Clustering Dynamic Class Coupling Data using K-Mean and Cosine Similarity Measure to Predict Class Reusability Pattern

Jitender Kumar Chhabra and Anshu Parashar

Department of Computer Engineering, National Institute of Technology, Kurukshetra, 136 119, India
E-mail: conf.ppr@gmail.com

Abstract
Data clustering techniques can be applied to cluster set of software components having similar characteristics like dependence to the same set of components and coupling with each other etc. This paper is an attempt to identify clusters of classes having dependence amongst each other and observe a common coupling pattern existing in the same repository. We explore both document clustering technique based on tf-idf weighing and cosine similarity measure to cluster classes from the collection of class coupling data for particular java application. For this purpose firstly dynamic analysis of java application is done using UML diagrams to collect class import coupling data. Then in second step, this coupling data of each class is represented in N-dimensional vector space. Further class coupling frequency and inverse class frequency is calculated using tf and idf. Then finally in the third step basic K-mean clustering and cosine similarity techniques are applied to find clusters of classes. Further each cluster is ranked for its goodness based on some user specified criteria and threshold. The proposed approach has been applied on simple Java application and our study indicates that by browsing such clusters a developer can discover class’s reusability patterns and behaviour.

Keywords: Coupling, Data Clustering, Software Reusability.

Introduction
Identification of reusable components during the process of software development is an essential activity it helps to develop, identify and store reusable components [4]. Software Reuse is defined as the process of building or assembling software applications from previously developed software [23] and to increase productivity, quality, maintainability etc [20,5]. Object oriented development [15] offers many features above the traditional development approaches to improve software reusability through encapsulation and inheritance [17]. In object-oriented paradigm, coupling between classes is well-recognized structural attribute and plays a vital role in measuring the reusability. One can define a class C_a related to class C_b if C_a must use C_b in all future reuse. These highly coupled groups of classes should be reused together for ensuring the proper functioning of the application [8, 24]. Hence it is always desirable to find out the classes along with their associated classes [15]. So to discover the clusters of classes that should be reused in combination, clustering approach can be used. By using clustering, one can find frequently used classes in the same cluster and can know their coupling behaviour in a particular application.

Data Clustering & Reusability
Data mining is the process of extracting new and useful knowledge from large amount of data. Mining is widely used to solve many business problems such as customer behavior modeling, product recommendation, fraud detection etc [27]. Data mining techniques can be used to analyze software engineering data to better understand the software and assist software engineering tasks. Clustering plays an important role in mining task like in data analysis and information retrieval [12]. Clustering means to clusters, where a cluster is a collection of objects, which are similar and closer to each other and dissimilar/distant objects belong to different clusters. Clustering can be applied to cluster the documents in the process of information retrieval. The Vector Space Model (VSM) is the basic model for document clustering. In this model, each document, d_j, can be represented as a term-frequency vector in the term-space: 

\[ d_j = (tf_{i1}, tf_{i2}, ..., tf_{iv}) \quad j=1,2,...,D \]

where \( tf_{ij} \) is the frequency of the ith term in document \( d_j \), \( V \) is the total number of the selected vocabulary, and \( D \) is the total number of documents in the collection [25]. One can weight each term based on its Inverse Document Frequency (idf) [25,2]. After having VSM representation, K-mean algorithm can be applied to cluster the documents. Basic K-means Algorithm for finding K clusters involves following steps [22]:

1. Select K points as the initial centroids.
2. Assign points to their closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids don’t change.

Clustering technique can be applied to cluster the Classes/components that may often be reused in combinations [27]. Due to the popularity of open source concept large amount of source code of classes is available on internet as software repositories. Some also exists in large software companies where developer in one group may reuse classes written by other groups. For this reason, it is desirable to have clustering mechanism that forms clusters of classes based on their association or coupling patterns/behaviour. By searching for class patterns with high probability of repetitions.
Related Works
For object-oriented development paradigm, class coupling has been used as an important parameter effecting reusability. Efforts have been made by the researchers to measure reusability through coupling and cohesion of components [18].

ISA [19] methodology has been proposed to identify data cohesive subsystems. Gui et al [10] proposed a new static measure of coupling to assess and rank the reusability of java components. Arisholm et al[3] have provided a method for identifying import coupled classes with each class at design time using UML diagrams. Data mining is focused on developing efficient techniques to extract relevant information from very large volumes of data that may be exploited, for example, in decision making, to improve software reliability and productivity [26]. Cluster formation from large databases is an important data mining task. Few algorithms like CLARANS [21], BIRCH [29] has been proposed for clustering large data sets. Li et al [16] devised a set-theoretic clustering method called PCS (Pair-wise Consensus Scheme) for high-dimensional data. Abrantesy et al [1] described a method for the segmentation of dynamic data. Document clustering has been an interesting topic of study since a long time. Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [13] is one of the best algorithms for agglomerative clustering [30]. K-Mean's and its family of algorithms have been extensively used in document clustering [14]. Kiran et al [14] proposed a hierarchical clustering algorithm using closed frequent item sets that use Wikipedia as an external knowledge to enhance the document representation. Fung et al [9] proposed to use the notion of frequent itemsets and applied TF-IDF to define the score function Score (Ci ← docj) for measuring the goodness of a cluster Ci for a document docj. Alzghool et al [2] also proposed a technique based on clustering the training topics according to their tf-idf (term frequency-inverse document frequency) properties. Some researchers have been using clustering in software development tasks. Czibula et al [7] introduced a search based approach for identifying instances of design patterns in a software system. For Evaluation of cluster quality authors in [28] proposed cluster ranking to quickly single out the most significant clusters. Also to measure goodness and quality of cluster Entropy, F-measure has been used [22]. There are some distance measures available in literature like Absolute distance, Euclidean distance and cosine similarity/distance [31,32,33,6]. We find the basic K-mean algorithm and cosine similarity measure very simple and go well with our initial idea.

Proposed Methodology
In our approach we propose to cluster class import coupling data for a particular java application. To collect class import coupling data dynamic analysis of java application is done using UML. Then collected import coupling data of each class is represented using N- VSM and tf-idf. Then basic K-mean clustering technique and cosine similarity measures are applied to find cluster of classes. Our approach consists of three steps:

1. Collection of Class import coupling data through UML.
2. Representation of Collected Data.
3. Clustering of class import coupling data

The steps are described in section 3.1 to 3.3.

Collection of Class Import Coupling Data through UML
Dynamic analysis of a program is one way of finding the coupling between classes. During this step, the existing application is analyzed through UML diagrams [11] as described by Erik Arisholm [3] in order to extract import coupling of its classes. They used following formula for calculating class import coupling IC_OC (Ci) to measure dependency of one class to other classes.

\[ IC_{OC}(c) = \{(m1, c_1, c): \forall (o, c) \in R_w, \exists (o, c) \in R_e, \forall N)c \neq c_1 \land (o, m) | o, m \in ME \} \]

IC_OC (Ci) counts the number of distinct classes that a method in a given object uses.

Representation of Collected Data using N-dimensional import coupling vector and tf-idf weighing scheme
Data collected in step 1 is then represented as N-dimensional import coupling vector and 2-D import coupling vector using tf-idf weighing scheme. For an application A, the class set of A is represented as Class_Set(A)=\{C1,C2,C3,...,Cn\} where n is total number of classes in an application A. Each class Ci is represented as N-dimensional import coupling vector NIC_V(Ci) where N is the total number of classes in an application A. For each class Ci this vector is represented as

\[ NIC_V(C)_{ij} = x_{ij}, x_{j2}, ..., x_{jn} \]

The value xi (also called as cfij) of class Ci represents import coupling usage frequency of class Ci in class Cj. Next inverse class frequency (ICF) weighing is applied using idf formula \(1:\)
ICF(Ci) = log \left( \frac{n}{ICoupF(Ci)} \right) \quad (1)

Where \( n \) is total number of classes, \( ICoupF(Ci) \) 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 \( (cf_{ij}, ICoupF(C_i) \cdot cf_{ij}) \).

### Clustering of class import coupling data

#### Clustering using K-Mean approach

In this step clustering technique is to be applied on 2D representation of each pair of classes \( ICoup(C_i, C_j) \). Each cluster will have set of classes that can be reused together. So, K-mean algorithm as described in section 1.1 is used to find out such clusters. Here K is the pre-assumed required number of clusters and value of K is decided by the user. Once we decide on what are the K clusters and their initial centroids then K-mean algorithm starts as per section 1.1. The absolute distance function (formula 2) is used to measure the closeness.

\[
d_{d}(x,y) = \sqrt{x_{1}^2 + x_{2}^2 + \ldots + x_{n}^2}
\]

In every iteration centroids are recalculated. Each class pair \( (C_i, C_j) \) is assigned to the cluster with the nearest centroid point. By iterating K-mean algorithm (until there is no movement of points), it will discover clusters of the form e.g \( \{C_1, C_2, C_3\} \). Their union \( \{C_1, C_2, C_3\} \) will form the final cluster. One can interpret this as classes in a cluster are coupled with each other and will be reused together.

After having K clusters each cluster is ranked by taking average and sum of its \( x, y \) points, e.g. Cluster I = \{(a,b), (c,d)\}. then \( \text{RankC(Cluster I)} = \frac{(a+c/2)+(b+d/2)}{2} \). This \( \text{RankC(k)} \) should pass the threshold \( \text{th}_k \) specified by the user. Threshold is the lowest possible permissible rank that will be used to classify cluster as good or bad. If \( \text{RankC(k)} < \text{th}_k \) then cluster \( k \) is discarded otherwise \( k \) is retained.

### Clustering Using Cosine Similarity

The N-Dimensional Class vector representation \( NIC_V(C_i) \) of classes is used to compute cosine similarity between classes on a scale of \([0, 1]\). The Cosine Similarity \([7]\) of two vectors \( NIC_V(C_i) \) & \( NIC_V(C_j) \) is defined as:

\[
\text{Cos_Sim}(C_i, C_j) = \frac{C_i \cdot C_j}{||C_i|| ||C_j||} \quad (3)
\]

Where \( C_i, C_j = C_i[0]*C_j[0] + C_i[1]*C_j[1] \ldots \) and \( ||C_i|| = \sqrt{C_i[0]^2 + C_i[1]^2 + \ldots} \)

The \( \text{Cos_Sim}(C_i, C_j) = 1 \) can be interpreted as \( C_i & C_j \) are coupled with exactly same set of classes. \( \text{Cos_Sim}(C_i, C_j) = 0 \) can be interpreted as coupling set of \( C_i & C_j \) do not have any common class. So for each k-cluster we have to decide its permissible similarity scale \( \text{Sim_Scale} \). If any class pair \( \text{Cos_Sim}(C_i, C_j) \) satisfy this \( \text{Sim_Scale} \), then \( (C_i, C_j) \) is included in cluster \( k \).

This similarity scale \( \text{Sim_Scale} \) of each cluster also reveals its rank. So after computing \( \text{Cos_Sim}(C_i, C_j) \) for each pair of classes we place that pair in nearby cluster as per its cosine similarity value. In next section, we demonstrate our methodology of clustering class import coupling data using K-Mean and Cosine Similarity approaches.

### Example

We are using a small example to illustrate our approach for clustering of class import coupling data to have K-clusters of classes. Let application \( A \) having classes Class_Set(A) = \( \{ C_1, C_2, C_3, C_4 \} \). As per the first step we assume to have import coupling data collected for application \( A \) using UML approach. The next sections 4.1 & 4.2 show the second and third step of our approach.

### Representation of Collected Data

In second step the collected coupling data is represented as N-dimensional Class Import Coupling Vector \( NIC_V(C_i) \) of all classes in an application A as shown in table 1. Then inverse Class Frequency (ICF) of each class is calculated using formula 1 and represented in table 2.

#### Table 1. N-dimensional Class import coupling Vector.

<table>
<thead>
<tr>
<th>NIC_V(Ci)</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC_V(C1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NIC_V(C2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NIC_V(C3)</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NIC_V(C4)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Table 2. ICF value of each class

<table>
<thead>
<tr>
<th>Class</th>
<th>ICF(Ci)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>.30</td>
</tr>
<tr>
<td>C2</td>
<td>.60</td>
</tr>
<tr>
<td>C3</td>
<td>0</td>
</tr>
<tr>
<td>C4</td>
<td>.60</td>
</tr>
</tbody>
</table>

#### Table 3. Point representation and K-mean clustering iterations

<table>
<thead>
<tr>
<th>ICoup(Ci, Cj)</th>
<th>Point</th>
<th>Iteration I</th>
<th>Cluster</th>
<th>Iteration II</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dist 1</td>
<td>Dist 2</td>
<td>Dist 3</td>
<td>Dist 1</td>
</tr>
<tr>
<td>(C1, C2)</td>
<td>0</td>
<td>1.60 7.8</td>
<td>0</td>
<td>1.45 6.3</td>
<td>I</td>
</tr>
<tr>
<td>(C1, C3)</td>
<td>3</td>
<td>1.8</td>
<td>3.8</td>
<td>3.2</td>
<td>3.0</td>
</tr>
<tr>
<td>(C1, C4)</td>
<td>6</td>
<td>1.8</td>
<td>7.8</td>
<td>6.2</td>
<td>0</td>
</tr>
<tr>
<td>(C2, C3)</td>
<td>1</td>
<td>.60</td>
<td>1.6</td>
<td>0</td>
<td>6.2</td>
</tr>
<tr>
<td>(C2, C4)</td>
<td>0</td>
<td>0</td>
<td>1.60</td>
<td>7.8</td>
<td>0</td>
</tr>
<tr>
<td>(C3, C4)</td>
<td>1</td>
<td>.30</td>
<td>1.3</td>
<td>.30</td>
<td>6.5</td>
</tr>
</tbody>
</table>
Table 6. NIV\_V(C\_i) and Cluster formation using Cos\_Sim(C\_i,C\_j)

<table>
<thead>
<tr>
<th>NIC_V(C_i)</th>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>Cos_Sim(C_i,C_j)</th>
<th>Sim_Scale</th>
<th>Assigned Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC_V(C_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Cos_Sim(C_1,C_2)</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>NIC_V(C_2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Cos_Sim(C_2,C_3)</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>NIC_V(C_3)</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Cos_Sim(C_3,C_4)</td>
<td>0</td>
<td>I</td>
</tr>
<tr>
<td>NIC_V(C_4)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Cos_Sim(C_4,C_3)</td>
<td>.98</td>
<td>III</td>
</tr>
</tbody>
</table>

After this import coupling ICoup(C\_i,C\_j) between all classes is represent as 2D-points (c\_ij, ICF(C\_j)\_* c\_ij) as shown in table 3.

**Clustering of Class Import Coupling Data Using K-Mean Approach**

In this step K-mean algorithm is applied on the import coupling data represented as points in the table 3. We assume to have 3 clusters as output of K-mean algorithm (K=3) and th\_c = 3. The initial cluster-means (centroids) of three clusters are (0, 0), (1,60) and (6,1.8), chosen randomly. Next, in Iteration I the distance of all the points to each of the three centroids are calculated by using the distance function: \( \text{dist}(a, b) = |x2 - x1| + |y2 - y1| \) where \( a=(x1, y1) \) and \( b=(x2, y2) \). Then each point (C\_i,C\_j) is placed in its nearest distance cluster. Then, in Iteration II re-compute the new cluster centers (centroids) by taking the mean of the points in each cluster. So new centroids become (0, 0), (1,45) and (4,5,1.8) for cluster I, II and III respectively. Then again the distance from all points to each three centroids are calculated and each point (C\_i,C\_j) assigned to its nearest cluster. After iteration II we found that no point movement is there i.e. all points remain in their previously assigned clusters. It means, these are the final three clusters as shown in table 3. After obtaining three clusters I, II and III, their ranks are calculated and compared with th\_c =3 as shown in below table 4.

**Cosine Similarity Approach**

The N-Dimensional import coupling vector (table 1) is used to find coupling behaviour of all classes. For each K cluster we decide permissible Sim\_Scale. Clusters I will have class pairs having Sim\_Scale < .20, cluster II will have class pairs having Sim\_Scale < .60 and cluster III will be have class pairs having Sim\_Scale < .80 as shown in table 5.

**Results & Discussion**

After applying the clustering step using K-Mean as in section 4 we obtain these clusters I, II and III as shown bellow in figure 1.

![Final Clusters Formation using K-mean](image1)

**Table 4. Rank of Clusters formed by K-mean and status**

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Sim_Scale</th>
<th>Cluster Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>&lt; .30</td>
<td>Class pair having least import coupling similarity</td>
</tr>
<tr>
<td>II</td>
<td>&lt; .60</td>
<td>Classes pair having some import coupling similarity</td>
</tr>
<tr>
<td>III</td>
<td>&lt; .80</td>
<td>Classes pair having significant import coupling similarity</td>
</tr>
</tbody>
</table>

**Table 5. Cluster ranks based on Sim\_Scale**

<table>
<thead>
<tr>
<th>Cluster</th>
<th>RankC(k)</th>
<th>Comparison with th_c=3</th>
<th>Cluster status</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>&lt; th_c</td>
<td>Discarded</td>
</tr>
<tr>
<td>II</td>
<td>1.45</td>
<td>&lt; th_c</td>
<td>Discarded</td>
</tr>
<tr>
<td>III</td>
<td>6.5</td>
<td>&gt; th_c</td>
<td>Retained</td>
</tr>
</tbody>
</table>

![Clusters formed by K-mean and their Rank](image2)

Then Cos\_Sim(C\_i,C\_j) between two classes are measured by cosine similarity formula (formula 3) and pair (C\_i,C\_j) is assigned to appropriate cluster according to table 5. The
For an application \( A \) coupling set of classes \( C_1, C_2 \) and \( C_3 \) are not having any class pairs having \( \text{Sim}_{\text{Scale}} < 0.80 \) as shown in table 5. Table 4 and Figure 2 show the rank of these clusters and it has been observed that cluster I and cluster II are not satisfying the required threshold \( \text{th}_{C}=3 \). Only cluster III is retained. So the union of cluster III \( \{C_1, C_3, C_4\} \) will form the final cluster. We interpret this as classes in a cluster are coupled with each other and will be reused together. Ranking of cluster III suggests that these classes are highly used in application \( A \) and mostly used together.

Further cluster formed by cosine similarity approach in section 4.2.2 are shown below:

- **Cluster I**: \( \{ (C_1,C_2), (C_1,C_3), (C_1,C_4), (C_2,C_3), (C_2,C_4) \} \)
- **Cluster II**: \( \{ \} \)
- **Cluster III**: \( \{(C_3, C_4)\} \)

For an application \( A \) cluster I suggests that import coupling set of classes \( C_1, C_2 \) and \( C_3 \) are not having any common class. Their import coupling behavior is entirely different. Cluster III suggested that class \( C_3 \) and \( C_4 \) have some common classes in their import coupling set and their import coupling behavior is significantly similar. So developer can measure the coupling behavior of classes to predict its reusability pattern by browsing these clusters.

**Conclusions**

In this paper, an attempt has been made to determine class reusability pattern and behavior from dynamically collected class import coupling data of java application. We have explored the idea of document clustering (using tf-idf weighting scheme) and N-dimensional Vector space to represent the coupling between two classes. Our initial study indicates that basic technique of K-mean clustering can be constructive to place of most frequently reusable classes together in a same cluster. It means classes in a cluster are coupled with each other and will be reused together. Further cluster formation using cosine similarity measure is also helpful to know which classes have similar/different coupling behaviour and also to know whether classes are coupled with some common classes or not. So reuse issues like deciding what group of classes should be incorporated into repository and identifying exact set of classes to reuse, can be addressed through these clustering mechanism. Currently, we have applied our approach on a simple example. However the approach can also be applied on larger java applications. Moreover, other mining and clustering algorithms can be applied our approach on a simple example. However the approach can also be applied on larger java applications.

**References**

17. Michail, A. “Data Mining Library Reuse Patterns in User-Selected Applications”, In 14th IEEE
International Conference on Automated Software Engineering, pp. 24--33, 1999


[21] Ng,R.T., Han, J., “Efficient and effective clustering methods for spatial data mining”. In Proceeding of VLDB conference, pp. 144-155,1994


