Research Journal of Applied Sciences, Engineering and Technology 10(7): 831-840, 2015
DOI:10.19026/rjaset.10.2437
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2015 Maxwell Scientific Publication Corp.
Submitted: February 26, 2015 Accepted: March 25, 2015 Published: July 10, 2015

Research Article

A Comparative Study of Different Software Fault Prediction and Classification Techniques

[1]C.D. Rajaganapathy and [2]A. Subramani
[1]Department of Computer Science, PERI Institute of Technology, Chennai, Tamilnadu, India
[2]Department of MCA, KSR College of Engineering, Tiruchengode, Tamilnadu, India

Abstract: The main aim of this study is to survey about various techniques of fault prediction, clustering and classification to identify the defects in software modules. A software system consists of various modules and any of these modules can contain the fault that harmfully affects the reliability of the system. But early predictions of faulty modules can help in producing fault free software. So, it is better to classify modules as faulty or non-faulty after completing the coding. Then, more efforts can be put on the faulty modules to produce a reliable software. A fault is a defect or error in a source code that causes failures when executed. A faulty software module is the one containing number of faults, which causes software failure in an executable product. A software module is a set of functionally related source code files based on the system's architecture. Fault data can be collected from problem reporting system based on the module level. Defect prediction is particularly important in the field of software quality and reliability. Accurate prediction of faulty modules enables the verification and validation activities focused on the critical software components. A software quality classification model predicts the risk factor for software modules, which is an effective tool for targeting timely quality improvement actions. A desired classification technique provides better classification accuracy and robustness. This study surveys various fault prediction, clustering and classification techniques in order to identify the defects in software modules.

Keywords: Bayesian classification, Expectation Maximization (EM), Fuzzy C-Means (FCM) clustering, Hyper Quad Tree (HQT), k-means clustering, Similarity-based Software Clustering (SISC), spiral life cycle model, Support Vector classification (SVM)

INTRODUCTION

Software fault prediction (Karpagavadivu *et al.*, 2012) method is used to enhance the quality of the software and to assist software inspection by locating possible faults. It is a major part of software quality assurance, which is very popular and essential concept for researchers within the software engineering community. The software quality prediction is performed by identifying the prediction of module as faulty or non-faulty. Faults are major problem in software systems that needs to be resolved. A software fault or error refers a defect in a system. The major classes of software faults are shown in below:

- Syntactic faults
- Semantic faults
- Service faults
- Communication faults
- Exceptions

The major problem of fault prediction (Catal, 2011; Hall *et al.*, 2012) is finding the relationship between the modules in the software. There are many software defect prediction and classification methods that are available to detect and isolate faults. Each approach has their benefits and limitations. This study presents the survey on software fault prediction (Rohit Mahajan *et al.*, 2014; Gao *et al.*, 2011) and classification models. The spiral life cycle model is a type of iterative software development model, which is generally used in high risk projects such as, defect prediction and fault classification. Under the spiral life cycle model, identifying the faulty modules early in an iteration leads to a more reliable prototype. The high reliability of each iteration translates into a highly reliable product.

The spiral model (Hashmi and Jongmoon, 2007) has four quadrants, which includes, object determination, risk identification, product development and next phase planning. The first phase begins with the identification of the product objectives and functionalities. The next step deals with evaluating the alternative relative to the objectives and constraints. The third quadrant follows the waterfall model to incorporate the further incremental development. Finally, planning for the next phase starts after the end of this incremental model. Quality is built into the spiral

Corresponding Author: C.D. Rajaganapathy, Department of Computer Science, PERI Institute of Technology, Chennai, Tamilnadu, India

model by means of activities involved in each phase, like risk analysis, development plan, validation, verification and acceptance testing. Moreover, the development phase of spiral performs step by step analysis of the product, which ensures that no faults are escaped. In this model, the Dependent and Independent (DID) modules are identified based on their functionality. A data detection is the quantifiable expression of a rule and the rules can be detected in the source code by using the fragments of the data. The process of data detection includes, standardization, data centering and whitening.

Classification (Lessmann *et al*., 2008) is a process of finding a set of models that describe and distinguish data classes or concepts. The derived model is represented in various forms such as classification rules, decision tree etc. In software engineering, there are many studies representing the use of Bayesian classification to solve different challenges. Bayesian classifiers produce probabilities for class assignments, rather than a single definite classification. It has been surveyed in this study to predict the software prone modules at an early stage with the help of probability distribution models. Bayesian classifiers provide better reliability, when compare with the existing methods like logistic regression, Support Vector Machine (SVM) and classification trees.

Clustering is a non-hierarchical procedure in which items are moved among sets of clusters until the desired set is reached. Each clustering technique makes some assumptions about the underlying dataset. It is hard to satisfy all the assumptions, so it is beneficial to apply different clustering methods on the same dataset. Similarity-based Soft Clustering (SISC) is a clustering technique that is based on the similarity function given. SISC is similar to other clustering techniques, such as K-Means, Fuzzy C-Means. It starts with a carefully selected set of initial cluster and uses an iterative approach to improve the clusters. This approach only requires a similarity function to be defined properly and does not rely on any underlying probability assumptions. SISC is able to run faster than the traditional hard clustering algorithms. This technique is robust against outliers and it is also able to find clusters that hard clustering algorithms cannot able to find.

A software system may have defects in the software module and the defect may affect the reliability of the software. So, the defects must be predicted in order to improve the quality of the software system. This survey study depicts about various techniques for the identification of defects in software modules.

SOFTWARE FAULT PREDICTION AND CLASSIFICATION TECHNIQUES

The software fault prediction and classification comprises of following stages:

- Software fault prediction
- Software reliability enhancement
- Defect classification
- Clustering

Software fault prediction techniques: Software fault prediction (Rawat and Dubey, 2012) is the process of classifying software modules into fault prone and non-fault prone. Early detection of fault prone module enables verification experts to concentrate their time and resources on the problem areas of the system under development. Due to some faulty modules, the maintenance phase of software products could become really painful for the users and costly for the enterprises. This section presents some of the software fault prediction techniques. The techniques are, Spiral model life cycle, Hyper Quad Tree (HQT) and the EM algorithm, Ripple Down Rule (RIDOR), metric based approaches and hybrid feature selection method.

Spiral Life Cycle model (SDLC): The spiral life cycle model is related to the incremental model in Software Development Life Cycle (SDLC) (Ruparelia, 2010), with more emphases placed on risk analysis. The spiral model has four phases, planning, risk analysis, engineering and evaluation. A software project repeatedly passes through these phases in iterations (called spirals) (Madachy *et al*., 2006). Each iteration of a spiral life cycle model produces a prototype system that is more suitable for operational testing. With the help of spiral life cycle model, the software fault prone modules are predicted at early stage, which improves better system reliability. Discriminant analysis can be a useful tool in identification of fault modules in tactical systems. A key benefit of the spiral model is that it attempts to contain project risks and costs at the outset. It has more advantages compared than the other models. The spiral life cycle model is one of the most flexible model in SDLC. It is more suitable for high risk and mission critical software projects, where business needs may be unstable. An important feature of the spiral model is that each cycle is completed by a review involving the primary people or organizations concerned with the product. A highly customized product can be developed using life cycle and it also used for high amount risk analysis. The spiral model is characterized by iterative development of evolutionary prototypes, which is more suitable for operational testing. It creates a risk driven approach to the software process rather than a document driven or code driven process. It combines the strengths of the other models and resolves many of their difficulties and problems. The advantages of this model is that its range of options provides the good features, while its risk driven approach avoids many of their difficulties.

Hyper Quad Tree (HQT) and Expectation-Maximization (EM) algorithm: Quad Tree (4-ary tree) is the recursive data structure, this tree stands for a division of the matrix into sub matrices (nodes). Leafs of the QT are classified into complete and blank nodes. The QT (Bishnu and Bhattacherjee, 2012) based method assigns the suitable initial cluster centers and eliminates the outliers. It has some general features, it decomposes the space into adaptable cells in which each cell has a high capacity. Hyper quad tree (Sasidharan and Sriram, 2014) works in *n*-dimensions, hence it finds better initial cluster centers than former algorithms. This algorithm is mainly used to predict the software faults in a given module. It offers a better cluster center and lower fault ratio in a given dataset. The HQT is a universal quad tree, which represents a total recursive division of the *n*-dimensional vector space. Every inner node of HQT includes a covering hyper-quad and 2*n* links to all its sub hyper-quad. Finally, it divides the regions recursively, so that no region contains more than one data point. The Expectation Maximization (EM) (Meenakshi *et al*., 2012) algorithm is a popular iterative refinement technique that can be used for finding parameter estimation. HQT based EM algorithm (Rawat and Dubey, 2012) is known to be an appropriate optimization for finding compact clusters. It guarantees an elegant convergence and assigns an object to a cluster based on the probability of membership functions. After that, it iteratively rescores the objects and updates the estimates. The accuracy of fault prediction is enhanced using the HQT based EM (Varade and Ingle, 2013) algorithm. However, this algorithm also has some limitation. The user has to initialize the number of clusters, which is very difficult to identify using HQT based EM algorithm. It is not providing the exact centroid.

Ripple Down Rule (RIDOR): Ripple Down Rules (RIDOR) (Najadat and Alsmadi, 2012) is a knowledge acquisition method, which controls the communications between the expert and a shell to attain the correct knowledge. Multiple Classification Ripple Down Rules is an extension of RIDOR, which provides a basis for solving a general classification problems by using beyond classification. This approach does not use any notion for extracting or mining the expert's knowledge. The RIDOR inference operation is based on searching the knowledge base as a decision list. It also shifts the development emphasis to maintenance by blurring the dissimilarity between the initial development and maintenance. This algorithm learns defect prediction using mining static code attributes, which is used to predict the faults with high accuracy and low error rate. This approach used the rule based classification method to classify the modules from their fault prone. The main intention of this method is to enhance the software development process and effectively allocate the resources. The limitation of RIDOR is that the knowledge base is ill structured, which results a repetition of knowledge. However, this could exponentially increase the knowledge acquisition task. In order to overcome these drawbacks, the spiral development life cycle model is developed.

Metric based approaches: Faults and failures are the cost factors in software, which describes the significant amount of any project budget. This information can be used as a feedback to the enhancement of the development process. It also used for process improvement and cost reduction. Based on these reasons, it is clear that the methods (Radjenović *et al*., 2013) are needed to enhance, control and predict fault handling in general. This type of methods is categorized into two major classes: methods for predicting the number of faults in a specific module and methods for identifying the fault prone modules. The first type of methods are problematic to develop a valid model, which is transferable between projects or organizations. Thus, methods for identification of fault and failure prone modules and models for prediction are a potential way to enhance software quality and to diminish cost. Effective defect prediction models help to enhance the quality assurance activities on defect prone modules. Machine learning approaches are used to predict the probability of fault proneness. The dependent variable is predicted based on the faults found during the software development life cycle. Both level metrics and class metric methods are used to predict the defects. Level metrics are suitable for both procedural and object oriented programs and class metrics are only suitable for object oriented programs. The diversity of these metrics (Catal, 2012; Shanthini and Chandrasekaran, 2012) inhibits the progress that often results from focusing on one simple target. It requires a long term commitment, which are the disadvantages of this method. In order to overcome these limitations a spiral life cycle model is developed.

Software reliability enhancement: Software reliability (Lyu, 2007) can be enhanced through extensive testing and debugging. Reliable software is compulsory, complex mission critical systems.

Decision tree and fuzzy logic: Decision trees (Pandey and Goyal, 2010) are great and standard tools for classification and prediction. It produces classifiers in a form of tree structure, where each leaf node illustrates a decision node. In this method (Sehgal *et al*., 2012), classification starts from the root and continues to move down until the leaf node is reached. It helps to classify the faulty and non-faulty modules in software. In fuzzy decision tree, each path from the root node to a terminal node corresponds to a fuzzy rule. Generally, the decision tree technique is used for inductive learning and it is used to feature subset selection process in the software cost estimation model. ID3, C4.5 and CART methods are widely for constructing a decision tree. This method is weak in handling uncertainty and fuzziness, which are the drawbacks of fuzzy decision tree. ID3 (Elyassami and Idri, 2011) is a type of decision tree method for software effort estimation, which is designed by incorporating the concepts of fuzzy set. C4.5 (Wang *et al*., 2012) is also a type of decision tree, which is used to build a decision tree from a set of training data by using information entropy. The disadvantage of this method is the module selection and the distribution of defects tends to one or two defects in one file. In order to overcome these drawbacks, the spiral life cycle model is enhanced.

Radial Basis Function (RBF): The software reliability and quality are enhanced by using Radial Basis Function (RBF) (Buchtala *et al*., 2005), which is the best approach to identify the software faults. Initially, the data is split into clusters using fuzzy subtractive clustering after that, the RBF is applied to predict the faults. It is a real valued function, whose values depends on the distance from its respective field center. RBF provides a flexible way to generalize linear regression function, which possess strong mathematical properties of best approximation. This model can be viewed as a realization of a sequence of two mappings. The first method is a nonlinear mapping of the input data and the second method is a linear mapping of the basis function output. The selection of RBF centers is difficult, which are not guaranteed to capture the structure of information that is the limitation of this method.

Defect classification: The classification perspective determines the information extracted from the fault classification. The main intention of fault classification is to identify the faults correctly and the key issue is to enhance the process based on the faulty information. It is necessary to monitor the agreement between the clusters to assure the correctness in the classification. A classifier performs the fault placement process into a certain classification category. This section presents some techniques to classify the faults in software.

Bayesian classification: Bayesian classification (Catal *et al*., 2011) provides a natural statistical framework for decision making by using the software utilities. Their representation of causal relationships among variables are meaningful to software practitioners, which is based on the Bayes theorem. It allows to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. This technique (Tosun Misirli and Basar Bener, 2014) can predict class membership probabilities, such as the probability of a given tuple belongs to a particular class. It can be exploited to support effective decision making for improving the software process. This classification method (Mahajan *et al*., 2012) is used to deduce the probability distribution for a target variables (i.e., defect detected). This model is feasible, adaptable to object oriented systems and useful to predict the faulty prone classes in software. This technique successfully classifies the software components into faulty and fault free. It offers the following advantages, it maintains the observations, statistical distribution and prior assumptions. It encodes the causal relationships among variables to predict the future actions. It is necessary to use a suitable fault classification technique to handle the software faults.

Logistic regression: Logistic Regression (Reddy and Babu, 2013) is a type of statistical classification model, which is used to predict a binary response from a binary predictor. It measures the relationship between a dependent and independent variables, which are commonly continuous. This technique uses class level metrics for the software fault prediction that is based on the statistical approach. A negative binomial regression model is developed to identify the number of faults on each file of the system. It provides the number of faults for each file of a release based on a characteristics, such as, the file size, number of faults, programming language and age of the file. The logistic regression consists of two models, namely, discrete and conventional. The parameter estimation in the logistic curve models reproduced the values of the parameters very accurately. The parameter estimates of the continuous model vary with the number of data points. The discrete model proves the stable values for various numbers of data points. This characteristic is very essential for the software reliability enhancement. Logistic regression offers an easier interpretation compared to other classification techniques. It provides low design quality, which is the drawback of this technique. The mentioned drawback of this method is overwhelmed by enhancing the Bayesian classification technique.

Support Vector Machine (SVM) classification: SVM is a classification technique, which has been successfully applied for solving classification and regression analysis. It is adaptive to model nonlinear functional relationships that are difficult to model with other techniques. It provides nonlinear function approximations by nonlinearly mapping input vectors into feature spaces. This method (Xing *et al*., 2005) combines the advantages of linear and non-linear methods by embedding the data into a feature space. It uses machine learning techniques and method level metrics. It is robust in nature than other techniques for software quality prediction. SVM provides the best prediction performance in terms of precision, recall and accuracy. The results indicate that the performance of SVM (Elish and Elish, 2008) is better than the other classification methods. But, the disadvantage is that it does not work well in public data sets. The mentioned drawback is overwhelmed by enhancing the Bayesian classification method.

Classification trees: Classification tree is a popular approach for software defect prediction, which are based on the statistical based approach. It involves the process of modules categorization represented by a set of software metrics or code attributes into faulty and non-faulty modules. This method (Catal and Diri, 2009) uses two different types of datasets, namely, JM1 and KC1. Each dataset includes various software modules together with their number of faults and characteristic code attributes. The accuracy metric is not suitable for software fault prediction studies because, imbalanced datasets cannot be evaluated with this metric, which is the drawback of this metric. But, the accuracy metric of the Bayesian classification method is suitable for predicting software faults.

Clustering: Clustering is defined as the classification of data or object into diverse groups. It is the process of partitioning the data set into diverse subsets. This sector presents some of the clustering algorithms for the prediction of software faults. The clustering methods are, SISC, K-Means clustering and Fuzzy C-Means clustering.

Similarity-based Soft Clustering (SISC): Similarity-based Soft Clustering (SISC) (Shanthini and Chandrasekaran, 2012) is a clustering techniques based on the similarity function given. It aims to provide a soft clustering on a set of documents based on a similarity measure. In this model, the documents can be clustered into multiple clusters. SISC is able to execute the codes in an efficient manner and it provides security against outliers. In this technique, the software metrics are used as independent variables and fault data is used as dependent variables. This clustering method is an unsupervised learning approach, which is used to group the modules having similar metrics by using similarity measures or distances. After clustering, the mean values of each the software metrics within clusters can be checked against the metric threshold values. If the limits are exceeded, the cluster can be labeled as fault prone. After that, the evaluation parameters are used to evaluate the performance of the clustering process. Hence, False Positive Rate (FPR), False Negative Rate (FNR) and the error values are calculated based on the outcomes.

K-means clustering: K-Means clustering (Hribar and Duka, 2010) is a non-hierarchical clustering procedure in which items are moved among sets of clusters until the desired set is reached. This technique follows a partitional clustering approach in which partitioning method creates an initial partitioning. After that, it uses an iterative relocation technique that attempts to enhance partitioning by moving objects from one group to another. This algorithm begins with finding the initial centroids for potential clusters i.e., each cluster is associated with a centroid. The observation is assigned to each cluster based on their distance from the centroid. When partitioning the data set the sum of the intra-cluster distance is diminished to an optimum value. This algorithm reassigns and executes the data points until the convergence criterion is met. The K-Means algorithm (Sandhu *et al*., 2010) has some general properties, every member of a cluster is closer to its cluster than any other cluster. There are always K-clusters and it has at least one item in each cluster, which are non-hierarchical. However, the K-Means algorithm also has some drawbacks. The user has to initialize the number of clusters, which is very difficult to identify in most of the cases. It requires the selection of the suitable initial cluster centers, which is again subject to error. The structure of the cluster depends on the initial cluster center this may result in an inefficient clustering. K-Means algorithm is very sensitive to noise. In order to overcome these drawbacks, the Similarity-based Soft Clustering (SISC) technique is developed.

Fuzzy C-Means clustering (FCM): Fuzzy C-Means (FCM) (Bisht *et al*., 2012) iteratively moves the cluster centers to the right location within a data set. It is the fuzzified version of the k-means clustering algorithm, which allows one piece of data to two or more clusters. This method iteratively moves the cluster centers to the right location within a data set by updating the cluster centers and the membership grades for each data point. It plays an essential role in solving problems in many areas including fuzzy intelligent control, pattern classification and fault pattern classification. FCM is mostly used algorithm for classifying faults and pattern classification. In this method, it assigns a data point to distinct cluster and membership values to each observation in all derived clusters. After clustering, the fuzzy model is constructed based on the clustered data. The validity measures are scalar indices that assess the goodness of the obtained partition. Hence, this measure is designed to quantify the separation and compactness of the clusters. FCM (Yang *et al*., 2011) is used to improve the software process control and attain high software reliability, when predicting the faults in software. However, this technique also has the disadvantage that the number of fuzzy sets must be informed. The mentioned drawback is overwhelmed by developing the SISC method.

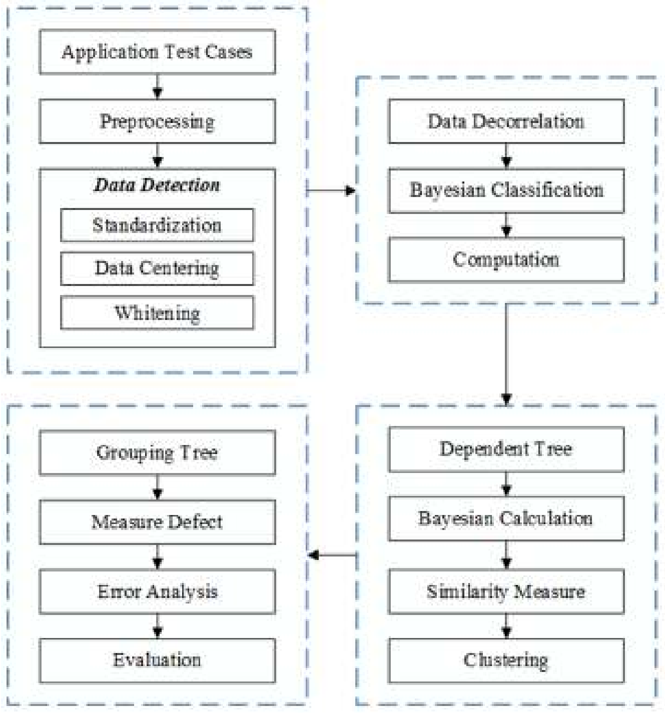

Fig. 1: Software fault prediction using SDLC

PROPOSED METHODOLOGY

The proposed software defect prediction methodology will use the spiral life cycle model to predict the faults. This method is used to improve the quality of the software and avoid building an error prone modules in future. Figure 1 shows the overall flow of the proposed software fault prediction model. Initially, the Dependent and Independent (DID) modules are identified in this model based on their functionality. The standardization, data centering and whitening processes are performed for faulty data detection. Then, the Bayesian classification algorithm is used to classify the faulty and non-faulty modules in software. Moreover, the SISC method is proposed to cluster the similar data based on the similarity measure. The performance comparison of a software fault prediction will be done by using the SISC method. The proposed system accurately predicts and classifies the software faults to improve the system reliability and quality.

RESULTS AND DISCUSSION

Various techniques for software fault prediction and classification are illustrated. The results of this survey are shown in Table 1. The defect prediction with the Bayesian classification enhances the software reliability and quality. From the survey, it is evident that the Spiral life cycle model and Bayesian classification provide, the better fault prediction results compared than the existing methods such as, HQT-EM, RIDOR and Metric based approaches. Also, the Bayesian classification classifies the faulty and non-faulty modules effectual compared than the existing methodologies, such as, logistic regression, SVM and classification trees. Moreover, the SISC method performs well than the other clustering methods such as, K-Means and FCM.

Summary: From this survey it is observed that, the existing software fault prediction models provides inappropriate risk management decisions. It does not effectively take dependencies between the attributes into consideration. Thus, we use a SDLC method to improve the performance of the fault prediction results. The existing classification methods are very difficult to understand and it does not provide the exact classification result. Some of the disadvantages of existing classification algorithms are its memory dependency, computational complexity and large computational time. The above mentioned drawbacks are overwhelmed in this study by using a Bayesian classification method. The existing clustering techniques also has some drawbacks, such as, the user has to initialize the number of clusters, which is very difficult to identify. It needs to select the suitable initial cluster centers and it is very sensitive to noise. Moreover, it does not improve the quality of clustering in an efficient manner. In order to avoid these drawbacks, the SISC clustering technique is proposed in this study. The SISC method provides the best

Table 1: Information about different fault prediction and classification techniques

Techniques	Author and reference	Year	Performance	Quality measurement
Software fault prediction methods				
Spiral life cycle model	Ruparelia (2010)	2010	It splits the SDLC models into three broad categories such as, linear, iterative and combination of both. It provides a visual interface to an end user.	• Drive iterations • Requirements management • Visual models • Control changes • Customization
	Madachy *et al*. (2006)	2006	The spiral model is extended to address the problems of software intensive systems. It uses three specialized teams to estimate the cost and schedule for hybrid process.	• Simulation inputs • System response to volatility • Tradeoff functions • Parameterization
Hyper Quad Tree (HQT) -Expectation Maximization (EM)	Bishnu and Bhattacherjee (2012)	2012	It evaluates the effectiveness of QT based K-means clustering algorithm to predict faulty software modules.	• AR3, AR4 and AR5 dataset • False Positive Rate (FPR) • False Negative Rate (FNR) • Error value
	Varade and Ingle (2013)	2013	It proves that the results of hyper-quad tree is more accurate than QT and it overcomes the weakness of K-means algorithm.	• AR1, AR3, AR4, AR5 and AR6 datasets • False Positive Rate (FPR) • False Negative Rate (FNR) • Error rate
	Meenakshi *et al*. (2012)	2012	It improves the accuracy of fault prediction by using EM algorithm. It proves that EM algorithm is more accurate than K-means owing to lower error rate.	• Incorrectly classified species • Correctly classified species • Error rate • Label prediction
Ripple Down Rule (RIDOR)	Najadat and Alsmadi (2012)	2012	This system classifies the software modules into faulty and non-faulty prone. It learns defect prediction using mining static code attributes.	• PC1, PC2, PC3, PC4, CM1, MW1, KC3 and KC4 datasets • Size • Faulty module number • % of faulty modules • Metric number • Accuracy
Metric based approaches	Radjenović *et al*. (2013)	2013	It identifies and depicts the current state-of-the-art software metrics to assess their applicability in software fault prediction.	• Dataset P2 to P10 • Fault classification • Fault ranking • Dependent variable granularity • Object oriented metrics
	Shanthini and Chandrasekaran (2012)	2012	This study investigates the significance of various software metrics in order to predict the defects. It analyzes the performance of various classifiers to predict faults based on public domain such as NASA and KC1 dataset.	• KC1 and NASA dataset • Precision • Recall • F-measure • Accuracy • True positive rate
Reliability enhancement				
Decision tree and fuzzy logic	Huang *et al*. (2006)	2006	It embedding risk assessment information into software cost estimation model by fuzzy decision tree approach.	• Fuzzy the software cost drivers • Risk assessment model construction • Risk estimation • Prediction Accuracy Rate (PRED) • Mean Magnitude of Relative Error (MMRE)
	Sehgal *et al*. (2012)	2012	This technique helps project manager to make efficient use of limited resources to target those modules that are defected.	• PC1 and NASA dataset • Metrics Data Program (MDP) • Binary splits • Confidence factor • Reduced error pruning
	Elyassami and Idri (2011)	2011	This study investigates the use of fuzzy ID3 decision tree for software cost estimation. It handles uncertain and imprecise data, when describing the software projects.	• Tukutuku and COCOMO'81 datasets • Magnitude of Relative Error (MRE) • Mean Magnitude of Relative Error (MMRE) • Prediction rate • Significant Level (SL) value
Radial Basis Function (RBF)	Buchtala *et al*. (2005)	2005	Evolutionary Algorithms (EA) performs feature and model selection process simultaneously for RBF.	• ID dataset • Signature verification • Lift factors • Process optimization • Training time
Defect classification				

Table 1: Continue

Techniques	Author and reference	Year	Performance	Quality measurement
Bayesian classification	Mahajan *et al*. (2012)	2012	It provides a best way to support software quality through improved scheduling and project control.	• Mean Absolute Error (MAE) • Root Mean-Squared Error (RMSE) • Probability of detection • Probability of false alarms
Logistic regression	Reddy and Babu (2013)	2013	It provides an additional decision making rule to the software developers, when they managing the software resources.	• PL/I database software • Mean squared error • Discovery of errors STS2, STS3 and STS4
Support Vector Machine (SVM)	Xing *et al*. (2005)	2005	This study evaluates the software quality level and indicates the software quality problems in early stage. It performs well even in high dimensional spaces under small training sample conditions.	• Eigen value • Kernel function • T1ERR, T2ERR (error types) • Quality Discriminant Analysis (QDA)
	Elish and Elish (2008)	2008	This analysis evaluates the capability of SVM to predict software defect prone modules and compares its prediction performance against statistical and machine learning models.	• NASA dataset • Prediction rate • Defect prone modules
Classification trees	Catal and Diri (2009)	2009	This technique investigates the class level metrics to predict faults during design phase.	• RQ1, RQ2, RQ3 and RQ4 dataset • Distribution metrics (class, quantitative values and component) • Statistics machine learning
Clustering				
Similarity based Soft Clustering (SISC)	Kanimozhi and Balakrishnan (2014)	2014	This clustering method is used to group the test cases based on the similarity values to each cluster.	• Test cases • Fault detection rate • Redundancy level • Execution time
K-means clustering	Hribar and Duka (2010)	2010	This method predicts the weibull distribution parameters shape, slope and total number of faults in the system based on the software components.	• Weibull distribution • Prediction rate • Beta prediction • Accuracy rate
	Varade and Ingle (2013)	2012	It allocates the centroids to each cluster in a cunning way, because different location causes various results.	• Public dataset • False Positive Ratio (FPR) • False Negative Ratio (FNR) • Error rate • Precision
	Sandhu *et al*. (2010)	2010	This technique determines the intrinsic grouping in a set of unlabeled data. It is used to find the faulty modules in an open source software systems.	• Public dataset • Threshold value • Accuracy of prediction • Probability of detection • Probability of false alarms
Fuzzy C-Means (FCM) clustering	Bisht *et al*. (2012)	2012	It produces an optimal c partition by minimizing the sum of squared error objective function.	• Accuracy • Probability of detection • Probability of false alarms • MAE value • RMSE value
	Yang *et al*. (2011)	2011	This method detects and isolates the faults in order to avoid overall failure of the system, which includes the process of feature extraction, selection and classification.	• Partition co-efficient • Partition-entropy • Clustering result • Precision of fault diagnosis

clustering results, when compared to the k-means and FCM clustering algorithms.

CONCLUSION

In this study, an overview of various software fault prediction and classification methods are presented. From the survey, it is finding out that the spiral life cycle model and Bayesian classification are the very powerful fault detection techniques to accurately predict and classify the software defects. When, combined with the SISC clustering method, it improves the reliability and quality of the system. The best software defect prediction and classification techniques can be framed based on the spiral life cycle model, Bayesian classification and SISC to achieve the best system reliability.

REFERENCES

Bishnu, P.S. and V. Bhattacherjee, 2012. Software fault prediction using quad tree-based k-means clustering algorithm. IEEE T. Knowl. Data En., 24(6): 1146-1150.

Bisht, A., A.S. Brar and P.S. Sandhu, 2012. Prediction of faults in open source software systems using FCM. Proceeding of the International Conference on Computer Graphics, Simulation and Modeling.

Buchtala, O., M. Klimek and B. Sick, 2005. Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE T. Syst. Man Cy. B, 35(5): 928-947.

Catal, C., 2011. Software fault prediction: A literature review and current trends. Expert Syst. Appl., 38(4): 4626-4636.

Catal, C., 2012. Performance evaluation metrics for software fault prediction studies. Acta Polytech. Hung., 9(4): 193-206.

Catal, C. and B. Diri, 2009. A systematic review of software fault prediction studies. Expert Syst. Appl., 36(4): 7346-7354.

Catal, C., U. Sevim and B. Diri, 2011. Practical development of an eclipse-based software fault prediction tool using naive bayes algorithm. Expert Syst. Appl., 38(3): 2347-2353.

Elish, K.O. and M.O. Elish, 2008. Predicting defect-prone software modules using support vector machines. J. Syst. Software, 81(5): 649-660.

Elyassami, S. and A. Idri, 2011. Applying fuzzy ID3 decision tree for software effort estimation. Int. J. Comp. Sci. Issues, 8(4): 131-138.

Gao, K., T.M. Khoshgoftaar, H. Wang and N. Seliya, 2011. Choosing software metrics for defect prediction: An investigation on feature selection techniques. Software Pract. Exper., 41(5): 579-606.

Hall, T., S. Beecham, D. Bowes, D. Gray and S. Counsell, 2012. A systematic literature review on fault prediction performance in software engineering. IEEE T. Software Eng., 38(6): 1276-1304.

Hashmi, S.I. and B. Jongmoon, 2007. Software quality assurance in XP and spiral: A comparative study. Proceeding of the ICCSA International Conference on Computational Science and its Applications, pp: 367-374.

Hribar, L. and D. Duka, 2010. Software component quality prediction using KNN and Fuzzy logic. Proceeding of the 33rd International Convention MIPRO, pp: 402-408.

Huang, S.J., C.Y. Lin and N.H. Chiu, 2006. Fuzzy decision tree approach for embedding risk assessment information into software cost estimation model. J. Inf. Sci. Eng., 22(2): 297-313.

Kanimozhi, R. and J. Balakrishnan, 2014. Cosine similarity based clustering for software testing using prioritization. J. Comput. Eng., 16(1): 75-80.

Karpagavadivu, K., T. Maragatham and S. Karthik, 2012. A survey of different software fault prediction using data mining techniques methods. Int. J. Adv. Res. Comput. Eng. Technol., 1(8).

Lessmann, S., B. Baesens, C. Mues and S. Pietsch, 2008. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE T. Software Eng., 34(4): 485-496.

Lyu, M.R., 2007. Software reliability engineering: A roadmap. Proceeding of the Future of Software Engineering (FOSE'07). Minneapolis, MN, pp. 153-170.

Madachy, R., B. Boehm and J.A. Lane, 2006. Spiral lifecycle increment modeling for new hybrid processes. In: Wang, Q. *et al*. (Eds.), SPW/ProSim, 2006. LNCS 3966, Springer, Berlin, Heidelberg, pp: 167-177.

Mahajan, A., V. Gupta and P.S. Sandhu, 2012. A bayes network classification approach for finding faulty modules in open source software systems. Int. J. Res. Eng. Technol., 1(1): 45-48.

Meenakshi, P.C., S. Meenu, M. Mithra and P. Leela Rani, 2012. Fault prediction using quad tree and expectation maximization algorithm. Int. J. Appl. Inf. Syst., 2(4): 36-40.

Najadat, H. and I. Alsmadi, 2012. Enhance rule based detection for software fault prone modules. Int. J. Softw. Eng. Appl., 6(1): 75-86.

Pandey, A.K. and N.K. Goyal, 2010. Predicting fault-prone software module using data mining technique and fuzzy logic. Int. J. Comput. Commun. Technol., 2: 2-4.

Radjenović, D., M. Heričko, R. Torkar and A. Živkovič, 2013. Software fault prediction metrics: A systematic literature review. Inform. Software Tech., 55(8): 1397-1418.

Rawat, M.S. and S.K. Dubey, 2012. Software defect prediction models for quality improvement: A literature study. Int. J. Comput. Sci. Issues, 9(5): 288.

Reddy, K.V.S. and B.R. Babu, 2013. Logistic regression approach to software reliability engineering with fault prediction. Int. J. Softw. Eng. Appl., 4(1): 55-65.

Rohit Mahajan, E., G. Sunil Kumar and R.K. Bedi, 2014. Comparison of various approaches of software fault prediction: A review. Int. J. Adv. Technol. Eng. Res., 4(4): 13-16.

Ruparelia, N.B., 2010. Software development lifecycle models. ACM SIGSOFT, 35(3): 8-13.

Sandhu, P.S., J. Singh, V. Gupta, M. Kaur, S. Manhas and R. Sidhu, 2010. A K-means based clustering approach for finding faulty modules in open source software systems. World Acad. Sci. Eng. Technol., 72: 654-658.

Sasidharan, R. and P. Sriram, 2014. Hyper-quadtree-based K-means algorithm for software fault prediction. Adv. Intell. Syst. Comput., 246: 107-118.

Sehgal, L., N. Mohan and P.S. Sandhu, 2012. Quality prediction of function based software using decision tree approach. Proceeding of the International Conference on Computer Engineering and Multimedia Technologies (ICCEMT), pp: 43-47.

Shanthini, A. and R.M. Chandrasekaran, 2012. Applying machine learning for fault prediction using software metrics. Int. J. Adv. Res. Comput. Sci. Softw. Eng., 2(6): 274-278.

Tosun Misirli, A. and A. Basar Bener, 2014. Bayesian networks for evidence-based decision-making in software engineering. IEEE T. Software. Eng., 40(6).

Varade, S. and M. Ingle, 2013. Hyper-quad-tree based K-means clustering algorithm for fault prediction. Int. J. Comput. Appl., 76(5): 6-10.

Wang, J., B. Shen and Y. Chen, 2012. Compressed C4. 5 models for software defect prediction. Proceeding of the 12th International Conference on Quality Software (QSIC), pp: 13-16.

Xing, F., P. Guo and M.R. Lyu, 2005. A novel method for early software quality prediction based on support vector machine. Proceeding of the 16th IEEE International Symposium on Software Reliability Engineering, pp: 10-222.

Yang, Q., J. Guo, D. Zhang and C. Liu, 2011. Fault diagnosis based on fuzzy C-means algorithm of the optimal number of clusters and probabilistic neural network. Int. J. Intell. Eng. Syst., 4(2): 51-59.