FP- International Journal of Computer Science Research (IJCSR)
Volume 2, Issue 2, Pages 132-142, June 2015

Component Retrieval and Clustering for Component Reusability Paradigm

Iqbaldeep Kaur
Associate Professor
Chandigarh University
Gharuan,140301, India
iqbaldeepkaur.cu@gmail.com

Satvir Kaur
Research Scholar
Chandigarh University
Gharuan, 140301, India
savimadahar29@gmail.com

ABSTRACT
With the increasing need of quick software development in the software industry, every software company has started incorporating developing the software considering the reusability of software components. This has led to emergence of component based software engineering. CBSE means the development of any software by reusing already existing components by simply embedding them as such or slightly modifying it. Reusability of components decreases the development time and effort as in CBSE reusable component are not developed from scratch. This paper gives the brief introduction to the CBSE, process of component retrieval and techniques used for component retrieval. Component clustering reduces the search space to search components in a particular domain. In this paper various algorithms are discussed that can be used to cluster software components and to retrieve required components.

General Terms
Reusability, software component.

Keywords
CBSE, component clustering, component retrieval.

1. INTRODUCTION

1.1 Component Based Software Engineering
Pl Component Based Software Engineering is an approach to develop software that relies on software reuse [5]. Component based software engineering is the process of developing software by reusing already existing software components rather than developing them from scratch. In CBSE reusing existing software decreases the software development time, cost and effort. Nowadays software are becoming very large and complex, these software products are very difficult to develop from scratch. So it is desirable to reuse already existing software components that can be required in the development of the large software product.

1.2 Evolution of Component Based Software Engineering
The idea of reusing pieces of software has originated from early sixties. In 1968, M.D Mcilroy first introduced the idea of component based software engineering in his paper titled, "Mass Produced Software Components" at the NATO conference on software engineering in Garmisch, Germany [1]. He states the idea that the software may be componentized i.e. software can be developed by reusing pre-developed software components. Mcilroy's inclusion of filters and pipes into Unix operating system was the first implementation of an infrastructure of this idea. Later Brad Cox of Stepstone defined the modern concept of the software component. He defined them as Software IC'c and create an infrastructure and market for these components by inventing Objective C programming language. In early 1990's, IBM led the path by their System Object Model (SOM). Some claim that Microsoft paved the way for actual deployment of component software with OLE and COM. As of 2010 many successful software component models do exist [15].

1.3 Software Component
Software component is a nontrivial, independent piece of software with a specific functionality. Software component can be used as individual or can be embedded with any other software to perform any particular function.

[5] Some basic definitions of software component are:

Councill and Heinmann:

A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard.

Szyperski:

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties.

Brown [3]:

An independently deliverable piece of functionality providing access to its services through interfaces.

1.4 Component Repository

Component repository is a warehouse or database in which reusable software components are stored in a structured manner. A component repository thus organizes stores and manages the reusable components. It should provide searching and retrieval mechanisms for required components. A user should have clear understanding of the contents of the repository in order to make the efficient use of the repository [18]. But storing the components efficiently and retrieving the best matching component with user requirement is very challenging task. When there is large number of components, it becomes even more challenging.

1.5 Component Retrieval Process

As discussed above, component storage and retrieval are the main issues in CBSE. Component retrieval is the process of fetching the required software component from the large collection of components or software repository.

The basic architecture of component retrieval process is as shown in figure 1.

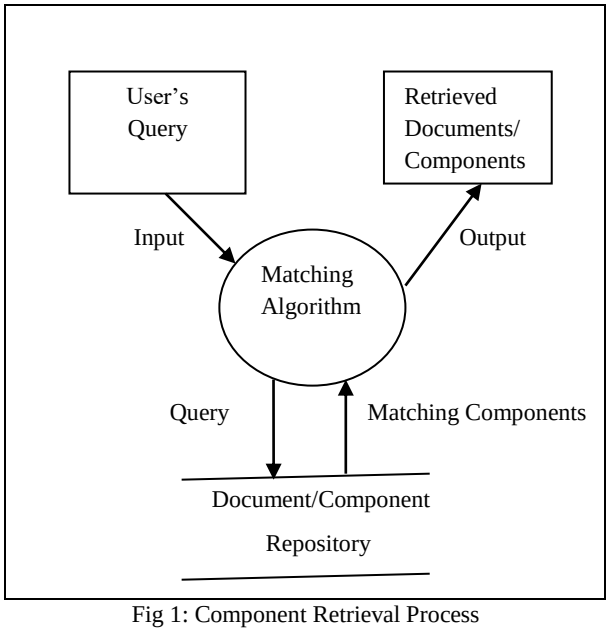

Fig 1: Component Retrieval Process

In retrieval process, there are mainly three parts, user queries i.e. input, matching algorithm and Component repository. User query is the user's request for required component. Matching algorithm matches the user query with the components stored in the repository. It retrieves all the matching components relevant to the user query and gives as output the most relevant components to the user. Component repository is the warehouse where all the components are stored.

2. TECHNIQUES USED FOR COMPONENT RETRIEVAL

Component retrieval is the most challenging task in the component based software engineering. It is required to retrieve the most relevant components to the user query before reusing them. There are various retrieval techniques in order to retrieve the most suitable components. Some of these techniques are discussed as follows:

2.1 Keyword based retrieval technique

Keyword based retrieval is the commonly used method to retrieve components. In this technique, reusable components are stored in the repository along with its description and a keyword. When any component is required for reuse, the user can search it by keyword. All the relevant components that match to keyword are retrieved. Keyword based search reduces the search space but it also results in ambiguity in retrieved components as all possible components are retrieved [19].

2.2 Facet based retrieval technique

Facet classification is the widely used technique for classification of components. A facet schema has facets. Each facet is the important characteristics of the component. Facet means assignment of component to multiple taxonomies i.e. set of attributes. Each facet makes classification of components in component repository from different perspective. Every facet has group of terms. The development of each facet is done by identifying important vocabulary in domain and grouping the relevant terms together into a facet. This method improves the search and retrieval of components and is also helpful in the development of a standard vocabulary for components and their attributes. In faceted classification the schema is divided into several facets and each facet again has several terms [19, 20].

2.3 Semantic based component retrieval technique

Semantic based retrieval technique retrieves the components on the basis of description of the component and its domain [19].This technique retrieves components on the basis of domain or features of the components. Components are retrieved by matching with the meaning given for every component in the repository. Ontology based knowledge base stores the semantic information of the components [9].

2.4 Hypertext based retrieval technique

In this technique the reusable components are organized in non-linear network of nodes and links. Hypertext technique is based on the concept of network of nodes and interconnection between them where each node is a reusable component or unit of textual information and

interconnection between nodes are represented as links [2]. The node from which the link starts is parent node and where the link ends is child node. Components can be searched by traversing the links.

2.5 Knowledge based retrieval technique

In this technique the components are retrieved using reasoning that relates the user query with the components or with an old query. The fuzzy rule set is described in knowledge base. The matching of the query with the component is done using fuzzy rules [7].

3. COMPONENT CLUSTERING TECHNIQUES

Component clustering is the method of grouping the components together into a cluster those having similar characteristics. Dividing the components in clusters make searching and retrieval easy and fast because the search space is reduced. It saves time to search. There are several techniques that are used in clustering some of which are discussed below:

3.1 K-Mean clustering

K-mean clustering is the most popular technique of clustering. In this technique a given data set is divided into k number of clusters. For each cluster, a centroid is defined. The centroids are chosen randomly from the dataset. In the next step, each data item belonging to the given data set is associated to the nearest centroid [29].

[21] Algorithm for K-mean clustering:

Input: D= { $a_1, a_2, \ldots\ldots\ldots, b$ //data set
k //no. of desired clusters
Output: K // set of clusters
Procedure:
assign initial values for means a1, a2, …..ak;
repeat
assign each item to the cluster which has the closest mean;
calculate new mean for each cluster;
until convergence criteria is met;

3.2 K-Mode Clustering

K-mode clustering uses modes to represent data centers. In this, k random modes are selected to represent k clusters. The remaining objects are placed in the cluster of nearest mode. After all the data objects are processed, new modes are selected from every cluster and the whole process is repeated. All the data items are again distributed according to new clusters. Mode changes its position in every iteration [10]. This process is repeated until no mode moves. The clustering algorithm using mode is as given:

K-mode clustering algorithm [10]:

Steps:

1. Select 'k' number of initial cluster centers(modes) from the given set of objects.
2. Compute the distance between each object and the cluster mode; assign the object to the cluster whose mode having shortest distance to the object; repeat this step until all the objects are assigned to the clusters;
3. Select a new mode of each cluster and compare it with the previous mode; If different, go to step 2; otherwise, stop.

K-mode clustering algorithm replaces the distance function used in K-mean algorithm by matching dissimilarity measure.

3.3 K-Medoid Clustering

In K-medoid clustering, data is distributed using medoids of the data sets. In this clustering, k random medoids are selected randomly. So there are k clusters. The remaining data items are placed in the cluster whose medoid is nearest. When all the data items are distributed, a new medoid can be there which can better represent a separate cluster, then the whole process is repeated. All the data items are again distributed according to the new medoid. So in every iteration, medoids change their location step by step. This process is carried out until no medoid moves [13]. So in this clustering the whole data set is divided into k clusters. Following is the algorithm given for K-medoid clustering:

Algorithm for K-medoid Clustering [13]:

Input: ' k' number of clusters to be divided, ' n' number of data items in data set to b clustered.

Output: set of 'k' clusters.

Steps:

1. Arbitrarily choose 'k' objects as initial medoids;
2. Repeat:
   a. Assign each remaining object to the cluster of nearest medoid;
   b. Randomly select a non medoid object;
   c. Compute the total cost of swapping old medoid with newly selected non medoid object;
   d. If the total cost of swapping is less than zero then perform that swapping operation to form a new set of k-medoids;
3. Until no change;

K-medoid algorithm is more robust than k-mean algorithm because it is less influenced by outliers or other extreme values than mean.

3.4 Hierarchical Clustering

In this clustering technique the hierarchy of clusters is build which is also called as dendrogram. It is a tree of clusters in which cluster node has child clusters. Sibling clusters partition the points covered by their common parent [16].

It is of two types:

Agglomerative- It is bottom up clustering algorithm. It first assume each data item a single cluster and moving up it iteratively merges clusters into larger and larger clusters until all data items form a single cluster or certain objective criteria is met.

Divisive- it is top down clustering method. In this clustering method firstly all the data items are grouped into a single cluster, then successively splitting the clusters into sub-clusters moving down until a cluster of single data item is obtained [16].

Hierarchical Clustering Algorithm

For the set of N data items and N*N similarity or distance matrix, the process of hierarchical clustering is:

1. Start by assigning each item to a cluster, so that if we have N items, we now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.
2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now we have one cluster less.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into K number of clusters.

3.5 XOR Similarity Function based Clustering

This component clustering technique was proposed by Chintakindi Srinivas et.al [24]. In this technique the components are classified on the basis of similarity function using hybrid XOR. In this first the frequent item sets are obtained for each document using existing association rule mining algorithm. Then a Boolean matrix is formed with rows indicating documents and columns indicating unique frequent item sets. Then binary feature vector is computed for each document pair by redefining the XOR function as hybrid XOR logic with little modification in the function introducing high impedance variable as Z.

Truth table for hybrid XOR Similarity Function is defined as follows:

A	B	S(A,B)
0	0	Z
0	1	1
1	0	1
1	1	0

Algorithm for Clustering:

Begin of Algorithm

Steps:

1. For each document D do

Begin

a. Remove stop words and stemming words from each document.
b. Find unique words in each document and count of the same.
c. Find frequent item sets of each document
End for
2. Form a word set W consisting of each word in frequent item sets of each document.
3. Form Dependency Boolean Matrix with each row and column corresponding to each Document and each word respectively For each document in document set do
Begin
For each word in word set do
Begin
If (word Wk in Word set W is in document Di)
Begin
Set D[Di, Wk] = 1
Else
Set D[Di, Wk] = 0
End if
End for
End for
4. Find the Feature vector similarity matrix by evaluating similarity value for each document pair applying Hybrid XOR Function defined in table 1 to obtain the matrix with feature vectors for each document pair.
5. Replace the corresponding cells of matrix by count of number of zeroes in tri state feature vector.
6. At each step, find the cell with maximum value and document pairs containing this value in the matrix. Group such document pairs to form clusters. Also if document pair (I, J) is in one cluster and document pair (J,

K) is in another cluster, form a new cluster containing (I, J, K) as its elements.
7. Repeat Step6 until no documents exist or we reach the stage of first minimum value leaving zero entry.
8. Output the set of clusters obtained.
9. label the clusters by considering candidate entries.

End of algorithm

This algorithm is dynamic in nature i.e. it form the component clusters dynamically. It does not need to predefine the number of clusters as in K-Mean, K-Mode and K-Medoid algorithms**.**

3.6 XNOR Similarity Function based Clustering

In this classification technique components are clustered using similarity function using XNOR function. Firstly frequent item sets are obtained using association rule mining algorithm, then Boolean matrix is formed with rows indicating documents and columns indicating unique frequent item sets from each document. Then binary feature vector is computed for each document pair by redefining the XNOR function as hybrid XNOR logic with little modification in the function introducing high impedance variable as Z [30].

Truth Table for Hybrid XNOR Similarity Function is given as:

A	B	S(A,B)
0	0	Z
0	1	0
1	0	0
1	1	1

Clustering Algorithm:

Input: Document set, frequent items.
Output: set of clusters.
Begin of Algorithm
Steps:
1. For each document D do
Begin
   a. Remove stop words and stemming words from each document.
   b. Find unique words in each document and count of the same.
   c. Find frequent item sets of each document

End for
2. Form a word set W consisting of each word in frequent item sets of each document.
3. Form Dependency Boolean Matrix with each row and column corresponding to each Document and each word respectively
For each document in document set do
Begin
For each word in word set do
Begin
If (word wk in Word set W is in document Di)
Begin
Set D[Di, wk] = 1
Else
Set D[Di, wk] = 0
End if
End for
End for
4. Find the Feature vector similarity matrix by evaluating similarity value for each document pair applying Hybrid XNOR Function defined in table 1 to obtain the matrix with feature vectors for each document pair.
5. Replace the corresponding cells of matrix by count of number of zeroes in tri state feature vector.
6. At each step, find the cell with maximum value and document pairs containing this value in the matrix.
Group such document pairs to form clusters. Also if document pair (X,Y) is in one cluster and document pair
(Y, Z) is in another cluster, form a new cluster containing (X, Y, Z) as its elements.
7. Repeat Step6 until no documents exist or we reach the stage of first minimum value leaving zero entry.
8. Output the set of clusters obtained.
9. Label the clusters by considering candidate entries.

End of algorithm

This algorithm also dynamically forms component clusters.

Table 1. Comparison of Component Clustering Algorithms

Sr. No	Algorithm Name	Technique Used	No. of clusters(k) specification	Sensitive to Outliers	Efficiency	Type of Data
1	K-Mean	Euclidean Distance Function	Predefined	Yes	Comparatively more	Numeric data only.
2	K-Mode	Matching Dissimilarity measure	Predefined	Less than K-Mean	Average	Both numeric and Categorical data
3	K-Medoid	Distance Function	Predefined	No	Comparatively less	Both numeric and categorical data
4	Hierarchical Clustering	Cost Functon	Dynamic	Less	Less	-
5	XOR Based Clustering	HXOR Similarity Function	Dynamic	No	High	Text documents and software components
6	XNOR Based Clustering	HNOR Similarity Measure	Dynamic	No	High	Text documents and software components

5. VARIOUS ALGORITHMS FOR COMPONENT RETRIEVAL

Various types of algorithms are used in component based software engineering for component storage and retrieval process such as evolutionary algorithms like GA, Artificial intelligence or learning based algorithm like Neural network, Knowledge based algorithms and probabilistic algorithm. Genetic algorithms are based on natural selection. Neural networks are learning algorithms that are based on the way the human brain learns. Knowledge based algorithms are based on the knowledge stored in the knowledge base and make decisions by using inference rules. Probabilistic algorithms are based on the estimation of relevance probability of a component with the user

6. query [4]. Various algorithms used for clustering (storage) and retrieval of components, are as follows:

4.1 Genetic Algorithm

[4]Genetic algorithm is based on evolution by natural selection. GAs update a population of individuals iteratively. In every iteration of population, the individuals are evaluated using a fitness function. New population of individuals is generated by applying genetic operators such as Crossover and Mutation to create new offspring.

Following is the GA in pseudo code:

Algorithm: GA(n, χ, μ)
//**Initialize generation 0:**
k:=0;
:= a population of n randomly- generated individuals;
//**Evaluate** :
Compute *fitness*(i) for each i ϵ
do
{
//**Create generation** k **:**
//**1. Copy:**
Select(1-χ)× n members of and insert into p;
//**2. Crossover**:
Select χ × n members of p; produce offspring; insert offspring into p
//**3. Mutate**:
Select μ× n members of p; invert a randomly selected bit in each;
//**Evaluate** p**:**

Compute *fitness*(i) for each i ϵ ;
// **Increment:**
$k = k$ **:**
}
while fitness of fittest individual in is not high enough;
return the fittest individual from p_k

4.1.1 Genetic Operations:

1. Selection and Reproduction: An initial population is developed and an individual that best fits the whole population is selected.

2. Crossover: From the set of fit individuals, crossing takes place between any two fit individuals. In this the bits are exchanged.

3. Mutation: After crossover the bits are changed. For example the entire bit 0 is changed to 1 and the bit 1 is changed to 0 [4].

4.2 Neural Network

Neural network is the simplified model of the biological neuron system. In neuron network, information is represented in form of weighted interconnected nodes. Neural networks are self processing networks i.e. there is no need of external program to operate neural network. The neural network process itself with its intelligent behavior. The model for neural network is based on this theory.

Mathematical model of neural network if shown in Figure 2.

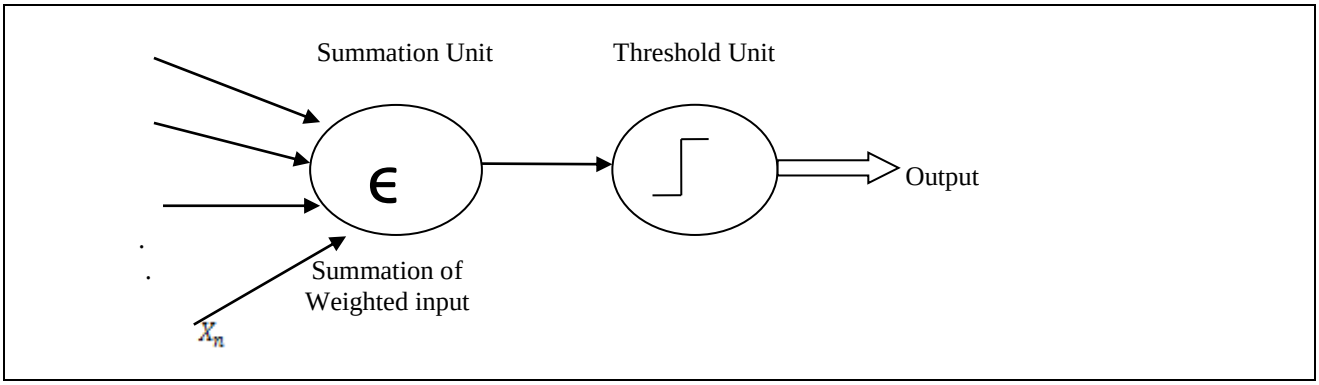

Figure 2. Neural Network Model [28]

In the above figure, X_1, X_2, X_3 …….. X_n are the inputs and $w_1, w_2, \ldots w_n$ are the weights associated with these inputs links. These weights model the acceleration of inputs. If an effective synapse transmits a stronger signal it will have larger weight [28].

Algorithm for Component Retrieval using PSO-RBF Neural Network

[12] Guo Su-wei presents Particle Swarm Optimization algorithm and Radial Basis Function Neural Network for software component retrieval. RBFNN i.e. RBF neural network is a feed forward neural network. It solves optimization problems. PSO is used here to select appropriate parameters in the RBFNN [12].

In Particle Swarm optimization, position and velocity of each particle are updated by below given formula in order to obtain optimal solution.

$$v_{i+1} = w.v_1 + c_1 rand(pbest_i - p_i) + c_2 rand(gbest_i - p_i) \quad (1)$$

$$p_{i+1} = p_i + v_{i+1} \quad (2)$$

where

$pbest_i$

: own best previous experience of the particle.
$gbest_i$
: best experience of all other particles.
p_i
: position of each particle
v_1
: velocity of the particle
w
: intertia factor
c_1
: congnitive parameter
c_2
: social parameter
$rand$
: random numbers between 0 and 1

In RDF Neural Network, there is a input layer with 10 nodes, a nonlinear hidden layer with 3 hidden nodes and a linear output layer with 10 nodes. The hidden node uses radial basis function:
$F_i = \exp(-\|x_i - c_i\|/2\sigma_i^2$
Ci is the center and σi is the width of radial basis function.

Steps:
1. Randomly generate a population of initial particles. Each particle consists of the center of radial basis function, the width of radial basis function and the weight of RBF neural network. Randomly generate a population of particles.
2. The fitness of each particle in the population is used to measure the performance of the model with current parameters of RBF neural network.
3. For each particle, its velocity is computed by Eq.(1),and its position is computed by Eq.(2).
4. The new population of particles is produced, and the new RBF neural network is gained.
5. If stopping criterion is satisfied, the procedure is stopped, then the optimal parameters are gained, and used to train RBF neural network. Otherwise, go to step 2.
6. Gain software component retrieval model based on PSO-RBF neural network.

5. LITERATURE REVIEW

Praveen Pathak[4] et.al addresses the issue of improving retrieval performance using GA to adapt various matching functions. This type of adaptation of matching functions helps in better retrieval performance than a single matching function. The vector space model is used in which documents and queries are located in a multi-dimensional vector space. Retrieval is accomplished by searching for documents that are close to the query vector.

Valerei Maxville[6] et.al proposes an AI technique to classify components based on ideal user specification. The short listed components are ranked using weighted score method (WSM).

Dimitrios G. Vogiatzis[8] et.al proposes a new methodology for intelligent classification and retrieval of software components based on user–defined requirements. GA algorithm is used for classification of components which identifies a small number of classifiers by dividing the set of available components into subsets or clusters. Every classifier is thus the representative of its subset. When a user wants to search any component, he identifies the required characteristics and then these are compared with the characteristics of the classifiers. The closest classifier matching the required characteristics over a user-defined threshold will result in the required set of components belonging to its cluster, which are presented to the user in descending matching fitness.

Chengpu Li[9] et.al proposes a semantic based approach to solve the mismatch issue between the user's query and the retrieved component to improve the component retrieval process. Rating factors for component retrieval and their calculating method are proposed. An ontology based rating factor is then used to calculate the precision degree of the retrieval result.The calculation mechanism used involves two parts : multi layered user's query description and multilayered component description.

Jasmine Sudhakarn[14] et.al proposes a new method for component classification and retrieval consisting K-nearest Neighbor (KNN) algorithm and vector space model approach.

Amandeep Bakhshi[18] et.al developed a component repository to store reusable components. Various search techniques are used to search the required component and retrieval of those components. The techniques used for searching and retrieving of components are keyword based search , signature based search and Operational semantic keyword based retrieval technique is proposed which gives best retrieval results. The results obtained from the proposed technique are compared with traditional keyword based technique.

Shweta Yadav[19] et.al proposed an algorithm which is the combination of two most popular retrieval techniques i.e. Keyword based approach and Semantics based component retrieval technique. After this the retrieved components are ranked according to the previous user demands. So, this paper proposes an efficient model for retrieving the reusable components in more optimized manner.

Shekhar Singh[20] et.al presents a meta-data model and faceted classification to store and retrieve software components using domain semantic information based on ontologies and texonomies. Most of existing

repositories retrieve a limited set of components, the proposed metadata model makes possible the recommendation of interrelated components, because of ontology and taxonomies characteristics. The results of proposed model improves the component retrieval precision.

Pankaj Vohra[23] et.al proposes a software repository is constructed which stores automatically, fragmented code on the basis of inputs and outputs. Staged technique named Auto-Detect-Fragment-Store is used for the automatically store and retrieve source-code components. Component repository is constructed which stores automatically, fragmented code on the basis of inputs and outputs.

Chintakindi Srinivas[24] et.al proposes a new approach for clustering of reusable components or documents by using hybrid XOR function which finds the degree of similarity between any two components or documents. By applying hybrid XOR function a matrix named as similarity matrix is defined which is of order n-1 where n is number of components in a given set. The proposed algorithm takes the similarity matrix as input and gives output as set of clusters of components. The approach can be justified as it carries out very simple computational logic and efficient in terms of processing with reduced search space and can be also be used in general for document clustering or software component clustering.

Ashima Singh[26] et.al proposes a software repository for storage and retrieval of software reusable components. This repository implements two types of search techniques-

Keyword based search-In this keyword is entered to repository and relevant components which match the keywords are retrieved.

User Priority-Based Component Retrieval: Select Component Type, Select Component Language, Select Component Domain and Select Component Name and the exact matching component is retrieved.

Chintakindi Srinivas[27] et.al state idea to cluster the software components and form a subsets of available components. These clusters help to select the required component with high cohesion and low coupling quickly and efficiently. A similarity function is defined to be used software component clustering and in estimating the cost of new project.The approach carried out is a feature vector based approach.

Kanwaljeet Sandhu[31] et.al proposes a suffix tree based component representation in the form of a tree model. It is a structural model for information generation and representation so that the effective component modeling and component specification will be obtained. A suffix tree is build with various levels. With each level the traversing over the keyword list is performed. Algorithms are proposed for construction of suffix tree and search in suffix tree.

Chintakindi Srinivas[30] et.al proposes an approach to cluster a given set of components by using Hybrid XNOR function as similarity function which finds the degree of similarity between any two software components or documents. A similarity matrix is obtained for a given set of documents or components by applying hybrid XNOR function. The Proposed algorithm takes similarity matrix as input and gives set of clusters as output. The output is a set of highly cohesive pattern groups or components.

Nitish Madaan[32] et.al presents the role of Requirement analysis in component selection process that helps the software developers to model their requirements. It describes the following component selection technique that helps to select the components which can satisfy the requirements: Cluster Based Selection Process: It consists of 3 stages: goal-oriented specification, dependency analysis, and cluster analysis. Goal-oriented specification uses a goal-based requirements elicitation method to specify Component based Selection (CBS) requirements.

Dr.N.Adlakha[33] et.al proposes an Ant Colony Optimization algorithm that can help reuser to identify and retrieve software component but only in a specific domain. This can be enhanced by using ceremonial methods in component representations so that components relevant to the requirements can be retrieved.

As shown in the above review, a number of researchers have been carrying out research in the area of component retrieval and clustering. However exhaustive this research , still there is a scope of optimizing the results in the field of information retrieval and repository management.

6. RESULTS AND CONCLUSION

The issue of reusability of component is generally related to the efficient storage and retrieval of reusable components from software repository. To reuse the software reusable components efficiently, they must be efficiently stored and retrieved. So, there are some component classification and retrieval techniques that can be used in order to increase the efficiency which are discussed above.

7. REFERENCES

1. M. Douglas McIlroy, J. M. Buxton, Peter Naur, and Brian Randell. 1968. "Mass-produced software components." In *Proceedings of the 1st*

International Conference on Software Engineering, Garmisch Pattenkirchen, Germany, pp. 88-98.
2. Savoy Jacques, and Daniel Desbois. 1991. "Information retrieval in hypertext systems: an approach using Bayesian networks." *Electronic publishing* 4, no. 2 pp. 87-108.
3. Brown, Alan W. *Large-scale.* 2000. *component-based development*. Vol. 1. Englewood Cliffs: Prentice Hall PTR.
4. Praveen Pathak, Michael Gordon, and Weiguo Fan. 2000. "Effective information retrieval using genetic algorithms based matching functions adaptation." In*System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pp. 1-8. IEEE.
5. Ian Sommerville. 2004. "Software Engineering. International computer science series."
6. Valerie Maxville, Jocelyn Armarego, and Chiou Peng Lam. 2004 "Intelligent component selection." In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pp. 244-249. IEEE.
7. Ioana Sora, and Doru Todinca. 2006 "Specification-based retrieval of software components through fuzzy inference." *Acta Polytechnica Hungarica* 3, no. 3 pp. 123-135.
8. Andreas S. Andreou, Dimitrios G. Vogiatzis, and George A. Papadopoulos. 2006 "Intelligent classification and retrieval of software components." In *Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International*, vol. 2, pp. 37-40. IEEE.
9. Chengpu Li, Xiaodong Liu, and Jessie Kennedy. 2008. "Semantics-Based Component Repository: Current State of Arts and a Calculation Rating Factor-based Framework." In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pp. 751-756. IEEE.
10. Joshua Zhexue Huang. 2009. "Clustering Categorical Data with k-Modes." pp. 246-250.
11. Nedhal Al Saiyd, Intisar A. Al Said, and Ahmed H. Al Takrori. 2010. "Semantic-Based Retrieving Model of Reuse Software Component." *IJCSNS* 10, no. 7, pp. 154-161.
12. Guo Su-wei. "Software component retrieval method based on PSO-RBF neural network. 2010. "In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 7, pp. V7-339-341. IEEE.
13. Shalini S Singh, and N. C. Chauhan. 2011. "K-means v/s K-medoids: A Comparative Study." In *National Conference on Recent Trends in Engineering & Technology*.
14. Jasmine Kalathipparambil Sudhakaran, and Ramaswamy Vasantha. 2011. "A mixed method approach for efficient component retrieval from a component repository." *Journal of Software Engineering and Applications*, vol. 4, pp. 442-445.
15. Ardhendu Mandal, and S. C. Pal. 2012. "Emergence of component based software engineering." *Int'l Journal of Advanced Research in Computer Science and Software Engineering* 2, no. 3, pp. 311-315.
16. Manish Verma, Mauly Srivastava, Neha Chack, Atul Kumar Diswar, and Nidhi Gupta. 2012. "A comparative study of various clustering algorithms in data mining."*International Journal of Engineering Research and Applications (IJERA) Vol* 2, pp.1379-1384.
17. Anupama Kaur, Himanshu Monga, Mnupreet Kaur, and Parvinder S. Sandhu. 2012 "Identification and performance evaluation of reusable software components based Neural Network." *International Journal of Research in Engineering and Technology* 1, *Vol* 2, pp.100-104.
18. Amandeep Bakshi. 2013. "Development of a software repository for the precise search and exact retrieval of the components." PhD diss., THAPAR UNIVERSITY PATIALA.
19. Shweta Yadav, Kamaljeet Kaur Mangat. 2013. "Design of rank based reusable component retrieval algorithm." *International journal of advanced research in computer science and software engineering, Vol* 3, pp.850-857.
20. Shekhar Singh. 2013. "An experiment in software component retrieval based on metadata and ontology repository." *International Journal of Computer Applications (0975–8887) Vol* 61, pp.33-40.
21. S.Revathi, and T. Nalini. 2013. "Performance comparison of various clustering algorithm." *International Journal of Advanced Research in Computer Science and Software Engineering Vol* 3, pp.67-72.
22. Aastha Joshi, and Rajneet Kaur. 2013 "A review: Comparative study of various clustering techniques in data mining." *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, pp.55-57.
23. Pankaj Vohra, and Ashima Singh. 2013. "Automatic Fragmentation and Storage of Code in Component Repository wrt their Input and Output Interfaces: A Tool." *International Journal of Innovative Technology and Exploring Engineering*, vol.2, pp.235-238.
24. Chintakindi Srinivas, Vangipuram Radhakrishna, and CV Guru Rao. 2013. "Clustering Software Components for Program Restructuring and Component Reuse Using Hybrid XOR Similarity Function." *AASRI Procedia*, vol.4, pp.391-328.
25. Suresh Chand Gupta and Ashok Kumar. 2013. "Reusable Software Component Retrieval System." *International Journal of Application or Innovation in Engineering and Management*, vol.2, pp.187-194.
26. Ashima Singh, Janhavi. 2014. "Development of Component repository." *International Journal of Advanced research in Computer Science and Software Engineering*, vol.4, pp.952-957.
27. Chintakindi Srinivas , Vangipuram Radhakrishna, and CV Guru Rao. 2014. "Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries." *Procedia Computer Science*, vol.31, pp.1044-1050.
28. Ajay Kumar, Lokesh Kumar Shrivastav. 2014. "Application of neural network in information retrieval." *Journal of Advanced Computing and Communication Technologies*, vol.2, pp.1-4.
29. Amita Verma, Ashwani Kumar. 2014. " Performance enhancement of k-mean clustering

algorithm for high dimensional data sets." *International Journal of Advanced Research in Computer Science and Software Engineering*, vol.4, pp.791-796.

30. Chintakindi Srinivas, Vangipuram Radhakrishna, and CV Guru Rao. 2014. "Clustering software components for program restructuring and component reuse using hybrid XNOR similarity function." *Procedia Technology*, vol.12, pp.246-254.
31. Kanwaljeet Sandhu, and Trilok Gaba. 2014. "A Novel Technique for Components Retrieval from Repositories." *An International Journal of Advanced Computer Technology*, vol.3, pp.912-920.
32. Nitish Madaan, and Jagdeep Kaur. 2014. "A Survey on Selection Techniques of Component Based Software." *An International Journal of Advanced Computer Technology*, vol.4, pp.1245-1250.
33. Vikas Verma, Munish Mehta, and Naveeta Adlakha. 2014. "Augmentation of Component Retrieval Using Ceremonial Methods.", *International Journal of Research in engineering*, vol.4, pp.26-35.
34. Anubha Jain, Swati V. Chande, and Preeti Tiwari. 2014. "Relevance of Genetic Algorithm Strategies in Query Optimization in Information Retrieval."*International Journal of Computer Science & Information Technologies*, vol.5, pp.5921-5927.