IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 13, Issue 6 (Jul. - Aug. 2013), PP 32-36
www.iosrjournals.org

Using Fuzzy Clustering and Software Metrics to Predict Faults in large Industrial Software Systems

Nurudeen Sherif[1], Nurudeen Mohammed[2]
[1](*Faculty Of Informatics/ Universiti Sultan Zainal Abidin, Malaysia)*
[2](*School Of Information Science and Engineering / Central South University, China)*

Abstract *: Faults are a key problem in software systems. Awareness of possible flaws from the initialization of a project could save money, time and work. Estimating the possible deficiency of software could help in executing software development activities. This paper proposes a model to predict the possibility of faults on a software system before testing. The model predicts possible faults during software development using Fuzzy Clustering and Software Metrics. This research is aimed at predicting faults in large software systems by creating clusters and then finding out the distance of each point in the data set with the clusters created to determine their degree of membership within each cluster*

Keywords*: Software, fault prediction, software metrics, fuzzy clustering*

I. INTRODUCTION

Reliance on software in our daily lives has increased so much in the last decade that in our day living without devices controlled by software is almost impossible. The Industrial domains such as medical applications, power plants, air traffic control and railway signaling have all integrated software as a fundamental part of their operation. Software engineers have to deal with a large number of quality requirements such as reliability, safety, availability, performance, maintainability and security which makes the development of these large software applications very challenging. The industrial reliance on software gives rise to the likelihood of gross crises in the case of a failure and the effect of these catastrophes ranges from economic damage to loss of lives. Therefore, there is an increasing necessity to ensure the steadfastness of software systems. Moreover, it is well known that the earlier a problem can be identified, the better and more cost effectively this problem can be fixed. Therefore, it is necessary to predict faults during the software development.

There are numerous techniques and metrics for investigating fault prone modules which may aid software developers in performing testing activities during development. It is almost impossible to produce software that is free of faults due to the rising complexity and the constraints under which the software is developed. Such faults may lead to an increment in development & maintenance cost and time, due to software failures and decrease customer's satisfaction [1].

Data Clustering is a basic technique in many modeling algorithms. The objective of clustering is to construct new collections of data from large data set. One of the most acceptable contributions to the field of data clustering is Fuzzy C-Means clustering. It has more benefits compared to other methods of data clustering, specifically the ability to split data for different size clusters with fuzzy logic. The Fuzzy C-Means can be seen as the modified version of the k-means algorithm. Which is a method of clustering that allows one piece of data to belong to two or more clusters. The degree of being in a certain cluster is related to the inverse of the distance to the cluster [2]. Fuzzy C-Means iteratively moves the cluster centers to the "right" location within a data set. This research is aimed at predicting faults in large industrial software systems by creating clusters and then finding out the distance of each point in the data set with the clusters created to determine their degree of membership within each cluster. The Factors like Mean Absolute Error, Accuracy and Root Mean Square Error help us in predicting the software system as faulty or fault-free.

The literature, [3]-[17] presents various types of Fault-Proneness Estimation Models. The results are also compared with [18] in which Hierarchical clustering based approach is used for Finding Fault Prone Classes in large software systems. The paper is organized as follows: section II exploits some literature on related works, section III explains the methodology followed in this research and section IV the result of the study. Finally conclusions of the research are presented in section V.

II. RELATED WORKS

Quite a number of efforts have been made in research for software fault prediction and assessment using various techniques [3] – [5]. Agresti and Evanco [6] 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 [7, 8]. Gaffney and Davis [9, 10] 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 [11]. 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. In order 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. In order 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 [12] 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 [13], 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 [14] made an effort for early software reliability prediction considering the six top ranked measures given by [13] and software operational profile. Sometimes, it may happen that some of these top ranked measures are not available, making the prediction result unrealistic. Also they have considered only product metrics and ignored process metrics that have a great impact on software reliability [15].

Software metrics can be classified in three categories: product metrics, process metrics, and resources metrics [16]. 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 [13]. From the above literature we have observed that

1. Predicting faults early is very important for the entire software development process and reliability.
2. The reliability of software is a function of the number of the remaining faults.
3. Software metrics plays a vital role in early fault prediction in the absence of failure data.

Review of literature indicates that traditional models have not considered the both software metrics and development process maturity, for the early fault prediction. Therefore this paper proposes a model for early software fault prediction considering software metrics and process maturity together.

III. METHODOLOGY

There are several ways of identifying fault prone modules in a software application. First of all, find the structural code and design attributes of software systems. Thereafter, select the suitable metric values as representation of statement. Next step is to analyze, refine metrics and normalize the metric values. We used JEdit open source software in this study. JEdit is a programmer's text editor developed using Java language. JEdit combines the functionality of Window, UNIX, and Mac OS text editors. It was released as free software and the source code is available on [20]. JEdit includes 274 classes. The number of developers involved in this project was 144. The project was started in 1999. The number of bugs was computed using SVC repositories. The release point for the project was identified in 2009. The log data from that point to 2012 was collected. The header files in C++ were excluded in data collection. The word bug or fixed was counted. Details on bug collection process can be found in [19]. The following are the metrics used in the classification process:

1. Coupling between Objects
2. Lack of Cohesion
3. Number of Children
4. Depth of inheritance
5. Weighted Methods per Class
6. Response for a class
7. Number of Public Methods
8. Lines of Code

Using Fuzzy C Means Clustering algorithm we can group the software components into faulty and fault-free systems. 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.

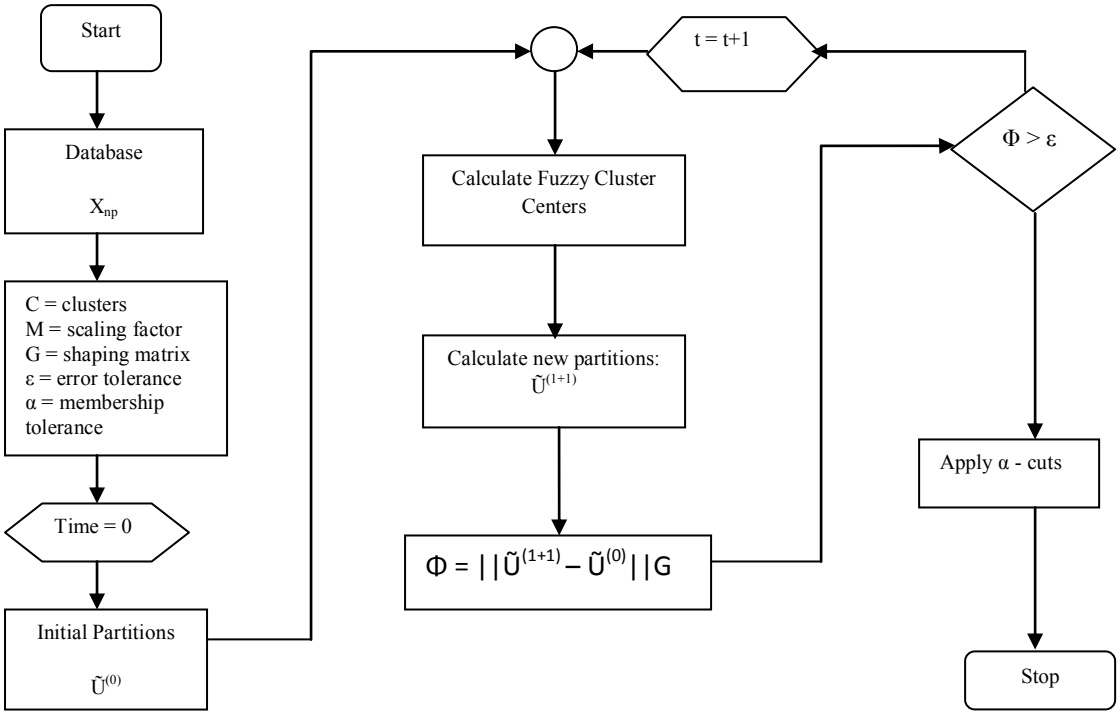

Fig.1 A flowchart depicting Fuzzy C - Means Clustering algorithm.

The Fuzzy C - Means Clustering algorithm attempts to partition a finite collection of n elements $X = \{x_1, ..., x_n\}$ into a collection of c fuzzy clusters with respect to some given criterion. Given a finite set of data, the algorithm returns a list of c cluster centres $C = \{c_1, ..., c_c\}$ and a partition matrix $W = w_{i,j} \in [0,1],\ i = 1, ..., n,\ j = 1, ..., c$, where each element $\mathbf{W_{ij}}$ tells the degree to which element $\mathbf{X_i}$ belongs to cluster $\mathbf{c_j}$. Like the k-means algorithm, the Fuzzy C - Means aims to minimize an objective function. The standard function is:

$$w_k(x) = \frac{1}{\sum_j \left(\frac{d(\text{center}_k, x)}{d(\text{center}_j, x)}\right)^{2/(m-1)}}.$$

Any point *x* has a set of coefficients giving the degree of being in the kth cluster $w_k(x)$. With Fuzzy C - Means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$c_k = \frac{\sum_x w_k(x)x}{\sum_x w_k(x)}.$$

The degree of belonging, $w_k(x)$, is related inversely to the distance from x to the cluster center as calculated on the previous pass. It also depends on a parameter m that controls how much weight is given to the closest center.

To predict the results, we have used confusion matrix as shown in Table I. The confusion matrix has four categories: True positives (TP) are the modules correctly classified as faulty modules. False positives (FP) refer to fault-free modules incorrectly labeled as faulty. True negatives (TN) are the fault-free modules correctly labeled as such. False negatives (FN) refer to faulty modules incorrectly classified as fault-free modules.

Table I
Matrix of Prediction

Prediction	Data		
		Fault	No Fault
	Fault	TP	FP
	No Fault	FN	TN

The following set of evaluation measures are being used to find the results:

1. **Mean Absolute Error** is a quantity used to measure how close forecasts or predictions are to the eventual outcomes.
2. **Root Mean Square Error** is a quadratic scoring rule which measures the average magnitude of the error. The difference between forecast and corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken.
3. **Accuracy:** It indicates proximity of measurement results to the true value, precision to the repeatability or reproducibility of the measurement.

The accuracy is the proportion of true results (both true positives and true negatives) in the population. The Mean Absolute Error and the Root Mean Square Error can be used together to diagnose the variation in the errors in a set of forecasts. The Root Mean Square Error will always be larger or equal to the Mean Absolute Error; the greater difference between them, the greater the variance in the individual errors in the sample. If the Root Mean Square Error = Mean Absolute Error, then all the errors are of the same magnitude. Both the Mean Absolute Error and Root Mean Square Error can range from 0 to ∞. They are negatively-oriented scores: Lower values are better.

IV. RESULTS

During prediction the True positives (TP) is calculated as 18, means 18 modules are correctly classified as faulty modules. False positives (FP) calculated as 28, means 28 fault-free modules incorrectly labeled as faulty. True negatives (TN) is calculated as 228, means 228 modules are the fault-free modules correctly labeled as such and False negatives (FN) comes out to be 4, means 4 faulty modules incorrectly classified as fault-free modules. These values are recorded in confusion matrix as shown in Table II.

Table II
Recorded Matrix of Prediction

Prediction	Data		
		Fault	No Fault
	Fault	18	28
	No Fault	4	228

The Root Mean Square Error and Mean Absolute Error are thus calculated as 0.3393 and 0.1151 respectively while the accuracy of prediction is calculated as 88.49%.

V. CONCLUSION

This paper empirically evaluates performance of Fuzzy Clustering technique in predicting fault-prone classes in large industrial software. Here, the System generated from fault data using Fuzzy Clustering in MATLAB 7.4 environment is evaluated for the JEdit testing dataset. The proposed Fuzzy C Means Clustering based prediction technique shows the results are 88.49% percent Accuracy. This study confirms that construction of Fuzzy C Means Clustering based model is feasible and useful in predicting faulty prone classes. It is therefore concluded that, in case of large software systems, model is implemented using Fuzzy C Means Clustering based technique for classification of the software components into faulty/fault-free systems is found satisfactory. The contributions of the study can be summarized as follows: First large software systems analyzed. These systems are developed with different development methods than proprietary software. In previous studies mostly proprietary software were analyzed. Second, we examine Fuzzy clustering method to predict the faulty classes with better accuracy. The future work can be extended in following directions:

1. Most important attribute can be found for fault prediction and this work can be extended to further programming languages.
2. More algorithms can be evaluated and then we can find the best algorithm. We plan to replicate our study to predict model based on hybrid genetic algorithms or soft computing techniques.

REFERENCES

[1] Koru, H. Liu, "Building effective defect- prediction models in practice", IEEE Software, 2005, 23-29.
[2] James, C. et al, " FUZZY C MEANS : The Fuzzy C-Means Clustering Algorithm", Computers & Geosciences Vol. 10, No. 2-3, 1984. 191-203.
[3] Musa, J. D., Iannino, A., and Okumoto, K., Software Reliability: Measurement, Prediction, Application, McGraw-Hill, (1987).
[4] Kaner, C., Software Engineering Metrics: What do they Measure and How do we Know? 10th International Software Metrics Symposium, METRICS, (2004).
[5] Pham, H., System Software Reliability, Reliability Engineering Series, Springer, (2006).
[6] Agresti, W. W., and Evanco, W. M., Projecting Software Defect form Analyzing Ada Design, *IEEE Trans. On Software Eng.,* **18**, (11), (1992), 988-997.
[7] Yu, T. J., Shen. V. Y., and Dunsmore, H. E., An Analysis of Several Software Defect Models, *IEEE Trans. on Software Eng.,* **14**, (9), (1988), 261-270.
[8] Khoshgoftaar, T. M., and Munson, J. C., Predicting Software Development Errors Using Complexity Metrics, *IEEE Journal on Selected Areas in Comm.,* **8**, (2), (1990), 253-261.
[9] Gaffney, J. E., and Davis, C. F., An Approach to Estimating Software Errors and Availability, SPC-TR-88-007, Version 1.0, March 1988, Proc. 11th Minnow Brook Workshop on Software Reliability, (1988).
[10] Gaffney, J. E., and Pietrolewiez, J., An Automated Model for Software Early Error Prediction (SWEEP), Proc. 13th Minnow Brook Workshop on Software Reliability, (1990).
[11] Rome Laboratory (RL), Methodology for Software Reliability Prediction and Assessment, Technical Report RL-TR-92-52, **1 & 2**, (1992).
[12] Zhang, X., and Pham, H., An Analysis of Factors Affecting Software Reliability, *The Journal of Systems and Software,* **50**, (1), (2000), 43-56.
[13] Li, M., and Smidts, C., A Ranking of Software Engineering Measures Based on Expert Opinion, *IEEE Trans. On Software Eng.,* **29**, (9), (2003), 811-24.
[14] Kumar, K. S., and Misra, R. B., An Enhanced Model for Early Software Reliability Prediction using Software Engineering Metrics, Proc. 2nd Int'l Conf. on Secure System Integration and Reliability Improvement, (2008), 177-178.
[15] Paulk, M. C., Weber, C. V., Curtis, B., and Chrissis, M. B., Capability Maturity Model Version 1.1, IEEE Software, 10, (3), (1993), 18-27.
[16] Fenton, N., Software Metrics-A Rigorous Approach, Chapmann & Hall, London, (1991).
[17] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl, S.J. Aud, Application of neural networks to software quality modeling of a very large telecommunications system, IEEE Transactions on Neural Networks, 8(4), 1997, pp. 902-909.
[18] Simranjit Kaur, Manish Mahajan, and Dr. Parvinder S. Sandhu, Identification of Fault Prone Modules in Source Software Systems using Hierarchical based Clustering, ISEMS, Bangkok, July 2011 ISBN:978-81-921733-1-3 (online).
[19] Promise. http://promisedata.org/repository/.
[20] Website sourceforge: www.sourceforge.net/ projects/jedit.
[21] http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/cmeans.html