An Approach for Clustering Class Coupling Metrics to Mine Object Oriented Software Components

Anshu Parashar and Jitender Kumar Chhabra
Department of Computer Engineering, National Institute of Technology, India

Abstract: *Unsupervised learning methods such as clustering techniques are a natural choice for analyzing software quality by mining its related metrics. It is well known that clustering plays an important role in data mining tasks like in data analysis and information retrieval. In this paper, we have proposed an approach to cluster the pool of java classes based on the proximity between them. To know the proximity, coupling between each pair of classes is calculated in terms of weights using the weighted coupling measures. We modified document representations scheme as per our requirement to represent collected class coupling measures before applying k-mean clustering algorithm. In order to, reduce the dependency of k-mean clustering results efficiency on the choice of initial centroids, neighbor and link based criteria's for selecting initial k centroids have been proposed in the context of Object Oriented (OO) design artifacts i.e., classes. We demonstrate our work in detail and compare results of K-mean algorithm based on random and neighbor and link based initial centroids selection criteria's. Further the results of clustering are analyzed through purity and F-measure. It has been observed that definition of neighbor and link can be interpreted well in terms of the coupling between OO classes and produces best K-mean clustering results. Our approach of software component clustering may become an integral part of a framework to analyze and predict software quality attributes.*

Keywords: *Software engineering, OO software clustering, mining coupling metric.*

Received September 18, 2012; accepted March 20, 2014; published online June 11, 2015

1. Introduction

It is well known that clustering plays an important role in data mining tasks like in data analysis and information retrieval. Software engineers are increasingly using data mining algorithms to various software engineering tasks like defect detection, testing, debugging, maintenance etc., to improve software productivity and quality [1, 24, 33]. Coupling measure between classes reflects valuable quality information about the design of software and enables the software designer to plan various important issues of the software engineering like reusability, maintainability, change propagation, fault prediction etc., [5, 7, 8, 14, 20, 32]. Clustering can be used to identify the clusters of a software system by utilizing information about the dependencies between the different parts of the system to easily comprehend the system [24]. Each cluster constitutes a subsystem because each cluster is comprised of a small number of classes that are strongly coupled and have related functionality and is somehow independent from the other parts of the system. So, coupling measure between classes are vital based upon which the software can be clustered. The primary contributions of the paper are:

- An approach for clustering the pool of java classes has been proposed based on the proximity between them to predict the software components.
- To know the proximity, coupling between each pair of classes have been calculated in terms of weights through the weighted coupling measures. We tend to choose this measure because by this extent of coupling between classes can be measured.
- Different representations of coupling measures have been suggested to pre-process the collected coupling measures to prepare them suitable to apply k-mean clustering algorithm.
- In order to, reduce the dependency of clustering results efficiency on the choice of classes as initial centroids, Neighbor and link based criteria's for selected initial k centroids have been proposed in the context of Object Oriented (OO) design artifacts i.e., classes.
- We demonstrated our approach, analyzed and empirically validated the results in details through a case study.

The paper is organized as follow: Section 1.1 discusses the modified representations for class coupling data. Section 1.2 discusses the clustering of class coupling data. Section 2 describes the related literature. Section 3 describes the proposed approach, extraction of coupling among the classes, its representation and clustering of software components. Section 4 discusses the case study, results and empirical validations, while section 5 concludes the paper and presents the scope of future work.

1.1. Representation of Collected Class Coupling Data

In our work, we intend some modification in existing document representations to make them appropriate for representing collected class coupling measures. For this purpose, four modified representations are described in following subsections.

1.1.1. N-Dimensional Binary Weighted Scheme

On the similar lines of document clustering approach, in case of OO application say *A*, the class set of *A* is represented as ordered set of classes $Class\_Set(A)=\{C_1, C_2, C_3, ..., C_N\}$, where *N* is total number of classes in A. Each class C_i to be clustered is represented as a tuple or vector of N-dimension binary coupling vector $NBC\_V(C_i)=[x_{i1}, . .,x_{iN}]$, $i=1, ..., N$, where the dimension *N* is the same as the number of classes in the software application. The value of each x_{ij} depends on the coupling between each pair of classes. So, coupling of a class *C* is represented as $NBC\_V(C_i)$= [1, 0, 1, 1, 0]. Here, 1 at place j[th] indicates that C_i class is coupled with class C_j and 0 at place j^{th} indicates no coupling of class C_i with class C_j. After having binary coupling vectors of all *N* classes, further the coupling data for the whole application can be viewed as a *N*N* matrix of all the class vectors.

1.1.2. 2-Dimensional TF-IDF Weighted Scheme

Another well known and widely acceptable representation is Vector Space Model (VSM) and TF-IDF. Both are the basic model for document clustering [2, 33]. Similarly, for the OO application, for each class C_i, we here propose to has coupling frequency vector $CF\_V(C_i)=[x_{i1},..... x_{iN}]$. The value x_{i1} (also called as cf_{1i}) of class C_i represents import coupling usage frequency of class C_i with class C_j. Another weighting scheme Inverse Class Frequency ICF(C_i) is used to weight each class C_i based on Inverse Document Frequency (IDF). So, we can calculate $ICF(C_i)$ of each class using Equation 1:

$$ICF(C_i) = log\left(\frac{n}{ICoupF(Ci)}\right) \quad (1)$$

Where *n* is total number of classes of an application, $ICoupF(C_i)$ is number of classes using C_i. Then finally import coupling $ICoup(C_i, C_j)$ of class C_i with C_j is represented as 2D-point (cf_{ji}, $ICoupF(C_i)$* cf_{ji}).

1.1.3. Class Coupling Set Representation

The collected class coupling data for each class can be represented by class coupling Set $CC\_Set(C_i)$. For each class C_i the class coupling set contains set of classes by which C_i is coupled. Let an application A having set of classes $Class\_Set(A)=\{C_1,C_2,C_3,..., C_N\}$, here *N* is total number of classes in the application A. If class coupling set for a class C_1 is $CC\_Set(C_1)=\{C_2, C_3\}$, it means class C_1 is only coupled with classes C_2 and C_3. So, instead of considering coupling weights, we are just making the class coupling set.

1.1.4. N-Dimensional Weighted Scheme

Each class can be represented as an N-dimensional weighted vector $NWC\_V(C_i)=[x_{i1}, x_{i2}, ..., x_{iN}]$, $i=1, ..., N$, here the dimension *N* is the same as the number of classes in the software application. Value of an each weight x_{ij} is the actual direct coupling measure between each pair of classes C_i and C_j instead of having binary weight. In this way of representation, coupling between class pairs is calculated based on the extent of coupling between classes on the scale of 0 to 1. If two classes are highly coupled, then their coupling is represented by value close to 1. After having weighted coupling vectors of all *N* classes, further the coupling data for the whole application can be viewed as a *N*N* matrix of all the class vectors.

As we know, the result of clustering algorithm is highly dependent upon the proper representation of collected class coupling data. The binary weighted scheme only gives whether coupling exists or not, it does not reflect the extent of coupling between classes. The binary weights of coupling do not estimate coupling quantitatively. The TF-IDF weighted scheme is well proven format for the documents/text [2, 33]. But it is less suited due to the fact there is no sound justification for the coupling frequency and inverse coupling frequency. Both are conveying the similar information one is straight frequency of coupling other is just its normalized value. There is no suitable value in OO programming equivalent to TF and IDF. Hence, the usefulness of second representation is also not guaranteed. The third representation has some sense and for each class C_i it provides the set of classes by which C_i is coupled. But again this representation is also totally subjective as it does not give any quantitative measure of coupling. Finally, from these four forms of representations, we propose to use the *N*-dimensional weighted scheme to have an intermediate form of coupling measure. The idea behind using this method is we can do clustering based on extent of coupling between classes. Because without having quantitative coupling measure one cannot predict how important the class is for other classes and the impact of coupling on the quality of the software. In next sub-section, we describe the clustering of class coupling data.

1.2. Clustering Class Coupling Data using K-Mean

The main objective of clustering is to place points into disjoint groups called clusters. The points in a cluster are more similar to each other and dissimilar to the points of other clusters. Clustering algorithms use some distance measure to compute the distance

between two points. Among various clustering techniques available in literature, K-Means [17] clustering approach is most widely used K-Means is an unsupervised clustering technique used to classify data into K clusters. Number of clusters K must be specified before the start of clustering [27]. The efficiency of clustering depends upon the effectiveness of the selection of *k* classes as initial *k* centroids. If the initial choice of centroids is good i.e., close to the optimum, then the K-mean gives good clustering results. But there is no guarantee that K-mean algorithm will reach a global optimum. Since, different sets of initial cluster centroids can lead to different final clustering results. There are some ways like random, buckshot, fractionation and links to select initial centroids for clustering the text documents [10, 13, 21]. In this paper, we transform some initial centroid selection criteria to make it suitable to be used (with our approach) to cluster collected class coupling measures for any OO application as discussed below.

1.2.1. Random Selection of Initial K Classes as Initial Centroids

The K-mean algorithm begins by taking intermediate form of coupling data as an input and randomly chooses *k* classes to represent the centroids of the initial clusters. The classes are then assigned to the initial clusters based on the cosine distance between their coupling vectors and centroids. Once, all classes have been initially placed in clusters, the mean of clusters are recomputed to find out new clusters centroids. This process is repeated until there is no change in the cluster centroids.

1.2.2. Selection of Initial K Centroids Classes Using Neighbor and Link

Guha *et al*. [13, 21] used the concept of neighbor and link for document clustering. In our context of OO software, the neighbors of a class C_i is a set of classes of an application that are coupled to it. As we are having weighted coupling [14] between each pair of classes, so for a class C_i all classes whose coupling with C_i is greater than or equal to the threshold θ are the neighbors of class C_i. Here, θ is a user-defined threshold to decide how much coupling is required between a pair of to consider them neighbors. Depending on the application, the user can choose an appropriate value for θ. The link function gives the set of common neighbor between two classes. The information about the neighbors of every class in the data set is then represented as an $N*N$ neighbor Matrix (M) in which an entry M [i, j] is 1 or 0 depending on whether classes C_i and C_j are neighbors or not. The next section describes related works.

2. Related Works

Clustering techniques have been used to solve various type of problems in different area of science e.g., information retrieval, biology, economics, image processing [1, 20, 21, 23, 28]. Literature shows that clustering has a good potential to be used as the additional or as an alternate approach to analyze the various tasks of software engineering [9, 11, 12, 22, 31]. Clustering techniques are preferably used in document mining [21, 23, 28]. Luo *et al*. [21] used the concept of neighbors and links for clustering the text documents. They found the selection criteria based on links worked well for clustering text document using K-mean.

For OO development paradigm, class coupling is a type of dependency relationship between classes of the OO application. Coupling has been used as an important parameter effecting quality of the software. Gui *et al.* [14, 15, 16, 19] proposed coupling measures at different level of abstraction like class and package level. Gui and Scott [14] proposed a static measure of coupling to assess and rank the reusability of java components. Arisholm [4] have provided a method for identifying import coupled classes with each class at design time using UML diagrams. Gupta and Ghhabra [15] proposed coupling measures at package level. Some authors also have investigated the use of coupling measures to support impact analysis in the OO system, change ripple effect or change proneness [6]. So, existence of coupling at class level can be analyzed with the help of the clustering techniques to measure the software quality. Lot of work has been done with respect to comprehension, remodularization or partitioning large software modules [1, 3, 11, 25, 30]. In order to understand classes, Zaidman *et al*. [34] proposed static web mining and coupling metrics for early program comprehension. Fokaefs *et al*. [11] apply the agglomerative clustering algorithm to find clusters of cohesive entities and ranked according to their impact on the design of the whole system. Camelia Serban presented an approach to identify classes with high coupling using fuzzy clustering analysis [29]. Cui and Chae [9] analyzed the application of agglomerative hierarchical clustering algorithms in software reengineering. They found from their investigation that clustering algorithms have varied capabilities but it is hard to mention which clustering algorithm is perfect for component identification. Antonellis *et al.* [3] also, investigating the use of mining to support the evaluation of system maintainability. They proposed a two-steps clustering process to facilitates the assessment of a system's maintainability. They applied k-Attractor clustering algorithm on the design metrics related to quality and maintainability. As clustering [1, 17, 28] basically places the group of items in a cluster. It is very much desire to have the item in the same cluster should have strong relationship. This relationship between items may be of one of the two kinds "so far or so close", depending upon the purpose of clustering. There are some distance measures available in literature like

absolute distance, Euclidean distance, cosine similarity/distance and widely used for document clustering [18, 26].

3. Proposed Approach

In this paper, we mainly propose to use the concept of neighbor and link (Guha *et al*. [13, 21] used this concept for document clustering) for clustering the classes of OO application. In order to, have clusters of interrelated set of classes we required some criteria by which we can measure the proximity between classes. We exploit coupling between the classes in order to measure proximity between classes of OO application in terms of neighbor and link. Here, we redefine the concept of neighbor and link in terms of class coupling. Two classes are said to be neighbors of each other if they are having coupling up to a specific extent. Which means that neighborhood of classes is defined in terms of extent of coupling between them. In our work, we define a class C_i to be neighbor of class C_j if and only if the coupling between them is greater than a specified threshold θ as described by Equation 2:

$$Neighborhood(C_i, C_j) = \begin{cases} if \quad coupling(C_i, C_j) \geq \theta & 1 \\ otherwise & 0 \end{cases} \quad (2)$$

Using above equation, we represent the neighborhood information of classes by making a *N***N* neighbour matrix *M*. Here, *N* is the number of classes in underlying application. Each *M* [*i*, *j*] is set to 1 or 0 depending upon their neighborhood values. Two classes C_i and C_j are neighbours if the degree of coupling between them is more than the coupling threshold θ. The degree of coupling is given by the coupling weights [14]. Coupled classes will not be considered as neighbors, if the coupling between them is less than the coupling threshold. We also consider that two classes are close or related if they are having some common neighbors. The link function measures this characteristic of the proximity between classes. For our approach, we define link function as:

Link(C_i,C_j)={ *set of common neighbor classes between* C_i *and* C_j }

|*link*(C_i,C_j)|=*m where m is the number of common neighbor classes between* C_i *and* C_j

So, the *link*(C_i, C_j) function gives the set of common neighbor classes between C_i and C_j and |*link*(C_i, C_j)| gives count of common neighbor classes between C_i and C_j. In our approach, we intend to use the concept of link and through this suggest two important points. Firstly, when classes C_i and C_j are having some common neighbors but not coupled to each other then also they should be considered to be close enough. Secondly, as the value of *link*(C_i, C_j) is large, the probability of these two classes to be in same cluster also increases. As it is a prerequisite to collect class coupling data and to represent it in a suitable format before applying clustering algorithm. So, firstly coupling measures are collected as proposed by Gui and Scott [14]. Secondly, the collected measures are represented as N-dimensional weighted coupling vectors described in section 1. Thirdly, for clustering using K-mean, a set of *k* classes is chosen as initial centroids using selection criteria described in section 1. Finally, K-mean algorithm is applied. Our main focus in this approach is on neighbor and link based K-mean clustering method in order to cluster classes of OO application. The detailed approach with an example is described in following sub-sections.

3.1. Collection of Weighted Class Coupling Data and its Representation as N-Dimensional Weighted Class Coupling Vector

In our approach a weighted measure of the total direct coupling between classes proposed by Gui and Scott [14] has been used. It is static coupling measure that estimates the proportion of a class say C_i's functionality that is obtained from say class C_j and more suitable to cluster underling coupling metric data set of a java application. Any Java application can be regarded as a directed graph in which the vertices correspond to its classes and the edges correspond to direct coupling between classes as shown in Figure 1 of an example application A.

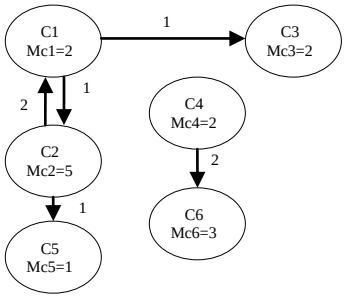

Figure 1. Class coupling graph of an example application.

Consider application A comprising a set of *N* classes, *Class_Set(A)*={C_1, C_2, ..., C_6}. The weighted direct coupling *WCD*(*i*, *j*) [14] between class C_i and C_j is represented as shown in Equation 3:

$$WCD(Ci,Cj) = \frac{|Xi,j|}{|Xi|+|Mi|} \quad (3)$$

Where M_j be the set of members of class C_j and $X_{i,j}$ be the set of members of class C_j invoked by class C_i. The set of all members of other classes invoked by C_i is $|X_i|$. It indicates the extent to which class C_i depends upon other classes for its functionality. An edge exists from C_i to C_j if and only if $X_{i,j}$ is not null, Figure 2 shows the coupling between classes in terms of WCD weights on their edges.

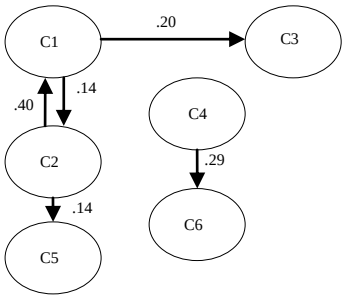

Figure 2. Directed coupling graph with WCD(i, j) weights of example application.

3.1.1. N-Dimensional Weighted Class Coupling Vectors

As it has been discussed in section 1 the collected coupling measures are represented as N-dimensional weighted class coupling vectors. Here, each class C_i is treated as a vector having N attributes in order of class C_1 to C_N. Here, N is total number of classes. For every class C_i the weights have been assigned to each pair of C_i and C_1 to C_N as per weighted direct coupling between them. For a sample application A, vectors of all classes are shown in Table 1. Each row *i* represents a weighted coupling vector of class C_i.

Table 1. N-dimensional weighted coupling matrix.

Classes	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	.40	.20	0	0	0
C_2	.14	1	0	0	.14	0
C_3	0	0	1	0	0	0
C_4	0	0	0	1	0	.29
C_5	0	0	0	0	1	0
C_6	0	0	0	0	0	1

3.2. Selection of K-Initial Centroids Using Neighbour and Link and its Application in K-Mean

A good candidate for initial centroids should be not only close enough to a certain group of classes but should also be well separated from other centroids. As it has been discussed earlier in section 1, our target is to select initial *k* clusters through neighbour and link for further application of K-mean algorithm to form clusters. This selection procedure comprises three basic steps as described below.

3.2.1. Creation of the Neighbour Matrix

The neighbours of a class in the class set are those classes that are directly coupled to it. Classes C_i and C_j are considered as neighbours if the $WCD(C_i, C_j) \geq \theta$, where threshold θ is the minimum coupling required to be neighbours. So, $N*N$ neighbour matrix M as shown in Table 2 has been created from the weighed coupling matrix by considering θ=0.20 and $M[i, j]$ marked as 1 if class C_i and C_j are neighbors. As coupling threshold θ increases the number of classes under clustering reduced. It is required to have low value of θ to have rich clusters that cover maximum number of classes of an application.

Table 2. Neighbor matrix.

Classes	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	1	1	0	0	0
C_2	0	1	0	0	0	0
C_3	0	0	1	0	0	0
C_4	0	0	0	1	0	1
C_5	0	0	0	0	1	0
C_6	0	0	0	0	0	1

3.2.2. Ranking of Classes Using Cosine and Link Functions

The value of the link function [21] $link(C_i, C_j)$ is defined as the number of common neighbor classes between C_i and C_j and it can be obtained by multiplying the i^{th} row of the neighbor matrix M with its j^{th} column using Equation 4:

$$link(Ci,Cj) = \sum_{m=1}^{n} M[i,m] * M[m,j] \quad (4)$$

Thus, if $link(C_i, C_j)$ is large, then it is more probable that C_i and C_j are close enough to be in the same cluster. So firstly, by checking the neighbor matrix of the data set, we list the classes in descending order of their number of neighbours i.e., C_1, C_4, C_2, C_3, C_5, C_6. The top m classes are selected from this list. A set of initial centroid candidates is created as S_m with $m=k+nplus$, where k is the desired number of clusters and n_{plus} is the extra number of candidates selected. Through n_{plus}, we can have more candidates that can take part in the selection of initial centroids. When k=3 and $nplus$=1, S_m has four classes $S_m=\{C_1, C_4, C_2, C_3\}$. Next, cosine and link values between every pair of classes in S_m is obtained, and then each class pairs are ranked in ascending order of their cosine and link values, respectively. For a pair of Classes C_i and C_j, let's define $rankcos(C_i, C_j)$ be its rank based on the cosine value, $Ranklink(C_i, C_j)$ be its rank based on the link value and $rank(C_i, C_j)$ be the sum of $rankcos(C_i, C_j)$ and $ranklink(C_i, C_j)$ [23]. In our case, a smaller value of $rank(C_i, C_j)$ represents a higher rank, it means 0 corresponds to the highest rank. This rank represents the dissimilarity of a class from other classes of the application. It means a low rank value represents that the class is likely to be at a significant distance from given set of classes. The calculated ranks of each pair of S_m classes of an application A is shown in Table 3.

Table 3. Similarity measurement and rank between initial centroid candidates.

C_i, C_j	Cos_sim	Rankcos	Link	Ranklink	Rank(C_i, C_j)
C_1, C_4	0	0	1	1	1
C_1, C_2	.12	1	2	2	3
C_1, C_3	0	0	2	2	2
C_4, C_2	0	0	0	0	0
C_4, C_3	0	0	0	0	0
C_2, C_3	0	0	0	0	0

However, when the clusters centroids are not well separated, partitioning them on the basis of only pairwise similarity is not good enough because some classes in different clusters may be similar to each other [21]. To avoid this problem, the concept of

neighbors and link has been applied for clustering the classes based on their coupling patterns, introduced and used in [13] for document clustering. So, similarity measure for the family of k-means algorithms by combining the cosine and link functions [21] can be represented as Equation 5. Where $0 \leq \alpha \leq 1$ and *Lmax* is the largest possible value of $link(C_i, C_j)$ and α is the coefficient set by the user. When α is set to 0, the similarity measure becomes the cosine function and it becomes the link function when α is 1. α in the range of [0.8, 0.95] produces the best clustering results [21]. To calculate $link(C_i, {}^kC_j)$ for class C_i and the cluster centroid ${}^kC_j$, k columns have been added to the neighbor matrix M. The new matrix is an $n*(n+k)$ matrix M_c, in which an entry $M[n+j]$ is 1 or 0 depending on whether a document C_i and a centroid ${}^kC_j$ are neighbors or not. The value of $link(C_i, {}^kC_j)$ can be obtained by multiplying the i^{th} row of M' with its $n+j^{th}$ column using this formula as shown in Equation 6

$$f(C_i, {}^kC_j) = \alpha * \frac{link(C_i, {}^kC_j)}{Lmax} + (1-\alpha) * cos(C_i, {}^kC_j) \quad (5)$$

$$link(C_i, {}^kC_j) = \sum_{m=1}^{n} M'[i, m] * M'[m, n+j] \quad (6)$$

3.2.3. Finding Best K Initial Centroids

Initial centroids are well separated from each other in order to represent the whole data set. Thus, the class pairs with high ranks could be considered as good initial centroid candidates. For the selection of k initial centroids out of m candidates, there are ${}^kC_2$ possible combinations. Each combination is a k-subset of S_m and we calculate the rank value of each combination (comk) using Equation 7:

$$Rankcomk = \sum rank(Ci, Cj) \quad (7)$$

That means, the rank value of a combination is the sum of the rank values of the ${}^kC_2$ pairs of initial centroid candidate classes in the combination. In our application A, there are 4 combinations available and their rank values are shown in Table 4.

Table 4. Combination of centroid candidates and their rank.

Comk	${}^kC_2$ Pairs of Centroid Candidates	Rankcomk
{C1,C2 ,C3}	{ C1,C2 },{ C1,C3}, {C2 ,C3}	5
{C1,C2 ,C4}	{ C1,C2 },{ C1,C4}, {C2 ,C4}	4
{C1,C3 ,C4}	{ C1,C3 },{ C1,C4}, {C3 ,C4}	3
{C2,C3,C4}	{ C2,C3 },{ C2,C4}, {C3 ,C4}	0

Then, we choose the combination with the highest rank (i.e., the smallest rank value) as the set of initial centroids for the k-means algorithm. As rank of {C_2, C_3, C_4} is the smallest among all combinations. So, classes in these combinations are considered as well separated from each other.

3.2.4. Similarity Measure based on the Cosine and Link Functions

Cosine and link measures are used for the similarity between the coupling patterns of two classes. Pair wise similarity is considered to determine whether a class is assigned to a cluster or not.

3.3. K-Mean Clustering Using both Initial Centroids Selection Criteria

3.3.1. Neighbor and Link Selection Criterion

So, as per section 3.2 for an application A, initial centroids using neighbors and links are ${}^kC_1$={1 .40 .20 0 0 0}, ${}^kC_2$={.14 1 0 0 .14 0}, ${}^kC_3$={0 0 0 1 0 .29}. Then, k-mean has been applied that used the similarity measure based on cosine and link and finally we have three clusters k_1={C_1, C_2, C_5}, k_2={C_3}, k_3={C_4, C_6}.

3.3.2. Random Selection Criterion

The K-mean algorithm begins by taking intermediate form of coupling data as an input and randomly chooses k classes to represent the centroids of the initial clusters. The classes are then placed to different K clusters based the cosine distance between their coupling vectors and centroids. Once all records have been initially placed in clusters, the mean of clusters are recomputed to find out new cluster centroids. This process is repeated as per the K-mean algorithm. For application A, suppose classes C_1, C_3, C_5 are randomly selected as initial centroids kc_1={1 .40 .20 0 0 0}, kc_2={0 0 1 0 0 0}, kc_3={0 0 0 0 1 0}. Then, k-mean has been applied that used the similarity measure based on cosine and produced k_1={C_1, C_2, C_4, C_6}, k_2={C_3}, k_3={C_5} three final clusters.

4. A Case Study and Evaluation of its Results

To evaluate our approach, a case study has been performed on the classes of different packages of java (JDK). Aim is to identify the components by clustering the classes according to their dependency characteristics based on the assigned coupling weights. As per our approach (described in section 3), we chose to represent dependency among classes using coupling weights in the range of 0 to 1 to show the extent of coupling instead of having binary weights. The steps for the case study and evaluation of results are described in following sub-sections.

4.1. Collection of Data and Computation of Metrics

For the case study, we have used four java packages (Applet, IO, Awt, Lang) and from these four packages 27 classes have been selected to find out the coupling between them through WCD metric proposed by Gui and Scott [14] Next, to have the extent of coupling between pair of classes based on Equation 1. The collected coupling weights between each class C_i with all n classes are represented by n-dimensional weighted class coupling representation as described in section 1.

4.2. Clustering Class Coupling Data

As we want to apply k-mean document clustering approach on the collected class coupling data, so after getting the class coupling vectors each class coupling vector is treated as document vector in VSM. Then, as per our methodology initial k-clusters have been chosen through both selection approaches i.e., random selection and selection using neighbor and link as described in Table 5. Here, we opted to have four clusters because our target to cluster coupling data set of four java packages, so value of *k* is four. Following Table 5 shows the selected k (*k*=4) initial centroids.

Then, finally k-mean clustering has been applied separately by using both types of initial *k* centroids selection criteria's. Tables 6 and 7 are shows the formed clusters.

Table 5. Initial Centroids selected through both selection criteria.

Selection Criteria	Initial k Centroids	Process Adopted for Selection
Random	C2, C7, C8, C12	Randomly selected(as described in sections 1 and 3)
Using Neighbor and Link	C1, C6, C7, C8	Centroids are selected by ranking based on cosine similarity and link (as described in section 1 and 3)

Table 6. Clusters formed based on neighbour and link selection of initial centroids.

Cluster No.	Classes Included in Cluster(Using Neighbour and Link Selection of Initial Centroids)
1	C3-Awteventmulticaster, C4-cardlayout, C5-checkbox, C6-ckeckboxmenuitem, C11-container, C18-component
2	C19-menuitem, C20- menucomponent, C25-inputstream
3	C1-Applet, C2- Alphacomposite, C7-Bufferedinputstream , C10-chararrayreader , C12-objectinputstream, C13-System, C14-string, C17-object , C21- bufferedoutputstream , C22-filterinputstream, C23-filteroutputstream, C24- reader ,C26 -package, C27-number
4	C8-bufferreader, C9-Bytearrayinputstream , C15-math , C16-panel

Table 7. Clusters formed based on random selection of initial centroids.

Cluster No.	Classes Included in Clusters (Using Random Selection of Initial Centroids))
1	C8-bufferreader
2	C2-Alphacomposite, C6- ckeckboxmenuitem, C7- Bufferedinputstream, C9-Bytearrayinputstream , C10-chararrayreader, C19-menuitem,C20-menucomponent, C21-bufferedoutputstream , C22- filterinputstream, C23-filteroutputstream, C24-reader, C25-inputstream, C26-package, C27- number
3	C12-objectinputstream
4	C1-Applet, C3-Awteventmulticaster, C4-cardlayout, C5-checkbox, C11-container C13-System, C14-string, C15-math , C16-panel, C17-object C18-component

4.3. Evaluation of Clustering Results

F-measure and purity values are used to evaluate the accuracy of our clustering algorithm for both initial k -centroids selection criteria. The F-measure is a harmonic combination of the precision and recall values used in information retrieval [21]. So, each cluster obtained can be considered as the result of a query, whereas each pre-classified component of classes can be considered as the desired set of classes for that query. Thus, we can calculate the precision $P(i, j)$ and recall $R(i, j)$ of each resultant cluster j for each pre-classified component i using formula described in [21]. The corresponding F-measure $F(i, j)$, F-measure of the whole clustering result, purity of a cluster and purity of the whole clustering result [21] are also, calculated. In general, the larger the F-measure is the better the clustering result. The larger the purity value is, the better the clustering result is. Tables 8 and 9, show the evaluation of results of k-mean for both types of initial centroids selection criteria's.

Table 8. Precision and recall values for both selection criteria's.

Selection Criteria for Initial Centroids(k=4)								
(i, j)	Random				Neighbour and Link			
	n_{ij}	P_R (i, j)	R_R (i, j)	F_R (i ,j)	n_{ij}	P_{NandL}(i, j)	R_{NandL}(i, j)	F_{NandL} (i, j)
(1, 1)	1	.05	1	.09	0	0	0	0
(1, 2)	0	0	0	0	0	0	0	0
(1, 3)	0	0	0	0	1	.07	1	.13
(1, 4)	0	0	0	0	0	0	0	0
(2, 1)	9	.47	.9	.62	6	1	.6	.75
(2, 2)	1	.2	.1	.13	2	.67	.2	.28
(2, 3)	0	0	0	0	1	.07	.1	.08
(2, 4)	0	0	0	0	1	.25	.1	.14
(3, 1)	5	.26	.5	.34	0	0	0	0
(3, 2)	3	.6	.3	.4	1	.33	.1	.15
(3, 3)	1	.5	.1	.17	7	.5	.7	.58
(3, 4)	1	1	.1	.18	2	.5	.2	.29
(4, 1)	4	.21	.67	.32	0	0	0	0
(4, 2)	0	0	0	0	0	0	0	0
(4, 3)	1	.5	.17	.25	5	.36	.36	.5
(4, 4)	0	0	0	0	1	.25	.25	.2

Table 9. Total purity and F-measure value of clustering.

Selection Criteria for Initial K Centroids(k=4)			
Random		Using Neighbour and Link	
Purity	**F-measure**	**Purity**	**F-measure**
.50	.43	.55	.60

Further, Table 10 summarizes purity comparisons among all four clusters and pre-classified components for both types of initial centroids selection criteria's. Here, we found that F-measure value for the neighbour and link ($F_{N\ and\ L}$) goes up to .60 and F_R for random selection goes up to.43. Further, Figures 3, 4 and 5 show the comparisons of precision, recall and F-measures of both type of selection criteria's. From experimental results, it has been found that the F-measure and purity values of clustering results using neighbours and link (*N* and *L*) selection method are more near to the actual components. It means that selection of initial centroids by using neighbors and link provided better clusters than random selection in terms of clustering accuracy.

Table 10. Purity of clusters.

Selection Criteria for Initial k Centroids (k=4)							
Random				Using Neighbour and Link			
Purity(1)	**Purity(2)**	**Purity(3)**	**Purity(4)**	**Purity(1)**	**Purity(2)**	**Purity(3)**	**Purity(4)**
.47	.6	.5	1	1	.67	.5	.5

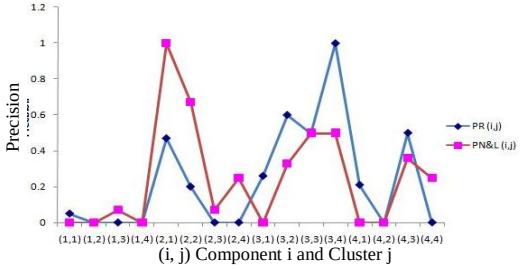

Figure 3. Comparison of precision values for both initial centroids selections.

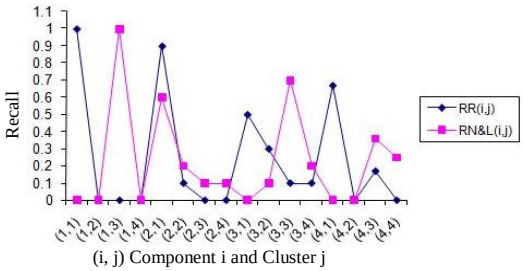

Figure 4. Comparison of recall values for both initial centroids selction criteria's.

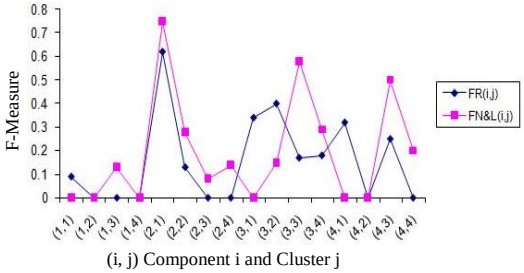

Figure 5. Comparison of F-measure values for both initial centroids selections.

It is required to mention that some time random selection may provide good results but its probability is totally dependent upon the effectiveness of the classes selected as initial centroids. As far as the neighbor and link criterion is concerned it surely provides best and well separated initial centroids. So, from all this, we can say that the concept of neighbor and link of text document clustering is well suited for clustering OO software i.e., to cluster the classes based on their coupling. The reason behind this is, there is no criterion behind the random selection. It is just a hit and trial method. Selection of the initial centroids classes using predefined criterion i.e., neighbors and links produced better clusters. The proposed approach give more precise clusters of classes and software experts can further utilize these clusters for the analysis of software quality.

5. Conclusions and Future Work

As strong believers of the important role of clustering to measure the quality of OO applications, we recognize mainly there are two ways where clustering can play its role. First is to measure any particular software quality attribute, one can use the clustering approach to cluster the various artifacts produced during the different phases of the software development life cycle. Secondly, to have a more authenticated quality measure and to do the more rigours analysis of quality, one can perform clustering on the software metrics data computed for any particular quality attributes. Presently, our focus is on second aspect and our target was to explore the usage of document clustering approach to cluster the classes of OO application based on their dependency on each other. Certainly coupling between classes is a significant property of OO paradigm. Coupling affects software quality in many ways and hence can be used as a predictor of software quality attributes such as reusability changeability, ripple effects of changes and fault-proneness. In our approach, clustering has been applied only on the collected static coupling measure, although coupling can be measured dynamically as well. But as per our idea, we considered clusters as the components of the system because more or less each cluster have those set of classes that are more related to each other to fulfil the nearly similar type of responsibility. So for this reason static coupling measures are more useful because they cover the complete set of classes of the system. Our approach mainly has three aspects. Firstly as discussed in introduction part, there may be different schemes to represent the coupling measures by using different ways primarily used in document representation for clustering. Among those, we chose and found N-dimensional weighted scheme more suitable. Secondly, we opt to apply k-mean document clustering approach to form clusters of classes on the bases of their relevance to each other measured through their coupling. It is a fact that the correctness of clusters formed by k-mean is largely dependent on the proper selection of initial *k* classes as initial *k* centroids. From results, we found neighbor and link based selection of initial centroids helps in better clustering results for documents. Finally, to evaluate our approach, we compare its results with results of k-mean that used random selection. As shown in results, the concept of neighbor and link is found suitable and works well in our context.

The key findings and usefulness of the clustering approach are:

- The Neighbor and link measure works well in our context. The definition of neighbor and link can be interpreted in terms of the coupling between classes.
- Classes are ranked and then *k* classes with higher ranks are selected as initial centroids as these classes are far apart from each other. So, it gives better clustering results.
- Some time it is difficult to maintain the large software systems because of inherent coupling between their components i.e., classes, so having clusters of strongly coupled classes, maintainer can predict error and change propagations easily which ultimately helps in changing/maintaining the classes.
- The emphasis of our assumption is that the classes which are contributing towards the similar functionality should be kept in same cluster. So, for legacy systems where design artifacts are not available, the maintainer can extract the key-components of the software system by exploring these clusters.
- Clustering software metrics i.e., coupling reflects the important aspects of a system concerning its quality. It can help to verify the design and easily comprehend the software system.

- Our clustering approach may become an integral part of a framework to analyze and predict software quality through mining some facts about quality of the software from collected software metrics.

Hence, clustering class coupling metrics is helpful to mine OO software components. In future, we will explore other suitable document clustering approaches that can be suitable to cluster the classes based on their coupling measures. Further, we will develop a frame for mining different software quality parameters like reusability changeability, ripple effects of changes and fault-proneness etc.

References

[1] Abreu F., Pereira G., and Sousa P., "A Coupling-Guided Cluster Analysis Approach to Reengineer the Modularity of Object-Oriented Systems," *in Proceedings of the Conference on Software Maintenance and Reengineering*, pp. 13-22, 2000.

[2] Alzghool M. and Inkpen D., "Clustering the Topics using TF-IDF for Model Fusion," *in Proceedings of the 2nd PhD Workshop on Information and Knowledge Management*, pp. 97-100, 2008.

[3] Antonellis P., Antoniou D., Kanellopoulos Y., Makris C., Theodoridis E., Tjortjis C., and Tsirakis N., "Clustering for Monitoring Software Systems Maintainability Evolution," *Electronic Notes in Theoretical Computer Science*, vol. 233, pp. 43-57, 2009.

[4] Arisholm E., "Dynamic Coupling Measurement for Object-Oriented Software," *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 491-506, 2004.

[5] Basili V., Briand L., and Melo W., "A Validation of Object-Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751-761, 1996.

[6] Briand L., Daly J., and Wust J., "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, 1999.

[7] Briand L., Wust J., and Louinis H., "Using Coupling Measurement for Impact Analysis in Object-Oriented Systems," *in Proceedings of IEEE International Conference on Software Maintenance*, Oxford, pp. 475-482, 1999.

[8] Chidamber S., Darcy D., and Kemerer C., "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 629-637, 1998.

[9] Cui J. and Chae H., "Applying Agglomerative Hierarchical Clustering Algorithms to Component Identification for Legacy Systems," *Information and Software Technology*, vol. 53, no. 6, pp. 601-614, 2011.

[10] Cutting D., Karger D., Pedersen J., and Tukey J., "A Cluster-Based Approach to Browsing Large Document Collections," *in Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 318-329, 1992.

[11] Fokaefs M., Tsantalis N., and Chatzigeorgiou A., "Decomposing Object-Oriented Class Modules using an Agglomerative Clustering Technique," *in Proceedings of IEEE International Conference on Software Maintenance*, Edmonton, pp. 93-101, 2009.

[12] Glorie M., Zaidman A., Hofland I., and Deursen A., "Splitting a Large Software Archive for Easing Future Software Evolution- An Industrial Experience Report using Formal Concept Analysis," *in Proceedings of the 12th European Conference on Software Maintenance and Reengineering*, Athens, pp. 153-162, 2008.

[13] Guha S., Rastogi R., and Shim K., "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Information Systems*, vol. 25, no. 5, pp. 345-366, 2000.

[14] Gui G. and Scott P., "Ranking Reusability of Software Components using Coupling Metrics," *Journal of Systems and Software*, vol. 80, no. 9, pp. 1450-1459, 2007.

[15] Gupta V. and Chhabra J., "Measurement of Dynamic Metrics using Dynamic Analysis of Programs," *in Proceedings of the WSEAS International Conference on Applied Computing Conference*, pp. 81-86, 2008.

[16] Henry S. and Lattanzi M., "Measurement of Software Maintainability and Reusability in the Object Oriented Paradigm," *Technical Report*, Computer Science, Virginia Polytechnic Institute and State University. 1994.

[17] Kaufman L. and Rousseeuw P., *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley and Sons, 1990.

[18] Korenius T., Laurikkala J., and Juhola M., "On Principal Component Analysis, Cosine and Euclidean Measures in Information Retrieval," *Information Science*, vol. 177, no. 22, pp. 4893-4905, 2007.

[19] Li W. and Henry S., "Object Oriented Metrics that Predict Maintainability," *Journal of Systems and Software*, vol. 23, no. 2, pp. 111-122, 1993.

[20] Liu B., Hsu W., and Ma Y., "Integrating Classification and Association Rule Mining," *in Proceedings of American Association for Artificial Intelligence International Conference on Knowledge Discovery and Data Mining*, pp. 1-7, 1998.

[21] Luo C., Li Y., and Chung S., "Text Document Clustering Based on Neighbors," *Data and*

Knowledge Engineering, vol. 68, no. 11, pp. 1271-1288, 2009.

[22] Maqbool O. and Babri H., "Hierarchical Clustering for Software Architecture Recovery," *IEEE Transactions on Software Engineering*, vol. 33, no. 11, 759-780, 2007.

[23] Meskine F. and Bahloul S., "Privacy Preserving K-means Clustering: A Survey Research" *the International Arab Journal of Information Technology*, vol. 9, no. 2, pp. 194-200, 2012.

[24] Mishra S., Kushwaha D., and Misra A., "Creating Reusable Software Component from Object-Oriented Legacy System through Reverse Engineering," *Journal of Object Technology*, vol. 8, no. 5, pp. 133-152, 2009.

[25] Praditwong K., Harman M., and Yao X., "Software Module Clustering as a Multi-Objective Search Problem," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 264-282, 2011.

[26] Qian G., Sural S., Gu Y., and Pramanik S., "Similarity between Euclidean and Cosine Angle Distance for Nearest Neighbor Queries," *in Proceedings of the ACM Symposium on Applied Computing*, pp. 1232-1237, 2004.

[27] Rao I., "Data Mining and Clustering Techniques," *in Proceedings of the DRTC Workshop on Semantic Web*, Bangalore, pp. 1-11, 2003.

[28] Romesburg H., *Cluster Analysis for Researchers*, Lulu Press North Carolina, 2004.

[29] Serban C., "High Coupling Detection using Fuzzy Clustering Analysis," *in Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques*, pp. 223-226, 2009.

[30] Simon F., Steinbruckner F., and Lewrentz C., "Metrics Based Refactoring," *in Proceedings of the 5th European Conference on Software Maintenance and Reengineering*, Lisbon, pp. 30-38, 2011.

[31] Wiggerts A., "Using Clustering Algorithms in Legacy Systems Remodularization," *in Proceedings of the 4th Working Conference on Reverse Engineering*, Amsterdam, pp. 33-43, 1997.

[32] Wilkie F. and Kitchenham B., "Coupling Measures and Change Ripples in C++ Application Software," *Journal of System Software*, vol. 52, no. 2-3, pp.157-164, 2000.

[33] Xia Y., "A Survey of Document Clustering Techniques and Comparison of LDA and moVMF," available at: http://cs229.stanford.edu/proj2010/Xiao-A%20Survey%20of%20Document%20Clustering%20Techniques%20&%20Comparison%20of%20LDA%20and%20moVMF.pdf, last visited 2010.

[34] Zaidman A., Bois B., and Demeyer S., "How Webmining and Coupling Metrics Improve Early Program Comprehension," *in Proceedings of the 14th IEEE International Conference on Program Comprehension*, Athens, pp. 74-78, 2006.

Anshu Parashar pursuing his PhD degree from National Institute of Technology, India. He did BTech (2002) and MTech (2008) in Computer Science and Engineering. He is working as Associate Professor in Department of Computer Science and Engineering in HCTM, India. He has published more than 25 papers in various International, National Conferences and Journals. He has more than 11 years of teaching experience. His area of interest includes software engineering, data mining and object-oriented systems.

Jitender Chhabra Professor, Department of Computer Engineering and HOD, Department of Computer Applications, National Institute of Technology. He did both his BTech and MTech in Computer Engineering from Regional Engineering College Kurukshetra (now National Institute of Technology) as Gold Medalist. He did his PhD in Software Metrics from Delhi. He has published more than 90 papers in various International and National Conferences and Journals including journals of IEEE, ACM, Springer and Elsevier. He has more than 20 years of teaching and research experience. He is author of three books from McGraw Hill including the one Schaum Series International book titled "Programming with C". He is Reviewer of IEEE Transactions, Elsevier, Springer, Wiley and many other Journals. He has worked in collaboration with multinational IT companies HP and TCS in the area of Software Engineering. His area of interest includes software engineering, data mining, soft computing and object-oriented systems.