two files having higher similarities as compared to others are combined into one cluster. As a result total number of clusters decreased by one and then second iteration of algorithm starts. This process ends in threshold checking process when total number of clusters becomes equal to threshold value.

4. <u>PROPOSED CLUSTERING ALGORITHM</u>

We like to discuss input for proposed algorithm before discussing proposed algorithm.

4.1 Input of Proposed Clustering Algorithm

Software reusable component is the input of our clustering algorithm that is basically a file. In this file both source code and its related parameters exist. This file consists of two portions. First portion contains parameters and second potion has source code of software reusable component. An example of input file is given below.

Parameters

Language:	C++
Category:	C++ Features
Technique:	Simple
Platform:	Windows

Source Code

```
#include<iostream>
using namespace std;
void func1(char* str);
void func2(int val);
int main()
{
func1(“character string passed\n”);
func1(123);
return(system(“pause));
}
void func1(char* str)
{
cout<<”string value:=”<<str<<endl;
}
void func1(int val)
{
cout<<”integer value :=”<<val<<endl;
}
```

4.2 Proposed Clustering Algorithm

Pseudo code of proposed algorithm is given below.
Input: Software components with parameters
Output: T Clusters

```
Begin
    T ← Threshold Value defined by the user
    Step1
    K ← Total number of clusters by considering each component as one cluster
        If K less than or equal to T
            Store Cluster into repository
            End Algorithm
        Else
            For i ← 1 to 4
                Parameters[i] ← Input Component Parameters[i]
            For i ← 1 to 4
                For j ← 1 to value of Parameter[i]
            Total_Number_of_Pairs ← 0
            For i ← 1 to K-1
            j ← i + 1
                For m ← j to K
                    Pair[i] ← (Component[i],Component[m])
                    Total_Number_of_Pairs increment by one
                    Increment j by one
            Sim[Pair] ← 0
            Diff[Pair] ← 0
            For i ← 1 to Total_Number_of_Pairs
                For j ← 1 to 4
If(value[Pair[i]Component[1]Parameter[j] == value[Pair[i]Component[2]Parameter[j])
Sim[Pair[i]] increment by one
Else
    Diff[Pair[i]] increment by one
            Pair_Of_Maximum_Similarity ← Sim[Pair] of Maximum Value
            Combine two components of Pair_Of_Maximum_Similarity
            Considered two combined components as one component
            Go to step1
End
```

Steps of clustering algorithm are given below.

1. Each component considered as one cluster and check if total number of clusters less than or equal to threshold value then stop algorithm otherwise go to step
2. Read parameters of each component and create associative file for each component.
3. Make pairs of components.
4. Compute Similarities and differences.
5. Combine two components into one cluster then go to step 1.

5. RESULTS AND DISCUSSIONS

Eight software reusable components had been taken to evaluate the proposed clustering algorithm. We applied this algorithm on these eight software reusable components to cluster these components according to their parameters. The threshold value in this scenario was 4. Threshold value 4 means, we had 4 clusters at the end. Here we present the first iteration of algorithm on these eight software reusable components. **(Table-4)** shows the eight components with their parameters.

Table 4. Input Software Components with Parameters

Component	Category	Language	Platform	Programming Technique
1	C++ Feature	C++	Windows	Structured
2	Game	C++	Windows	Simple
3	Algorithm	C++	Windows	Simple
4	Full Project	C++	Windows	OOP
5	Utilities	C++	Windows	Simple
6	Utilities	Java	Windows, Unix, Linux	OOP
7	Utilities	C	Windows, Unix, Linux	Simple
8	Algorithm	Julia	Windows, OS X, Linux	OOP

As we considered each component as one cluster so initially we had eight clusters and our threshold value was 4 so we moved toward next step that was associative file creation. In Table 5 associative file of these eight components are given.

The pairs of associative file given in **(Table -5)** are given below.
(1, 2), (1, 3), (1,4), (1,5), (1,6), (1,7), (1,8), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (3,4), (3,5), (3,6), (3,7), (3,8), (4,5), (4,6), (4,7), (4,8), (5,6), (5,7), (5,8), (6,7), (6,8), (7,8).

Similarity and difference values that were calculated for the above pairs are given in **(Table-6).**

According to the similarity values given in table 6, three pairs had higher similarity values then the other pairs. These were (2,3), (2,5) and (3,5). Now we had to take one pair and combined the files of the pair. As we had three pairs having same higher similarity so we applied "first come first serve" rule to take one pair. So according to this rule we took first pair that is (2,3) and combined the components 2 and 3. These two combined components considered as one component in next iteration. Therefore we had 7 components in next iteration which means we had 7 clusters at the start of next iteration. Similarly in each iteration, number of clusters decreased by one. This process remained continue until number of clusters became equal to threshold value. In the given scenario, after four iterations number of clusters became equal to the threshold value. **(Table-7)** shows the software reusable components after clustering. After getting results, we transformed these results in the form of graphs. **(Fig. 2)** shows these results. Bar chart was used to draw graph.

Table 5. Corresponding Files of Input Software Components

Component	Category	Language	Platform	Programming Technique
1	1	2	1	2
2	4	2	1	1
3	5	2	1	1
4	2	2	1	3
5	3	2	1	1
6	3	2	2	3
7	3	1	2	1
8	5	4	3	3

Table 6. Similarities and Differences Values

Pairs	Similarities	Differences	Pairs	Similarities	Differences
(1, 2)	2	2	(3,5)	3	1
(1, 3)	2	2	(3,6)	1	3
(1,4)	2	2	(3,7)	1	3
(1,5)	2	2	(3,8)	1	3
(1,6)	1	3	(4,5)	2	2
(1,7)	0	4	(4,6)	2	2
(1,8)	0	4	(4,7)	0	4
(2, 3)	3	1	(4,8)	1	3
(2,4)	2	2	(5,6)	2	2
(2,5)	3	1	(5,7)	2	2
(2,6)	1	3	(5,8)	0	4
(2,7)	1	3	(6,7)	2	2
(2,8)	0	4	(6,8)	1	3
(3,4)	2	2	(7,8)	0	4

Table 7. Software Reusable Components After Clustering

Clusters	Component	Category	Language	Platform	Programming Technique
1	1	C++ Feature	C++	Windows	Structured
2	2, 3,5,6, 7	Game, Algorithm, Utilities,	C++, Java, C	Windows	Simple, OOP
3	4	Full Project	C++	Windows	OOP
4	8	Algorithm	Julia	Windows, OS X, Linux	OOP

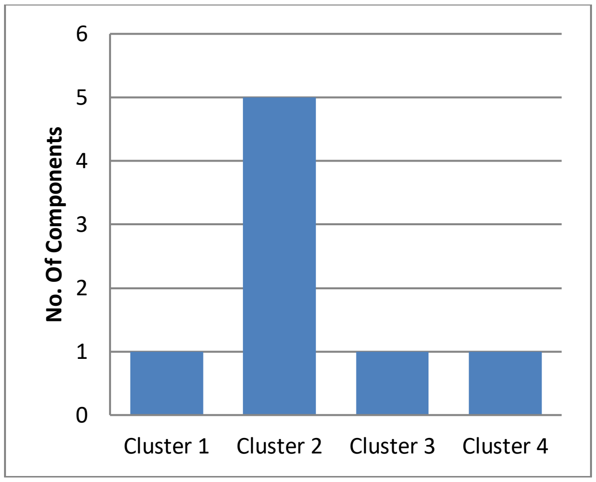

Fig. 2. Graphical Results of Clustering

6. CONCLUSION

The usage of software reuse is increasing day by day in software engineering so that's why the graph of importance of software reuse is growing. But the problem is to store software reusable components in a symmetric way. To solve this issue component must be classified in an efficient way. Classification is done in many ways but clustering is very effective technique for classification. So we propose a framework to understand the process of clustering. This framework consists of two main processes that are threshold process and clustering process. In threshold process, threshold value is compared with total number of clusters. If total number of clusters are less than or equal to threshold value, it means we achieve our required clusters and the process ends. But if total number of clusters is greater than threshold value then we proceed toward next process that is clustering process. In clustering process first we read the parameters attached with source code then we create associative file of software reusable components. After making associative file, we make the pairs of components from associative file. Then we take all pairs one by one and computer similarities and differences. After this we take one pair having higher similarity value then the other pairs and combine the components of that file. Then we move towards threshold process. For giving the practical shape to framework, we propose clustering algorithm and take 8 components to evaluate proposed clustering algorithm.

Our future aimed to generalize proposed clustering process. In proposed framework software components are clustered with specific format so later on we want to cluster software components without specific format and parameters for clustering software components extract at runtime. Another aim to analyze proposed algorithm is by making simulation in MATLAB and do statistical analysis.

REFERENCES:

AL-Badareen, A. B., M. H. Selamat, M. A. Jabar, J. Din and S. Turaev. (2011) Reusable Software Component Life Cycle. Int J Comput 5: 191-199.

Amin F., A. K. Mahmood, and A. A Oxley (2010) Proposed Reusability Attribute Model for Aspect Oriented Software Product Line Components. In: ITSim International Symposium in Information Technology; 15-17; 1138-1141.

Abdulfa, S. I., and M. Mikki (2013) K-means algorithm with a novel distance measure. Turk J Elec Eng & Comp Sci. 21: 1665-1684.

Bhagwan J., and A. Oberoi, (2011) Software Modules Clustering: An Effective Approach for Reusability. J Inform Eng APPL; 4: 18-28.

Fokaefs, M., N. Tsantalis, A. Chatzigeorgiou, and J. Sander (2009) Decomposing Object-Oriented Class Modules Using an Agglomerative Clustering Technique. In: ICSM 2009 IEEE International Conference on; 20-26, 93-101.

Haiguang F. (2010) Modeling and Analysis for Educational Software Quality Hierarchy Triangle. In: Seventh International Conference on Web Based Learning; 20-22 August 2008; Jinhua, China: IEEE. 14-18.

Kaya M. (2005) An Algorithm for Image Clustering and Compression. Turk J. Elec Eng & Comp Sci. 13: 79-91.

Kamalraj, R., A. R. Kannan, and P. Ranjani. (2011) Stability-based Component Clustering for Designing Software Reuse Repository. Int J Comput Appl. 27 33-36.

Ilyas M., M. Abbas, and K. A. Saleem, (2013) Metric Based Approach to Extract, Store and Deploy Software Reusable Components Effectively. Int J Comput Sci. 10: 79-91.

Mamaghani S. and M. R. Meybodi (2009) Clustering of Software Systems using New Hybrid Algorithms. In: CIT 2009 IEEE Ninth International Conference on Computer and Information Technology 20-25.

Manhas S., P. S. Sandhu V. Chopra, and N. Neeru (2010) Identification of Reusable Software Modules in Funcion Oriented Software System using Neural Network Based Technique. World Acad Sci. Eng. Tech; 67: 823-827.

Orwant J., J. Hietaniemi, and J. Macdonald (1999) Probability. In: Orman A, Orwant J, editors. Mastering Algorithms with Perl. USA: O'Reilly & Associates, Inc., 566-598.

Parsa S. and O. A. Bushehrian (2005) New Encoding Scheme and a Framwork to investigate Genetic Clustering Algorithms. J. Res Pract Inf. Tech; 37. 127-143.

Sommerville I. (2007) Software reuse. Software Engineering. 8th ed. China: China Machine Press,. 415-438.

Singh S., M. Thapa, S. Singh, and G. Sing. (2010) Software Engineering-Survey of Reusability Based on Software Component. Int. J. Comput Appl. 8: 39-42.

Singaravel G., V. Palanisamy, and A. Krishnan (2010) Overview Analysis of Reusability Metrics in Software Development for Risk Reduction. In: ICICTI 2010 International Conference on Innovative Computing Technologies; 12-13, Tamil Nadu, India: ICICTI, 1-5.

Sandhu, P. S., S. Aashima, and P. A. Kakkar, (2010) Survey on Software Reusability. In: ICMET 2010 2nd International Conference on Mechanical and Electrical Technology; 10-12. Singapore: IEEE, 769-773.

Srinivas, C., V. Radhakrishna, and G. Rao. (2013) Clustering Software Components for Program Restructuring and Component Reuse Using Hybrid XOR Similarity Function. In: AASRI 2013 Conference on Intelligence Systems and Control; 319-328.

Zafar, M. H. and M. A. Ilyas (2015) Clustering Based Study of Classification Algorithms. Int J. Database Theor Appl. 8: 11-22.