A Hierarchical Clustering Based Approach for Prediction of Level of Severity of Faults in Software Systems

Shilpa Sharma, Meenakshi Sharma, Dr. Parvinder S. Sandhu

Abstract— In this study, a survey on quality prediction of software systems is performed. Here, the quality is accessed using on the basis of fault proneness of the systems. The work of various researcher is discussed in brief.

Keywords— Software Quality, Software Metrics, Function Based, Object Oriented.

I. INTRODUCTION

SOFTWARE maintenance is defined as the process of modifying existing operational software after delivery to the customer to correct faults, to improve performance, and/or to adapt the product to a changed environment. Maintenance is inevitable for almost any kind of product. However, most products need maintenance due to the wear and tear caused by use. On the other hand, software products do not need maintenance on this count, but need maintenance to correct errors, enhance features, port to new platforms etc.

The software industry has been experiencing a software crisis, a difficulty of delivering software within budget, on time, and of good quality. At the same time, the industry has experienced a dramatic increase in the software life cycle costs of maintenance. It has been noted [1] that over 50% of programmer effort is dedicated to maintenance. According to Mall [8] the effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratios. Given this high cost, some organizations are beginning to look at their maintenance processes as areas for competitive advantage.

With real-time systems becoming more complex and unpredictable, partly due to increasingly sophisticated requirements, traditional software development techniques might face difficulties in satisfying these requirements. Future real-time software systems may need to dynamically adapt themselves based on the run-time mission-specific requirements and operating conditions. This involves dynamic code synthesis that generates modules to provide the functionality required to perform the desired operations in real-time. However, this necessitates the need to develop a real-time assessment technique that classifies these dynamically generated systems as being faulty / maintenance free [2].

A variety of software fault predictions techniques have been proposed, but there is very less work on the prediction of Level of severity of faults present in the software system. As the major fault requires immediate attention or maintenance but the minor fault can be treated later on or the maintenance of the minor fault can be avoided depending on its level of severity (in case when there is very less time left for the final supply of the software). Therefore, still there is a need to find the best prediction technique for a given prediction dataset to calculate the level of severity of faults in the software systems. Hence, main aim of this study is to study Prediction of Level of Severity of Faults in Software Systems with Hierarchical Clustering based technique.

II. RELATED WORK

The basic hypothesis of software quality prediction is that a module currently under development has defects if a module with the similar product or process metrics in an earlier project (or release) developed in the same environment had defects [3]. Therefore, the information available early within the current project or from the previous project can be used in making predictions. This methodology is very useful for the large-scale projects or projects with multiple releases.

Maintenance managers can apply existing techniques that have been traditionally been used for other types of applications. One system is not enough for prediction purposes. The empirical study detailing software maintenance for web based java applications can be performed to aid in understanding and predicting the software maintenance category and effort [4].

With the advent of Total Quality Management, organizations are using metrics to improve quality and productivity [5]. Software maintenance organizations are no exception. In 1987, the U.S. Navy established centralized Software Support Activity (SSA) to provide software maintenance for cryptologic systems. At that time two systems were supported and a software maintenance metrics program was established to support the goals of the SSA.

Visual approach [6] can be used to uncover the
relationship between evolving software and the way it is affected by software bugs. By visually putting the two aspects close to each other, we can characterize the evolution of software artifacts.

Software maintenance is central to the mission of many organizations. Thus, it is natural for managers to characterize and measure those aspects of products and processes that seem to affect cost, schedule, quality, and functionality of a software maintenance delivery [9]. The importance of software maintenance in today's software industry cannot be overstated.

Statistical, machine learning, and mixed techniques are widely used in the literature to predict software defects. Khoshgoftaar [10] used zero-inflated Poisson regression to predict the fault-proneness of software systems with a large number of zero response variables. He showed that zero-inflated Poisson regression is better than Poisson regression for software quality modeling. Munson and Khoshgoftaar [11, 12] also investigated the application of multivariate analysis to regression and showed that reducing the number of "independent" factors (attribute set) does not significantly affect the accuracy of software quality prediction.

Menzies, Ammar, Nikora, and Stefano [13] compared decision trees, naïve Bayes, and 1-rule classifier on the NASA software defect data. A clear trend was not observed and different predictors scored better on different data sets. However, their proposed ROCKY classifier outscored all the above predictor models. Emam, Benlarbi, Goel, and Rai [14] compared different case-based reasoning classifiers and concluded that there is no added advantage in varying the combination of parameters (including varying nearest neighbor and using different weight functions) of the classifier to make the prediction accuracy better.

Bayesian Belief Networks (also known as Belief Networks, Causal Probabilistic Networks, casual Nets, Graphical Probability Networks, Probabilistic Cause-Effect Models, and Probabilistic Influence Diagrams) [15] have attracted much recent attention as a possible solution for the problems of decision support under uncertainty. Although the underlying theory (Bayesian probability) has been around for a long time, the possibility of building and executing realistic models has only been made possible because of recent algorithms and software tools that implement them. Clearly defects are not directly caused by program complexity alone. In reality the propensity to introduce defects will be influenced by many factors unrelated to code or design complexity.

Many modeling techniques have been developed and applied for software quality prediction. These include logistic regression, discriminant analysis [16, 17], the discriminative power techniques, Optimized Set Reduction, artificial neural network [18-19] etc. For all these software quality models, there is a tradeoff between the defect detection rate and the overall prediction accuracy. The software quality may be analyzed with limited fault proneness data [20].

### III. METHODOLOGY USED

In course of the research work, the following steps are proposed to be carried out:

I. Study of the metrics needed for maintenance.

II. Collect the sampled relevant metric data.

III. Analyze and refine metrics data.

IV. Implement the Hierarchical Clustering based prediction system in Matlab environment.

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:

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 in figure 1. Thereafter, Deduce the results on the 10 fold cross validation accuracy, precision and recall values.

In case of the two-cluster based problem let's take major and minor fault class, the confusion matrix has four categories: True positives (TP) are 'Major Fault' modules correctly classified as 'Major Fault' modules. False positives (FP) refer to ‘Minor Fault’ modules incorrectly labeled as ‘Major Fault’ modules. True negatives (TN) correspond to ‘Minor Fault’ modules correctly classified as such. Finally, false negatives (FN) refer to ‘Major Fault’ modules incorrectly classified as ‘Minor Fault’ modules as shown in table I.

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 fault 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). The equation is shown below:

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

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:

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

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

REFERENCES


