A Hierarchical Clustering Approach for Modeling of Reusability of Object Oriented Software Components

Deepak Kumar, Gaurav Raj, Dr. Parvinder S. Sandhu

Abstract—Software Reusability modeling is helpful in evaluating the quality of developed or developing reusable software components and in identification of reusable components from existing legacy systems; that can save cost of developing the software from scratch. But the issue of how to identify reusable components from existing systems has remained relatively unexplored. In this research work, structural attributes of software components are represented quantifiably with help of software metrics and performance of Hierarchical Clustering based Approach is investigated to identify the reusable Object Oriented software systems. It is found that the performance of Hierarchical clustering based approach is satisfactory enough for the identification of the object based reusable modules from the existing reservoir of software components. So, the developed system can be used to enhance the productivity and quality of software development.

Keywords—Accuracy, Hierarchical Clustering, Software Reusability, Software Metric.

I. INTRODUCTION

The demand for new software applications is currently increasing at the exponential rate, as is the cost to develop them. The numbers of qualified and experienced professionals required for this extra work are not increasing commensurably [1]. Software professionals have recognized reuse as a powerful means of potentially overcoming the above said software crisis [2]-[3] and it promises significant improvements in software productivity and quality [4]-[5]. There are two approaches for reuse of code: develop the reusable code from scratch or identify and extract the reusable code from already developed code. The organizations that has experience in developing software, but not yet used the software re-use concept, there exists extra cost to develop the reusable components from scratch to build and strengthen their reusable software reservoir [4]. The cost of developing the software from scratch can be saved by identifying and extracting the reusable components from already developed and existing software systems or legacy systems [6]. But the issue of how to identify reusable components from existing systems has remained relatively unexplored. In both the cases, whether we are developing software from scratch or reusing code from already developed projects, there is a need of evaluating the quality of the potentially reusable piece of software.

The aim of Metrics is to predict the quality of the software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. The requirement today is to relate the reusability attributes with the metrics and to find how these metrics collectively determine the reusability of the software component. To achieve both the quality and productivity objectives it is always recommended to go for the software reuse that not only saves the time taken to develop the product from scratch but also delivers the almost error free code, as the code is already tested many times during its earlier reuse.

Tracz observed that for programmers to reuse software they must first find it useful [7]. Experimental results confirm that prediction of reusability is possible but it involves more than the set of metrics that are being used [8]. According to Poulin [9], in some sense, researchers have fully explored most traditional methods of measuring reusability: complexity, module size, interface characteristics, etc., but the ability to reuse software also depends on domain characteristics. It means we should concentrate on evaluating the software in terms of its relevancy to a particular domain.

The contribution of metrics to the overall objective of the software quality is understood and recognized [10]-[12]. But how these metrics collectively determine reusability of a software component is still at its naïve stage although a number of attempts are made in [15]-[22].

With the objective of taking advantage of the features of the hierarchical clustering, in this study Hierarchical clustering based approach is used to economically determining reusability of software components in existing systems as well as the reusable components that are in the design phase. Inputs to the system, are provided in form of five object oriented metric values as representation of the attributes of the software component and output is be obtained in terms of evaluation in terms of ‘Reusable’ or ‘Non-Reusable’ component.
The paper is organized as: the second section explains the steps undertaken to achieve the problem solution. Third section of this paper illustrates the results taken and discussions on the results are made. Finally, the conclusions are written in the last section and future scope is mentioned.

II. PROPOSED METHODOLOGY

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

A. Selection of Metrics

Selection of metrics targeting the quality of Object Oriented software system are spotted and parsing of the software system is performed to generate the Meta information related to that Software [23]-[25]. The metrics that are used in [23]-[25] are further used in this study and the metrics are as under:

   a. Weighted Methods per Class (WMC)
   b. Depth of Inheritance Tree (DIT)
   c. Number of Children (NOC)
   d. Coupling Between Object Classes (CBO)
   e. Lack of Cohesion in Methods (LCOM)

B. Perform Clustering

There are two main methods of hierarchical clustering algorithm.

First method is agglomerative approach, where we start from the bottom where all the objects are and going up (bottom up approach) through merging of objects. We begin with each individual objects and merge the two closest objects. The process is iterated until all objects are aggregated into a single group.

Second method is divisive approach (top down approach), where we start with assumption that all objects are group into a single group and then we split the group into two recursively until each group consists of a single object. One possible way to perform divisive approach is to first form a minimum spanning tree (e.g using Kruskal algorithm) and then recursively (or iteratively) split the tree by the largest distance.

Step by step algorithm of agglomerative approach to compute hierarchical clustering is as follow [20]:

1. Convert object features to distance matrix.
2. Set each object as a cluster (thus if we have 6 objects, we will have 6 clusters in the beginning)
3. Iterate until number of cluster is 1
   a. Merge two closest clusters
   b. Update distance matrix

The flow chart of agglomerative hierarchical clustering algorithm is given below:

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

C. Comparison Criteria

The comparisons are made on the basis of the least value of 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.

\[
\begin{array}{c|c|c}
\text{Predicted Value} & \text{Real Data Value} & \\
\hline
\text{Reusable} & TP & FP \\
\text{Non-Reusable} & FN & TN \\
\end{array}
\]

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 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 class) [8]. 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:

\[
\text{Recall} = \frac{TP}{(TP + FN)}
\]  

(2)

• **Accuracy**

The Accuracy is 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.

**D. Conclusions Drawn**

The conclusions are made on the basis of the comparison made in the previous section.

**III. RESULTS AND DISCUSSIONS**

The proposed methodology is implemented in MATLAB 7.4. MATLAB (Matrix Laboratory) environment is one such facility which lends a high performance language for technical computing.

The Object oriented dataset considered have the output attribute as Reusability value. The Reusability in the dataset is expressed in terms of two numeric labels i.e. ‘1’ and ‘2’. The label ‘1’ represents Non-Reusable and the label ‘2’ represents the Reusable 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 2.

**TABLE II**

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

<table>
<thead>
<tr>
<th>Number of Instances</th>
<th>Category of the Reusability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Class-1</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
</tbody>
</table>

The statistics shows that in the dataset, there are 32 examples of label ‘1’ and 55 examples of label ‘2’. The input attribute-wise statistical details of the count of the examples of the labels are shown in Table III to Table VII. 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 values.

**TABLE III**

**STATISTICS OF THE INPUT ATTRIBUTE TWMC IN THE DATASET**

<table>
<thead>
<tr>
<th>Selected attribute</th>
<th>Name: TWMC</th>
<th>Missing: 0 (0%)</th>
<th>Distinct: 3</th>
<th>Type: Nominal Unique: 0 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Label</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE IV**

**STATISTICS OF THE INPUT ATTRIBUTE LTDIT IN THE DATASET**

<table>
<thead>
<tr>
<th>Selected attribute</th>
<th>Name: LTDIT</th>
<th>Missing: 0 (0%)</th>
<th>Distinct: 3</th>
<th>Type: Nominal Unique: 0 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Label</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>38</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE V**

**STATISTICS OF THE INPUT ATTRIBUTE LTNOC IN THE DATASET**

<table>
<thead>
<tr>
<th>Selected attribute</th>
<th>Name: LTNOC</th>
<th>Missing: 0 (0%)</th>
<th>Distinct: 3</th>
<th>Type: Nominal Unique: 0 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Label</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>39</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE VI**

**STATISTICS OF THE INPUT ATTRIBUTE LCBO IN THE DATASET**

<table>
<thead>
<tr>
<th>Selected attribute</th>
<th>Name: LCBO</th>
<th>Missing: 0 (0%)</th>
<th>Distinct: 3</th>
<th>Type: Nominal Unique: 0 (0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Label</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The given data with five Input Attributes is applied to the hierarchical clustering algorithm and dendrogram is plotted as shown in figure 3. The dendrogram shows Indices of the components on the x-axis and distance between the components on the y-axis.

Fig. 3 Dendrogram Showing the Hierarchical Clustering of the Components

Thereafter, the clusters are constructed from linkages with maximum two clusters. With help of the actual and the calculated values of the components the confusion matrix is formed as shown in table VIII.

As Precision is the proportion of the examples which truly have class x among all those which were classified as class x. It means the Precision of the Reusable components i.e. Precision_{Reuse} is equal to 41/48 = 0.8541 and the Precision of the Non-Reusable components i.e. Precision_{Non-Reuse} is equal to 25/32= 0.781

It means the Recall of the Reusable components i.e. Recall_{Reuse} is equal to 41/55= 0.74 and the Recall of the Non-Reusable components i.e. Recall_{Non-Reuse} is equal to 25/32= 0.781

The Accuracy percentage of the proposed system is 75.8621%.

IV. CONCLUSION AND FUTURE SCOPE

In this study, Hierarchical Clustering approach is evaluated for Reusability Prediction of Object Oriented Software systems. Here, the metric based approach is used for prediction. Reusability value is expressed in the two linguistic values. Five Input metrics are used as Input and clusters are formed using Hierarchical Clustering, thereafter performance of the system is recorded. As deduced from the results it is clear that Precision value of the Reusable class is the more than Non-reusable Class, it means the system is able to detect the “Reusable” components precisely. Recall value of the Non-reusable class is the better than Reusable Class. The proposed technique is showing Accuracy value approximately equal to 75.8621%, so it is satisfactory enough to use the Hierarchical clustering based technique for the identification of the object based reusable modules from the existing reservoir of software components.

The proposed approach is applied on the C++ based software modules/components and it can further be extended to the Artificial Intelligence (AI) based software components e.g. Prolog Language based software components. It can also be tried to calculate the fault-tolerance of the software components with help of the proposed metric framework.

REFERENCES


