Abstract—Software reuse is the process of implementing or updating software systems using existing software assets. Anything that is produced from a software development effort can potentially be reused. In this study, the performance of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is evaluated for Reusability Prediction of Function based Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the six linguistic values. Five Input metrics are used as Input and clusters are formed using DBSCAN, thereafter 10 fold cross validation performance of the system is recorded. The proposed technique is showing Accuracy value approximately equal to 86.238%, so it is satisfactory enough to use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique for the prediction of the function based reusable modules from the existing reservoir of software components.

Keywords— DBSCAN, Reuse, Precision, Recall.

I. INTRODUCTION

THIS Software reuse is the process of implementing or updating software systems using existing software assets [1]. Software assets or components include all software products, from requirements and proposals, to specifications and designs, to user manuals and test suites. Anything that is produced from a software development effort can potentially be reused.

A great deal of research over the past several years has been devoted to the development of methodologies to create reusable software components and component libraries, where there is an additional cost involved to create a reusable component from scratch. That additional cost could be avoided by identifying and extracting reusable components from the already developed large inventory of existing systems. But the issue of how to identify good reusable components from existing systems has remained relatively unexplored. Our approach, for identification and evaluation of reusable software, is based on software models and metrics. As the exact relationship between the attributes of the reusability is difficult to establish so a Clustering based approach could serve as an economical, automatic tool to generate reusability ranking of software by formulating the relationship based on its training. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is one of the most common clustering algorithms and also most cited in scientific literature because of the following advantages [6]:

1. DBSCAN does not require you to know the number of clusters in the data a priori
2. DBSCAN can find arbitrarily shaped clusters. It can even find clusters completely surrounded by (but not connected to) a different cluster.
3. DBSCAN has a notion of noise.
4. DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database.

Hence, in our study we will experiment with Density-Based Spatial Clustering of Applications with Noise algorithm for the reusability prediction.

II. METHODOLOGY

Reusability evaluation System for function Based Software Components can be framed using following steps:

A) Selection and refinement of metrics targeting the quality of function based software system and perform parsing of the software system to generate the Meta information related to that Software. The metric of [2-5] are used and the metrics are as under:

The proposed five metrics for function Oriented Paradigm are as follows [2-5]:

i) Cyclometric Complexity Using Mc Cabe’s Measure
ii) Halstead Software Science Indicator
iii) Regularity Metric
iv) Reuse-Frequency Metric
v) Coupling Metric

B) Calculate the metric values of the sampled software components.

C) Use DBSCAN (Density-Based Spatial Clustering of Applications with Noise) based prediction system for the Reusability Prediction:

Jagdeep Kaur Saini, Amitabh Sharma, Dr. Parvinder S. Sandhu
DBSCAN's definition of a cluster is based on the notion of density-reachability. Basically, a point \( q \) is directly density-reachable from a point \( p \) if it is not farther away than a given distance \( \varepsilon \) (i.e., is part of its \( \varepsilon \)-neighborhood), and if \( p \) is surrounded by sufficiently many points such that one may consider \( p \) and \( q \) be part of a cluster. \( q \) is called density-reachable from \( p \) if there is a sequence of points with and \( p_1 = p \) and \( p_n = q \) where each \( p_{i+1} \) is directly density-reachable from \( p_i \). Note that the relation of density-reachable is not symmetric (since \( q \) might lie on the edge of a cluster, having insufficiently many neighbors to count as a genuine cluster element), so the notion of density-connected is introduced: two points \( p \) and \( q \) are density-connected if there is a point \( o \) such that \( o \) and \( p \) as well as \( o \) and \( q \) are density-reachable [6].

A cluster, which is a subset of the points of the database, satisfies two properties:

1. All points within the cluster are mutually density-connected.
2. If a point is density-connected to any point of the cluster, it is part of the cluster as well.

DBSCAN requires two parameters: \( \varepsilon \) (\( \varepsilon \)-values) and the minimum number of points required to form a cluster (\( \text{minPts} \)). It starts with an arbitrary starting point that has not been visited. This point's \( \varepsilon \)-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized \( \varepsilon \)-environment of a different point and hence be made part of a cluster.

If a point is found to be part of a cluster, its \( \varepsilon \)-neighborhood is also part of that cluster. Hence, all points that are found within the \( \varepsilon \)-neighborhood are added, as is their own \( \varepsilon \)-neighborhood. This process continues until the cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

The Pseudocode of the algorithm is as follows [6]:

DBSCAN(D, \( \varepsilon \), \( \text{minPts} \))

\[ C = 0 \]

for each unvisited point \( P \) in dataset D

mark \( P \) as visited

\[ N = \text{getNeighbors}(P, \varepsilon) \]

if sizeof(\( N \)) < \( \text{minPts} \)

mark \( P \) as NOISE

else

\[ C = \text{next cluster} \]

expandCluster(\( P \), \( N \), \( C \), \( \varepsilon \), \( \text{minPts} \))

expandCluster(\( P \), \( N \), \( C \), \( \varepsilon \), \( \text{minPts} \))

add \( P \) to cluster \( C \)

for each point \( P' \) in \( N \)

if \( P' \) is not visited

mark \( P' \) as visited

\[ N' = \text{getNeighbors}(P', \varepsilon) \]

if sizeof(\( N' \)) >= \( \text{minPts} \)

\( N = N \) joined with \( N' \)

if \( P' \) is not yet member of any cluster

add \( P' \) to cluster \( C \)

The cross validation performed to determine the number of clusters is done in the following steps:

1. the number of clusters is set to 1
2. the training set is split randomly into 10 folds.
3. EM is performed 10 times using the 10 folds the usual CV way.
4. the loglikelihood is averaged over all 10 results.
5. if loglikelihood has increased the number of clusters is increased by 1 and the program continues at step 2.

The number of folds is fixed to 10, as long as the number of instances in the training set is not smaller than 10. If this is the case the number of folds is set equal to the number of instances.

Deduce the results on the 10 fold cross validation accuracy, precision and recall values.

In case of the two-cluster based problem, the confusion matrix has four categories: True positives (TP) are modules correctly classified as Reusable modules. False positives (FP) refer to non-reusable modules incorrectly labeled as reusable modules. True negatives (TN) correspond to non-reusable modules correctly classified as such. Finally, false negatives (FN) refer to reusable modules incorrectly classified as non-reusable modules as shown in Table I.

<table>
<thead>
<tr>
<th>Predicted Value</th>
<th>Real Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reusable</td>
</tr>
<tr>
<td>Reusable</td>
<td>TP</td>
</tr>
<tr>
<td>Non-Reusable</td>
<td>FN</td>
</tr>
</tbody>
</table>

With help of the confusion matrix values the precision and recall values are calculated described below:

- **Precision**

  The Precision is the proportion of the examples which truly have class \( x \) among all those which were classified as class \( x \). The technique having maximum value of probability of detection and lower value of probability of false alarms is chosen as the best reusability prediction technique.

  Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the positive class). The equation is:

  \[
  \text{Precision} = \frac{TP}{TP + FP}
  \]

- **Recall**

  Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been) [8]. The
Recall can be calculated as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

- **Accuracy**

The percentage of the predicted values that match with the expected values of the reusability for the given data.

The best system is that having the high Accuracy, High Precision and High Recall value.

### III. RESULTS & DISCUSSION

The function oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of six numeric labels i.e. 1, 2, 3, 4, 5 and 6. The label 1 represents Nil and the label 6 represents the Excellent Reusability Label. The statistics of the count of the number of examples of certain reusability label is shown in the Table II. The Graphical representation of the count of the number of examples of certain reusability label is shown in the Figure 1.

#### TABLE II

**STATISTICS OF THE REUSABILITY OUTPUT ATTRIBUTE IN THE DATASET**

<table>
<thead>
<tr>
<th>Selected attribute</th>
<th>Name: Reusability Missing: 0 (0%)</th>
<th>Distinct: 6</th>
<th>Type: Nominal Unique: 0 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Label</td>
<td></td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

The statistics shows that in the dataset, there are 9 examples of label 1, 10 examples of label 2, 26 examples of label 3, 29 examples of label 4, 17 examples of label 5 and 18 examples of label 6.

The input attributes are expressed in the three linguistic labels i.e. 1, 2, and 3. The label 1 corresponds to the Low value, label 2 corresponds to the Medium value and label 3 corresponds to the high value.

The given data with five Input Attributes i.e. Coupling, Volume, Complexity, Regularity, Reuse Frequency, and Output attributes is loaded in the Weka environment. First, the DBSCAN clustering ignores Reusability output attribute.

The following are the parameters used in the DBSCAN algorithm implementation in the WEKA as shown in figure 2:

- **database_Type** – It is the used database name and path
- **database_distanceType** – It tells us the type of distance used. It is set to Euclidian Distance.
- **epsilon** – It is radius of the epsilon-range-queries. It is set to 0.9 value.
- **minPoints** – This parameter tells the minimum number of DataObjects required in an epsilon-range-query. It is set to 6 value.

The DBSCAN clustering algorithm has created clusters numbered as 0 to 3 and assigned the 15 (means 26%) examples to cluster number 0, 13 (means 23%) examples to cluster number 1, 10 (means 18%) examples to cluster number 2 and 19 (means 33%) examples to cluster number 3. It mean that the algorithm has clustered only 57 instances and remaining 52 instances are detected as noise and these 52 instances are unclustered instances. Further the cluster numbers are again assigned. Predicted Labels as follows:

#### TABLE III

**THE ASSIGNMENT OF PREDICTED LABELS TO THE CLUSTERS FORMED BY DBSCAN**

<table>
<thead>
<tr>
<th>Cluster Number</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 0</td>
<td>4</td>
</tr>
<tr>
<td>Cluster 1</td>
<td>3</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>5</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>6</td>
</tr>
</tbody>
</table>

The confusion matrix calculated is shown in Table IV.

#### TABLE IV

**THE CONFUSION MATRIX GENERATED AFTER APPLYING DBSCAN**

<table>
<thead>
<tr>
<th>Predicted Label of Reusability</th>
<th>Real Reusability Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 2 Snapshot of the Parameters Set for DBSCAN

Fig. 1 Bar-chart of Count of examples of the Reusability Output Attribute in the Dataset
The Precision and Recall values for different the Reusability levels if the reusability is shown in table V and VI respectively.

### TABLE V
**Precision Value of Different Classes of the Reusability Values**

<table>
<thead>
<tr>
<th>Reusability Level</th>
<th>Precision Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.307</td>
</tr>
<tr>
<td>4</td>
<td>0.867</td>
</tr>
<tr>
<td>5</td>
<td>0.9</td>
</tr>
<tr>
<td>6</td>
<td>0.84</td>
</tr>
</tbody>
</table>

### TABLE VI
**Recall Value of Different Classes of the Reusability Values**

<table>
<thead>
<tr>
<th>Reusability Level</th>
<th>Recall Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
</tr>
<tr>
<td>4</td>
<td>0.65</td>
</tr>
<tr>
<td>5</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

As evidenced from the confusion matrix the incorrectly clustered instances are 15 means 13.7615% is the inaccurate percentage value and Accuracy of prediction is 86.238%.

### IV. Conclusion

In this study, the performance of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is evaluated for Reusability Prediction of Function based Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the six linguistic values. Five Input metrics are used as Input and clusters are formed using DBSCAN, thereafter 10 fold cross validation performance of the system is recorded. As deduced from the results it is clear that Precision value of the sixth level is satisfactory and Recall value of the sixth level reusability class is the maximum, it means the system is able to detect the “Excellent” components precisely. But the Precision and Recall values of the Level-1 and Level-2 is zero, it means that the system is not detecting the system with poor reusability but able to identify the ‘good’ as well as ‘excellent’ reusable components. Precision value of the level-5 reusability class is the second best; it means the system is able to detect the “Good” components with good precision.

The proposed technique is showing Accuracy value approximately equal to 86.238%, so it is satisfactory enough to use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) technique for the prediction of the function based reusable modules from the existing reservoir of software components.

### REFERENCES