Received: February 17, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Fault Prone Analysis of Software Systems Using Rough Fuzzy C- means Clustering

Neha Singh[1] Agrawal Kalpesh[1] Swathi Jamjala Narayanan[1*]

[1]*School of Computer Science and Engineering, VIT University, Vellore, India*
* Corresponding author's Email: jnswathi@vit.ac.in

Abstract: Prediction of fault proneness of modules in software is one of the ways to ensure the achievement of software quality and reliability. Software delivered cannot not always be bug free, the more the number of bug the more the dis-satisfaction among client due to degradation in software quality and reliability. Though we have few models for detecting software fault prone modules, the intend of our work is to increase the reliability of the software by using an approach named Rough Fuzzy c-means (RFCM) clustering algorithm to analyse the fault proneness of the software modules under test. This helps in an efficient analysis of the modules having an ambiguous behaviour using rough set boundary which is not possible using traditional clustering methods. A dataset from PROMISE software engineering repository has been taken for the experimental analysis. The results were promising enough to determine software modules which fall in the boundary region of ambiguity emphasizing the software team to focus on those modules to achieve higher reliable system.

Keywords: Fault proneness, C- means, FCM, RFCM, Clustering, Ambiguous faults, Boundary region, Rough set.

1. Introduction

All Faults in software systems continue to be a major problem. Many software systems are delivered to users/client with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. Fault-proneness of a software module is the probability that the module contains faults. A correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly important and testing related issues are becoming crucial for software. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process. In the past, several metrics for measuring software complexity and testing thoroughness have been proposed. Static metrics, e.g., the McCabe's cyclomatic number or the Halstead's Software Science, statically computed on the source code and tried to quantify software complexity. Despite this it is difficult to identify a reliable approach to identifying fault-prone software components.

Clustering is used to determine the intrinsic grouping in a set of unlabelled data. It is the process of organizing objects into groups whose members are similar in some way. Among various clustering techniques available in literature K-Means clustering approach is most widely used technique. But due to the crisp nature of the technique some modules having an ambiguous behaviour may be misclassified into a wrong class. For example, some software module only show either fail result of pass result, there are changes that module sometimes pass

in certain cases but can fail in certain tests leading to ambiguous decision, whether the system is correctly working and must be delivered or is faulty must be re-engineered. Thus, leading to degradation in the process of analysis of the fault proneness of the system and thus affecting the quality of the software.

To overcome this shortcoming, we have used RFCM clustering technique. It is a hybrid technique which deals with ambiguities, vagueness and indiscernibility using the concepts from the theory of fuzzy sets and rough sets both. The fuzzy nature helps in finding the membership of an element for given cluster. Thus we are able to calculate that amount of similarity between element and the cluster to which it has been assigned. The drawback here is the generation of overlapping clusters. These overlapping clusters make it hard to decide the assignment of elements to one specific cluster in the real world scenario. This is the place where the use of rough set theory proves useful. It provides a region for data points which completely belong to a given cluster and this region is called lower approximation. The next region that exists is the upper approximation where the data points aren't a part of the cluster. The region in between the lower and the upper approximation is the region known as boundary region. It is here that the data points not having a crisp belongingness to either of the aforementioned area lie. In this paper RFCM is used for the process of investigating the fault proneness of software modules for the dataset obtained from the PROMISE software engineering repository.

The paper is organized as follows: section 2 provides the details of existing work related to software fault prediction. The next section deals with the traditional clustering methods and their drawbacks. Section 4 provides the proposed RFCM clustering for measuring the fault proneness of software modules. In section 5, the computational experimental results have been discussed followed by the references.

2. Literature review

All Several efforts have been made in research for software fault prediction and assessment using various techniques [1-3]. Agresti and Evanco [4] worked on a model to predict defect density based on the product and process characteristics for Ada program. There are many papers advocating statistical models and software metrics [5]. Gaffney and Davis [6, 7] of the Software Productivity Consortium developed the phase-based model. It uses fault statistics obtained during the technical review of requirements, design, and the coding to predict the reliability during test and operation.

One of the earliest and well known efforts to predict software reliability in the earlier phase of the life cycle was the work initiated by the Air Force's Rome Laboratory [8]. For their model, they developed prediction of fault density which they could then transform into other reliability measures such as failure rates.

To do this the researchers selected a number of factors that they felt could be related to fault density at the earlier phases. Most of them are based on size and complexity metrics. To achieve high software reliability, the number of faults in delivered code should be reduced. The faults are introduced in software in each phase of software life cycle and these faults pass through subsequent phases of software life cycle unless they are detected through testing or review process. Finally, undetected and uncorrected faults are delivered with software. To achieve the target software reliability efficiently and effectively, faults should be identified at early stages of software development process. During early phase of software development testing/field failure data is not available. Therefore, the prediction is carried out using various factors relevant to reliability.

A study was conducted by Zhang and Pham [9] to find the factors affecting software reliability. The study found 32 potential factors involved in various stages of the software life cycle. In another recent study conducted by Li and Smidt [10], reliability relevant software engineering measures have been identified. They have developed a set of ranking criteria and their levels for various reliability relevant software metrics, present in the first four phases of software life cycle. Recently, Kumar and Misra [11] tried for early software reliability prediction considering the six top ranked measures given by [10] and software operational profile. Sometimes, it may happen that some of these top ranked measures are not available, making the prediction result unrealistic.

Software metrics can be classified in three categories: product metrics, process metrics, and resources metrics [12]. Product metrics describe characteristics of the product such as size, complexity, design features, performance and quality level etc. Process metrics can be used to improve software development process and maintenance. Resources metrics describe the project characteristics and execution. Approximately thirty software metrics exist, which can be associated with different phases of software development life cycle. Among these metrics some are significant predictor

to reliability. From the above literature, following observations are made. Firstly, it is observed that predicting faults early is very important for the entire software development process and reliability. Secondly, the reliability of software is a function of the number of the remaining faults. Thirdly, data available from existing software's and their proneness to faults can be of great use for detecting the effectiveness of a technique in the prediction of same. The existing data can be analysed and the results compared with the already available classification results. Also, addition of soft computing techniques to these methods can improve the prediction ability of the software.

In this paper, we propose to use a hybrid RFCM technique which makes use of concepts from the theory of both fuzzy sets and rough sets to handle ambiguity and vagueness in better way. The fuzzy nature helps in finding the membership of an element for given cluster. Thus we are able to calculate that amount of similarity between element and the cluster to which it has been assigned. The drawback here is the generation of overlapping clusters.

These overlapping clusters make it hard to decide the assignment of elements to one specific cluster in the real world scenario. This is the place where the use of rough set theory proves useful. It provides a region for data points which completely belong to a given cluster and this region is called lower approximation. The next region that exists is the upper approximation where the data points aren't a part of the cluster. The region in between the lower and the upper approximation is the region known as boundary region. It is here that the data points not having a crisp belongingness to either of the aforementioned area lie.

3. Traditional clustering methods

In this session we discuss about traditional c-means and fuzzy c- means clustering technique.

3.1 C- means clustering

C-means is the most widely used prototype based partitional clustering algorithms. It is an iterative process until all data points stabilizes [13]. In hard c-means, each object must be assigned to exactly one cluster. The clustering ensures that the similarity between the data points with in a cluster is maximum than the data points of other clusters. The objective of this algorithm is to minimize the squared error function given in Eq. (1)

$$J=\sum_{j=1}^{k}\sum_{i=1}^{n}\left\|x_i^j-c_j\right\| \quad (1)$$

Where, $\left\|x_i^j-c_j\right\|$ is a chosen measure of distance between a data point x_i^j and the cluster center c_j is an indicator of the distance of the *n* data points from their respective cluster centers.

3.2 Fuzzy c- means clustering

The fuzzy c-means (FCM) algorithm proposed by Bezdek [14, 15] is the fuzzy variant of the conventional c-means algorithm. The algorithm is based on the minimization of the objective function given in Eq. (2)

$$J=\sum_{j=1}^{k}\sum_{i=1}^{n}u_{ij}^{m}\left\|x_i^j-c_j\right\|^2 \quad (2)$$

This methodology is useful when the required number of clusters is available. The primary improvement of FCM over conventional c-means is the fact that it can assign partial membership to data points. This means the use of concept of likelihood rather than complete membership. The method can be very useful at times but is highly susceptible to noise and outliers. The primary drawback is the fact that it generates overlapping clusters. The membership values is calculated using Eq. (3) and the fuzzy centers are calculated using Eq. (4)

$$u_{ij}=1/\sum_{p=1}^{k}\left(\frac{\|x^i-c_j\|}{\|x^i-c_p\|}\right)^{\frac{2}{m-1}} \quad (3)$$

subject to :

$$\sum_{j=1}^{k}u_{ij}=1, \quad \forall i$$

$$0<\sum_{j=1}^{n}u_{ij}<n, \quad \forall j$$

Where, μ_{ij} is the membership of i^{th} data to j^{th} cluster, m is the fuzzy index where $m \in (1,\infty)$, c is the number of cluster centres.

$$c_j=\sum_{i=1}^{n}(u_{ij})^m x_i/\sum_{i=1}^{n}(u_{ij})^m \quad (4)$$

Where, c_j is the fuzzy center of the cluster *j*.

The algorithm iteration stops, when the condition $\max_{ij}\{|u_{ij}^{k+1}-u_{ij}^{k}|\}<\varepsilon$, where ε is the termination criteria between 0 and 1 is and *k* is the number of iterations.

3.3 Limitations of traditional clustering methods

There are existent drawbacks to the c-means and the fuzzy variant of this clustering algorithm. They take long computational time and c-means is sensitivity to the initial guess and thus leads to local minima. The fuzzy c-means is also sensitivity to noise and provides low (or even no) membership degree for outliers or to the noisy data.

4. Rough fuzzy c-means clustering algorithm for software fault prediction

RFCM algorithm extends the c-means algorithm by integrating both fuzzy set and rough set theory [16, 17]. In this hybrid method, the benefits of both fuzzy sets and rough sets are added to the traditional c- means and thus we obtain a robust algorithm [18]. The fuzzy set algorithm provides membership values to each of the elements of the set and it ends as overlapping partitions which are helpful in few applications. In rough set theory, the concepts like upper approximation, lower approximation and boundary condition further helps to handle ambiguity in better way than Fuzzy c- means.

By hybridizing rough, fuzzy and c-means together it enables to efficiently handle the uncertainty, incompleteness and vagueness of dataset during classification or clustering [19, 20]. In RFCM technique, each cluster consists of 3 parameters, namely, a cluster centroid which is the mean or prototype of the cluster, a crisp lower approximation which holds the data points completely belonging to the set, and a fuzzy boundary which holds data points possibly belonging to the set and also falls under the rough boundary with fuzzy membership.

The formal representation of the notations and the RFCM algorithm used for predicting software fault prone modules is as follows: Let $\underline{L}(v_j)$ and $\overline{U}(v_j)$ be the lower and upper approximations of cluster v_j, and $B(v_j) = \overline{U}(v_j) - \underline{L}(v_j)$ denote the boundary region of cluster v_j.

4.1 Objective function

The objective function of the RFCM algorithm which is to be minimized is as given in Eq. (5). In Eq. (5) the parameters w and $\tilde{w}$ =1-w are the weights related to lower approxiamtion and the boundary region. μ_{ij} is the mebership of the object i in cluster j.

$$J_{RF} = \begin{cases} w \times A_1 + \tilde{w} \times B_1 & \text{if } \underline{L}(v_j) \neq \emptyset,\ B(v_j) \neq \emptyset \\ A_1 & \text{if } \underline{L}(v_j) \neq \emptyset,\ B(v_j) = \emptyset \\ B_1 & \text{if } \underline{L}(v_j) = \emptyset,\ B(v_j) \neq \emptyset \end{cases} \quad (5)$$

Where

$$A_1 = \sum_{j=1}^{k} \sum_{x_i \in \underline{L}(v_j)} u_{ij}^m \left\| x_i - c_j \right\|^2; \quad (6)$$

and

$$B_1 = \sum_{j=1}^{k} \sum_{x_i \in B(v_j)} u_{ij}^m \left\| x_i - c_j \right\|^2 \quad (7)$$

According to the lower approximations and boundary region definitions of rough sets, if an object $x_i \in \underline{L}(v_j)$, then $x_i \notin \underline{L}(v_p), \forall p \neq j$ and $x_i \notin B(v_j)$, $\forall j$. In simple, the object x_i is certainly enclosed in cluster v_j. In this case, the weights of the objects are independent of other centroids and clusters.

Further, the objects that fall in the lower approximation should have similar influence on the coresponding centroid and cluster rather, if $x_i \in B(v_j)$, then the object x_i possibly belongs to v_j and potentially belongs to another cluster. Hence, the objects in boundary regions should have different influence on the centroids and clusters. So, in RFCM, the data is partitioned into two classes - lower approximation and boundary and only the objects in boundary are fuzzified as in fuzzy c-means and the membership values of objects in lower approximation is μ_{ij}=1. Thus in RFCM A_1 reduces to

$$A_1 = \sum_{j=1}^{k} \sum_{x_i \in \underline{L}(v_j)} \left\| x_i - c_j \right\|^2 \quad (8)$$

and B_1 has the same expression as given in Eq. (7).

4.2 Cluster centroids

The new cluster prototypes (centroids) are calculated as given in Eq. (9).

$$C_j^{RF} = \begin{cases} w \times C_1 + \tilde{w} \times D_1 & \text{if } \underline{L}(v_j) \neq \emptyset, B(v_j) \neq \emptyset \\ C_1 & \text{if } \underline{L}(v_j) \neq \emptyset,\ B(v_j) = \emptyset \\ D_1 & \text{if } \underline{L}(v_j) = \emptyset,\ B(v_j) \neq \emptyset \end{cases} \quad (9)$$

Where

$$C_l = \frac{1}{|\underline{L}(v_j)|} \sum_{x_i \in \underline{L}(v_j)} x_i \quad (10)$$

$$D_l = \frac{1}{n_j} \sum_{x_i \in B(v_j)} u_{ij}^m \; x_i \quad (11)$$

$$n_i = \sum_{x_j \in B(\beta_i)} (u_{ij}^m) \quad (12)$$

and $|\underline{L}(v_j)|$ represents the cardinality of $\underline{L}(v_j)$.

The equation is based on the weighting average of the crisp lower approximation and fuzzy boundary and has both the effects of fuzzy memberships and lower and upper bounds.

From Eq. (9), we observe that the cluster prototypes (centroids) depends on the parameters w and $\tilde{w}$, and fuzzifer m rule their relative influence and the values are given by $0 < \tilde{w} < w < 1$.

4.3 RFCM algorithm

The algorithmic steps are as follows:

Step 1: Assign initial centroids c_j , where $j = 1,\ldots,k$
Step 2: Set value for m, ϵ , δ and iteration counter
Step 3: Compute u_{ij}^m using Eq. (3) for k clusters and n objects.
Step 4: If u_{ij} and u_{ip} are the two highest memberships of the object x_i and ($u_{ij} - u_{ip}$) $\leq \delta$, then $x_i \in \overline{U}(v_j)$ and $x_i \in \overline{U}(v_p)$ Further, x_i is not part of any lower bound.
Step 5: Otherwise, $x_i \in \underline{L}(v_j)$ and by the properties of rough sets, $x_i \in \overline{U}(v_j)$
Step 6: Update u_{ij} for k clusters and n objects considering their lower and boundary regions
Step 7: Compute new centroid using Eq. (9)
Step 8: Repeat the steps 2 to 7, by incrementing t, until $| u_{ij}(t) - u_{ij}(t-1)| > \epsilon$.

In RFCM, the partitioning of the data set into lower approximation and boundary is mainly based on the value of δ which is determined basically using Eq. (13)

$$\delta = \frac{1}{n} \sum_{i=1}^{n} (u_{ij} - u_{ip}) \quad (13)$$

Where, n is the total number of objects, u_{ij} and u_{ip} are the highest and second highest memberships of x_i.

The flow chart of the proposed algorithm is given in Fig. 1. The system design for the proposed work is given in Fig. 2.

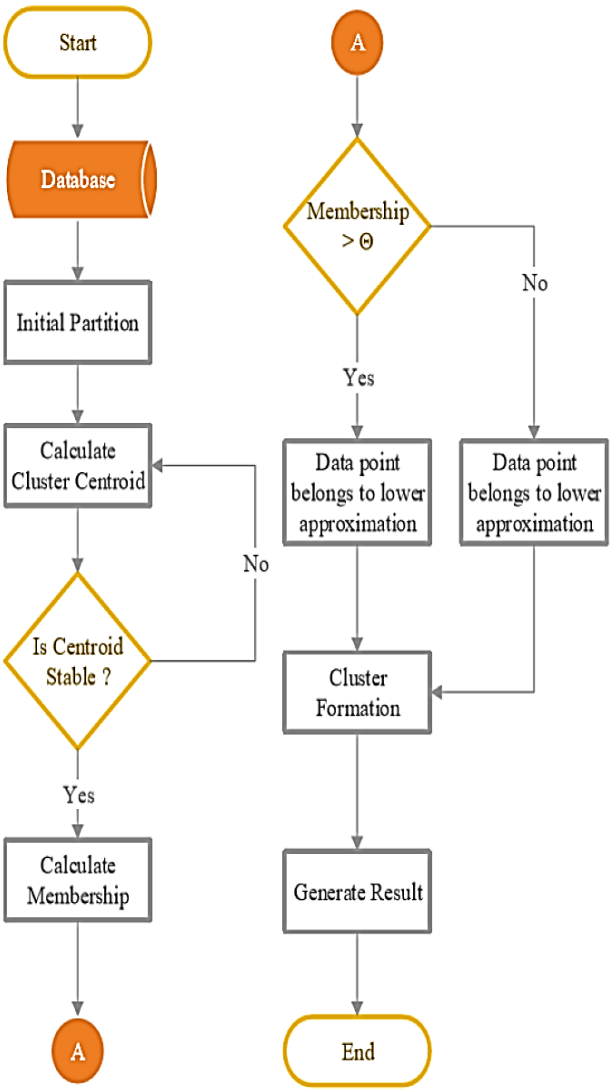

Figure.1 Flowchart for RFCM Approach

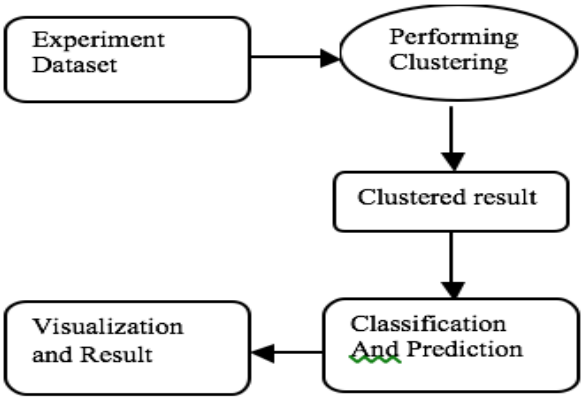

Figure.2 System design

5. Computational experiments and the results obtained

For conducting experiments on the proposed techniques, DATATRIEVE Transition/Software defect prediction dataset is chosen from PROMISE repository [21]. For predicting future error prone software modules, we have used unsupervised techniques namely, c means, fuzzy c- means and Rough fuzzy c- means.

The c-means algorithm basically partitions the dataset into specified number of crisp clusters where we don't have option of predicting future error prone modules. In the second technique, fuzzy c-means clustering technique, the software modules are partitioned into set of overlapping clusters where modules have the membership of both fault prone and non-fault prone cluster. As our intended aim is to find the future error prone ambiguous module, the third technique, hybrid rough fuzzy c- means clustering technique determines the ambiguous modules falling in the boundary region.

By the term ambiguous, we mean that the particular modules can't be classified with surety for their degree of being prone to potential errors. Such modules, if when classified using crisp clustering techniques may be misclassified as completely safe or vice versa thus increasing the risk of unexpected faults in the software system.

The results obtained for the set of experiments conducted are given in Table 1. The results show that traditional partitioning method, c- means has generated crisp binary results depicting whether a module is fault prone or non-fault prone.

Table 1. Computational experimental results

Class	**c- means**	**FCM**	**RFCM**
Non fault prone	115	110	107
Fault Prone	15	50	30
Total	130	160	137
Overlapping	No	Yes	Yes
Boundary	Not detected	Not detected	Detected with 7 modules
Ambiguous region	Not present	Not ambiguous but its overlapping clusters	Elements in boundary region are ambiguous

While applying Fuzzy c-means, we find the modules falling in both the clusters and hence there is overlap among the clusters stating that 30 modules belong to both the clusters which cannot be termed as ambiguous but stated as overlap. On the contrary, when Rough fuzzy c-means is applied, there are seven modules lying in the boundary showing an ambiguous behavior.

The benefit of RFCM would be that, the boundary region in RFCM accommodates the ambiguous elements (software modules) which could lead to unexpected failures. The existence of upper approximation region in RFCM makes it easy to deal with outliers and the lower approximations helps in keeping the membership of particular modules to a specific cluster thus dealing with their overlapping nature.

The data points in Fig. 3 are the set of data points representing the software modules to be clustered for partitioning into sets of fault prone modules and non-fault prone modules. When clustering techniques are applied on the data points, the data points are grouped into clusters as given in Fig. 4 and Fig. 5.

In these figures, the green colored data points belong to cluster 1 and yellow colored data points belong to cluster 2 and the red indicator indicates the cluster centroids. From Fig. 4, we also observe that there are few data points belonging to both clusters being on the same point thus indicating ambiguity.

In Fig. 5, the boundary shows some set of modules which are to be concentrated much as those are future error prone software modules which may affect software reliability. Hence, the results of these clustering techniques help the software test team and the developers to concentrate on these modules to great extent so that detailed analysis of those modules will help in achieving high reliable systems.

Figure.3 Set of software modules before partitioning

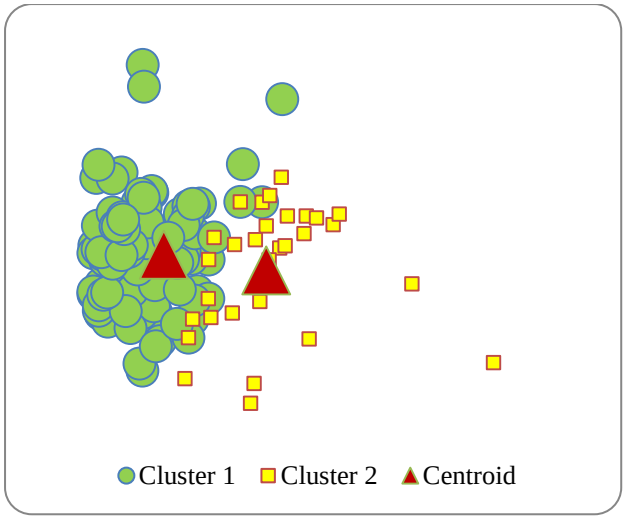

Figure.4 Software modules clustered into fault Prone and non-fault Prone without boundary region

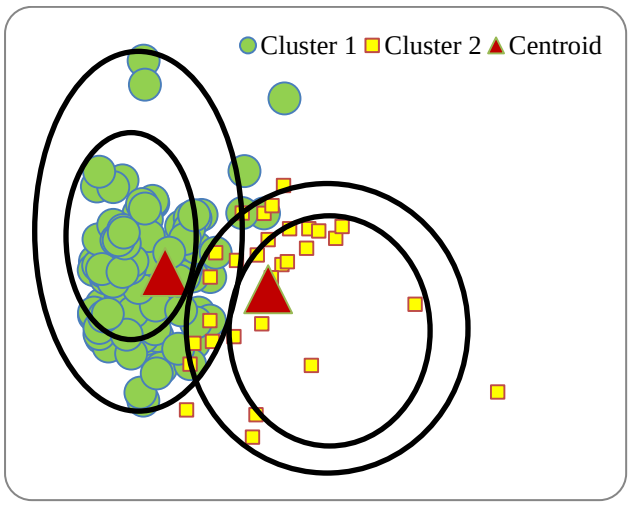

Figure.5 Software modules clustered into fault Prone and non-fault Prone with overlapping boundary region

6. Conclusion

Software fault prediction is considered to be an important task in software development process. The conventional machine learning methods for classification allows the user to know whether the module under test is fault prone or not- fault prone.

In reality, there is a chance that, the module which is passed in testing is not concentrated much for future faults that might arise due to several reasons. In this paper, we used RFCM clustering technique to find the ambiguous modules. Dataset from PROMISE software engineering repository has been taken for the analysis. The obtained results were promising enough in achieving the separation between fault prone modules, non-fault prone modules and the boundary region which holds modules which might be fault prone in future. This is the benefit obtained using the proposed method Rough Fuzzy c- means approach which helps in highlighting the ambiguous software modules whose prediction increases the reliability of the system when the software team focuses on those modules. Basically, it helps the software team to take pro-active measures towards possible software failures, which otherwise would go unnoticed if some conventional technique were used for the analysis.

The precision of this technique is higher when compared to conventional techniques and this makes it suitable for tasks which are critical in nature. Hence, the proposed approach has more chances of applicability for several other applications in real world. In future, the performance of the clustering can be assessed using appropriate cluster validity measures to know the quality of the clusters formed and also optimization techniques can be applied to further improve the performance of the clustering algorithms.

References

[1] S. Chidamber, and C. Kemerer, "A metrics suite for object-oriented design", *IEEE Transactions on Software Engineering*, Vol. 20, No.6, pp.476-493, 1994.

[2] A. Kaur and R. Malhotra, “Application of Random Forest in Predicting Fault-Prone Classes”, In: *Proc. of International Conf. On Advanced Computer Theory and Engineering*, Phuket, Thailand, pp. 37-43, 2008.

[3] F. Lanubile, A. Lonigro, and G. Visaggio, “Comparing Models for Identifying Fault-Prone Software Components”, In: *Proc. of Seventh International Conf. On Software Engineering and Knowledge Engineering*, pp. 12-19, June 1995.

[4] W.W. Agresti, and W.M. Evanco, "Projecting software defects from analyzing Ada designs", *IEEE Transactions on Software Engineering*, Vol. 18, No. 11, pp. 988-997, Nov 1992.

[5] C.W. Runeson, and M. C. Ohlsson, “A Proposal for Comparison of Models for Identification of Fault-Proneness”, *Dept. of Communication Systems*, Lund University, LNLS 2188, pp. 341-355, Profes 2001.

[6] J E. Gaffney, and C. F. Davis, “An Approach to Estimating Software Errors and Availability”, In: *Proc. of International Workshop On Minnow Brook Workshop on Software Reliability*, SPC-TR-88-007, Version 1.0, 1988.

[7] J.E. Gaffney, and J. Pietrolewiez, "An Automated Model for Software Early Error Prediction (SWEEP)", In: *Proc. of 13th Minnow Brook Workshop on Software Reliability*, Blue Mountain Lake, NY, 1990.

[8] Rome Laboratory (RL), *Methodology for Software Reliability Prediction and Assessment*, Technical Report RL-TR-92-52.Vol.1 and 2, 1992.

[9] X. Zhang, and H. Pham, "An Analysis of Factors Affecting Software Reliability", *The Journal of Systems and Software*, Vol. 50, No.1, pp. 43-56, 2000.

[10] M. Li, and C. Smidts, "A Ranking of Software Engineering Measures Based on Expert Opinion", *IEEE Trans. on Software Eng.*, Vol. 29, No. 9, pp. 811-824, 2003.

[11] K. S. Kumar, and R. B. Misra, " An Enhanced Model for Early Software Reliability Prediction using Software Engineering Metrics", In: *Proc. of 2nd International Conf. on Secure System Integration and Reliability Improvemen*t, pp.177-178, 2008.

[12] S. H. Kan, *Metrics and Models in Software Quality Engineering*, Vol. 2, Addison-Wesley Professional, 2002.

[13] J.MacQueen, "Some Methods for Classification and Analysis of Multivariate Observation", In*: Proc. Fifth Berkeley Symp. Math. Statistica and Probability*, pp. 281-297, 1967.

[14] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[15] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm", *Computers & Geosciences*, Vol. 10, No.2-3, pp. 191-203, 1984.

[16] Z. Pawlak, "Rough sets", *International Journal of Parallel Programming*, Vol.11, No.5, pp. 341-356, 1982.

[17]Z. Pawlak,"Rough Classification", *International Journal of Man-Machine Studies*, Vol. 20, No.5, pp. 469-483, 1984.

[18] P. Maji, and S.K. Pal, "RFCM: A hybrid clustering algorithm using rough and fuzzy sets", *Fundamenta Informaticae*, Vol. 80, No.4, pp. 475-496, 2007.

[19] M. Banerjee, S. Mitra, and S. K. Pal, "Rough-Fuzzy MLP: Knowledge Encoding and Classification", *IEEE Transactions on Neural Networks*, Vol. 9, No.6, pp. 1203–1216, 1998.

[20] D. Dubois, and H. Prade, "Putting Fuzzy Sets and Rough Sets Together", *Intelligent Decision Support*, pp. 203–232, 1992.

[21]G. Boetticher, T. Menzies, and T. Ostrand, "PROMISE Repository of Empirical Software Engineering Data", *West Virginia University, Department of Computer Science*, 2007.