Identification of Reusable Procedure Based Modules using kNN Approach

Amritpal Kaur, Rajbir Singh Cheema and Parvinder S. Sandhu

Abstract—Software Reusability is primary attribute of software quality. Software Reuse promises significant improvements in software productivity and quality. Our approach, for identification and evaluation of reusable software, is based on software models and metrics. Hence, in this paper we have taken the dataset and framework of metrics, discussed in our earlier papers [2], [3] and [6]. With the objective of taking advantage of the features of the kNN clustering, in this study KNN 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 function/procedure based metric values as representation of the attributes of the software component and output is being obtained in terms of reusability.

Keywords—KNN Clustering, Procedure Oriented, Metric, Software Reusability.

I. INTRODUCTION

SOFTWARE reusability is an attribute that refers to the expected reuse potential of a software component. In computer science and software engineering, reusability is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification. [3]

Reusable modules and classes reduce implementation time, increase the likelihood that prior testing and use has eliminated bugs and localizes code modifications when a change in implementation is required. Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers [1].

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. 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. Hence, in this paper we have taken the dataset and framework of metrics, discussed in our earlier papers [2], [5] and [6], that make use of McCabe’s Cyclometric Complexity Measure for Complexity measurement, Regularity Metric, Halstead Software Science Indicator for Volume indication, Reuse Frequency metric and Coupling Metric values of the software component as input attributes and kNN based Approach is experimented to classify the procedure based software modules in the Reusable and Non-Reusable Categories.

This paper consists of 5 sections. The second section explains about KNN Clustering algorithm In the third section the methodology of evaluating reusability is discussed. In the fourth section implementation results are illustrated and in the fifth section conclusion is written.

II. METHODOLOGY

Reusability evaluation System for Procedure based Software Components can be framed using following steps:

A. Selection and refinement of metrics targeting the reusability of Procedure Based software system and perform parsing of the software system to generate the Meta information related to that Software. The metric of the [2], [5] and [6] are used and the details of metrics are as under. The proposed five metrics for Object-Oriented Paradigm are as follows:

i) Cyclometric Complexity Using Mc Cabe’s Measure: According to Mc Cabe, the value of Cyclometric Complexity (CC) can be obtained using the following equation:
\[ CC = \text{Number of predicate nodes} + 1 \] \hspace{1cm} (1)

Where predicate nodes are the nodes of the directed graph, made for the component, where the decisions are made.

Hence, the value of \( CC \) of a software component should be in between upper and lower bounds as a contribution towards reusability.

If \( CC \) is high with high regularity of implementation then there exists high functional usefulness.

\( \text{ii) Halstead Software Science Indicator} \)

According to this metric volume of the source code of the software component is expressed in the following equation:

\[ \text{Volume} = N1 + N2 \log 2(\eta 1 + \eta 2) \] \hspace{1cm} (2)

Where, \( \eta 1 \) is the number of distinct operators that appear in the program, \( \eta 2 \) is number of distinct operands that appear in the program, \( N1 \) is the total number of operator occurrences and \( N2 \) is the total number of operand occurrences.

The high volume means that software component needs more maintenance cost, correctness cost and modification cost. On the other hand, less volume increases the extraction cost, identification cost from the repository and packaging cost of the component. So the volume of the reusable component should be in between the two extremes.

\( \text{iii) Regularity Metric} \)

The notion behind Regularity is to predict length based on some regularity assumptions. As actual length (\( N \)) is sum of \( N1 \) and \( N2 \). The estimated length is shown in the following equation:

\[ \text{Estimated Length} = N' = \eta 1 \log 2 \eta 1 + \eta 2 \log 2 \eta 2 \] \hspace{1cm} (3)

The closeness of the estimate is a measure of the Regularity of Component coding is calculated as:

\[ \text{Regularity} = 1 - \frac{(N' - N')}{N} = \frac{N'}{N} \] \hspace{1cm} (4)

The above derivation indicates that Regularity is the ratio of estimated length to the actual length. High value of Regularity indicates the high readability, low modification cost and non-redundancy of the component implementation.

Hence, there should be some minimum level of Regularity of the component to indicate the reusability of that component.

\( \text{iv) Reuse-Frequency Metric} \)

Reuse frequency is calculated by comparing number of static calls addressed to a component with number of calls addressed to the component whose reusability is to be measured. Let \( N \) user defined components be \( X_1, X_2 \ldots X_N \) in the system, where \( S_1, S_2 \ldots S_M \) are the standard environment components e.g. printf in C language, then Reuse-Frequency is calculated as:

\[ \text{Reuse-Frequency} = \frac{\eta(C)}{\frac{1}{M} \sum_{i=0}^{M} \eta(S_i)} \] \hspace{1cm} (5)

Equation (5) shows that the Reuse-Frequency is the measure of function usefulness of a component. Hence there should be some minimum value of Reuse-Frequency to make software component really reusable.

\( \text{v) Coupling Metric} \)

Functions/methods that are loosely bound tend to be easier to remove and use in other contexts than those that depend heavily on other functions or non-local data. Different types of coupling effects reusability to different extent.

\( \text{Data Coupling} \): Data coupling exists between two functions when functions communicate using elementary data items that are passed as parameters between the two.

\( \text{Stamp Coupling} \): When two functions communicate using composite data item e.g. structure in C language then that kind of coupling is called Stamp Coupling.

\( \text{Control Coupling} \): If data from one function is said to direct the order of instruction execution in another function then Control Coupling is there between those functions.

\( \text{Common Coupling} \): In case of Common Coupling the two functions share global data items. Weight of coupling increases from category “\( a \)” to “\( d \)”, means Data Coupling is lightest weight coupling, whereas Content Coupling is the heaviest one.

Let

\( a_i \) be the number of functions called and Data Coupled with function “\( i \)”,
\( b_i \) be the number of functions called and Stamp Coupled with function “\( i \)”,
\( c_i \) be the number of functions called by function “\( i \)” and Control Coupled with function “\( i \)”
\( d_i \) be the number of functions Common Coupled with function “\( i \)”

\[ f(x, a, c) = \frac{1}{1 + e^{-a_i(a_0 + a_1 + a_2 + a_3) + c_i(d_0 + d_1 + d_2 + d_3)}} \] \hspace{1cm} (6)

Where \( a = 10, c = 0.5 \) and \( w_i \) for \( i = 1, 2, 3, 4 \) is the weights of the respective the coupling types.

As coupling increases, there is decrease in understandability and maintainability, so there should be some maximum value of the coupling.

\( \text{B. Calculate the metric values of the sampled software components.} \)

\( \text{C. Implement the k-NN Clustering based prediction system in Matlab environment.} \)

There are two main methods of hierarchical clustering algorithm.

Clustering can be a very effective technique to identify natural groupings in data from a large data set, thereby allowing concise representation of relationships embedded in the data. In our study, clustering allows us to group software modules into faulty and non-faulty categories hence allowing for easier understandability.

The k-nearest neighbor’s algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned
to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor. The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its k nearest neighbors. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. (A common weighting scheme is to give each neighbor a weight of 1/d, where d is the distance to the neighbor. This scheme is a generalization of linear interpolation.) The neighbors are taken from a set of objects for which the correct classification (or, in the case of regression, the value of the property) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. The k-nearest neighbor algorithm is sensitive to the local structure of the data.

Nearest neighbor rules in effect compute the decision boundary in an implicit manner. It is also possible to compute the decision boundary itself explicitly, and to do so in an efficient manner so that the computational complexity is a function of the boundary complexity [4].

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, k is a user-defined constant, and an unlabelled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. Usually Euclidean distance is used as the distance metric; however this is only applicable to continuous variables. In cases such as text classification, another metric such as the overlap metric (or Hamming distance) can be used.

Here is step by step to compute K-nearest neighbors KNN algorithm:

D. Compare the results on the Accuracy percentage, Probability of Detection and Probability of False Alarms.

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

### Table I. Confusion Matrix of Prediction Outcomes.

<table>
<thead>
<tr>
<th>Predicted Value</th>
<th>Reusable or Class-2</th>
<th>Non-Reusable or Class-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td></td>
<td>FP</td>
</tr>
<tr>
<td>FN</td>
<td></td>
<td>TN</td>
</tr>
</tbody>
</table>

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

- **Probability of Detection (PD),** also called recall or specificity, is defined as the probability of correct classification of a module that contains a fault.
  \[ PD = \frac{TP}{TP + FN} \] (3)
- **Probability of False Alarms (PF)** is defined as the ratio of false positives to all non defect modules.
  \[ PF = \frac{FP}{FP + TN} \] (4)
- **Accuracy %:** It is calculated as sum of TP and TN divided by total modules.

### III. RESULTS AND DISCUSSION

The function 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 representing Non-Reusable and Reusable components respectively. The Graphical representation of the count of the number of examples of certain reusability label is shown in the Fig. 1.

![Bar-chart of Count of examples of the Reusability Output Attribute in the Dataset](image)

The statistics shows that in the dataset, there are 55 examples of label 1 and 54 examples of label 2. 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 is with five Input Attributes i.e. Coupling, Volume, Complexity, Regularity, Reuse Frequency, and one Output attributes named as Reusability Level of the Software Component. Then hierarchical clustering based algorithm is
implemented in Matlab 7.4.

When the kNN is applied then the probability of detection is calculated as 1 and the probability of false alarm is calculated as 0 and the accuracy of the systems is 100% as evidenced by the following output snapshot of the system:

<table>
<thead>
<tr>
<th>Prob_of_Detection</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob_of_False_Alarms</td>
<td>0</td>
</tr>
<tr>
<td>Accuracy_percentage</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 2 Snapshot of the Output

IV. CONCLUSION

In this study, kNN based approach 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 two linguistic values. Five Input metrics are used as Input and clusters are formed using kNN based machine learning approach, thereafter performance of the system is recorded. The proposed technique is showing Accuracy value approximately equal to 100% and probability of Detection is calculated as 1 and the probability of false alarm is calculated as 0, so it is perfect enough to use the kNN based technique for the identification of the procedure based reusable modules from the existing reservoir of software components.

REFERENCES