Software Defects Measurements based on Software Matrix

Dr. K. Rajeshwar Rao [1]

Prof, CSE Department, Siddhartha Institute of Institute of Engineering & Technology (SIEI), Ibrahimpatnam, Hyderabad

Abstract— *With the growing demand for high quality software in industries with lower time complexity and lower requirements for memory consumptions, the need for identifying the defects in the software cannot be ignored. The software development industry deploys huge manpower to estimate, detect and resolve the defects in the software to match the customer requirements and ensure the quality in the product. The process involves development and testing in iteration to detect the defects in the software. However the process of detection and resolving the defects is long resulting in various problems like mismatching the cost estimations in the upcoming development modules. Whereas, an automatic prediction technique based on the pre-generated software defect matrix can be deployed to correctly or nearly correct predictions to help in estimating the defects may raise in the upcoming modules. Hence in this work we try to investigate and understand the data mining techniques for defect prediction in software development. This work also results in modification notes to the existing processes by deploying a unique pre-processing technique for the defect matrix data using cumulative and normalized distribution of the initial data. The work will demonstrate the improvements in the existing works with the proposed pre-processing techniques.*

Keywords*— *Software Matrix, Defect Detection, Data Mining, Neural Network

I. INTRODUCTION

The software projects without a proper detection technique for defects in developed modules may lead to software full of problems and not generating desired output mentioned in the required specification by customer. The requirements are generated during the mutual discussion between client and the development companies in the initial phases. The software development companies are always liable to deliver the software with agreed performance. Hence the software companies put lot of efforts in detecting and resolving the defects in the software. However the detecting and fixing process of detects is time consuming process and ignoring the defects may lead to malfunctions ranging from losing a small penny to million dollar loss. Software development companies deal with defects which are known and predictable and sometimes unknown and unpredictable defects. The known defects can be deal with pre-planned development strategies and are generally less time consuming. The known defects will not disturb the cost and time estimations for the project. However the unknown defects are unpredictable, hence the resolution of those defects also cannot be pre-defined. Hence the development industries keep huge efforts to deploy multiple prediction techniques to detect unknown defects for the next modules from the existing defect matrix generated during the development phases. The early documents have demonstrated the use of software defect matrix to demonstrate the time complexity, memory requirements and development cost in terms of time to market. An in depth calculation steps are to be executed to determine the number of defects in the software module or the complete program. However the recent researches have demonstrated the use of software defect

matrix to form the guidelines for defect detection. The parallel researches have also demonstrated good classification techniques for defects matrices to focus on one specific objective and ignore the rest of the matrices for the same objective. The rest of the matrices can also be mapped to other objectives present in the process. Here it is to be understood that the objectives are concerned to quality assurance and defect resolution only.

More recent researches demonstrate the use of data mining techniques [1] [2] with the use of neural networks under the perceptron category for better results. However, the effectiveness of the neural network based algorithms depends on two major factors [3]. Firstly, the correct dataset for training the neural network algorithm or application and secondly, the input data in the correct format [4]. Hence the applications of neural network algorithms depend majorly on the data which needs to be in correct format for processing in training and processing phase [5]. Hence it is the demand of the current researches to develop an automated pre-processing technique to provide the correct data to the neural networks.

Hence in this work we propose a unique approach for pre-processing the training and input data before getting fed to the neural network for prediction. This work also demonstrate the applications of the pre-processing technique on existing neural network driven defect prediction approaches proposed in the early researches. The comparisons are also been carried out with the raw data parallel to the pre-processed data for the same.

The rest of the work is organized as in Section II we understand the types of software matrices used in development best practices, in Section III we analyse the strategies for defect predictions using Data Mining, in Section IV we understand the algorithms used for defect prediction using neural networks, in Section V we realise the pre-processing technique, in Section VI we analyse the results of pre-processing on the existing algorithms using neural networks, in Section VII we conclude with the glimpse of the future work to be carried out in continuation to this work.

II. SOFTWARE MATRICES

The software matrices are created for various purposes ranging from describing the characteristics of the software product to information related to up-gradation to staff information deployed into the project. Software matrices are classified into major three categories as Product Related Matrix, Process Related Matrix and finally the Project Related Matrix [6]. In this work we understand different types and subtypes of these matrices:

A. Product Related Matrix

The product related matrices focuses on the product quality and the level of customer satisfactions. The matrices are designed to keep the information related to number of defects occurred in the software and level of customer satisfaction. The predicted information about how long the software will continue executing before failure is also stored in the matrices. A total of four sub-classifications are available for use under Product Related Matrix for various different purposes. We understand their purpose and use here [7]:

1) Matrix for Defect Density: The matrix for software defect density identification is majorly used for storing information related to number of defects in each software development modules. The process of calculation is as following, considering the density for defect for any module is d_{m1} defined as:

$$d_{m1} = \frac{\sum_{i=1}^{n} d_i}{n}, \quad \text{…. Eq 1}$$

Where d_i denotes each defect in the module and n denotes the number of total number of lines in the software module. Based on the rate of defects the software product and the

development companies are assured with the Capability Maturity Model CMM. The calculation for Capability Maturity Model is demonstrated in Table 1.

TABLE I: CAPABILITY MATURITY MODEL MEASUREMENT

Defect Rate	**CMM Level can be achieved**
Below 0.05	CMM 5
Above 0.05 and Below 0.14	CMM 4
Above 0.14 and Below 0.27	CMM 3
Above 0.27 and Below 0.44	CMM 2
Above 0.44 and Below 0.75	CMM 1

2) Matrix for Customer Feedback: The software matrix for customer feedback contains the information related to the reports reported by the customers. This matrix helps to understand and predict the amount of present and future defects to be addressed for each product or customer. The calculation for the problems can be calculated as followings, considering the number of problems as $P_{U/M}$,

$$P_{U/M} = \frac{\sum_{m=1}^{l} DP_m + \sum_{m=1}^{l} NDP_m}{l}, \quad \text{.... Eq 2}$$

Where DP_m and NDP_m denote the number of defect problems & number of non-defect problems per months respectively and l denotes the number of months under license period.

3) Matrix for Customer Satisfaction: The matrix for customer satisfaction is based on the feedback data generated during the complete process of software development. The factors related to customer satisfaction are reliability of the software, responsive nature of the software, quality assurance of the software, applicable empathy of the software and tangibility of the software. The software development companies assign weightage for each factor considering the organizational and functional goals for software and each module inside that software [Table 2].

TABLE III: SATISFACTION WEIGHTAGE MAPPING

Factors for Satisfaction	**Assigned Weightage**	**Weightage for each Modules**			
		M 1	**M 2**	**M 3**	**M 4**
Software Reliability	4	3	4	4	5
Responsive Nature	3	4	2	3	3
Quality Assurance	5	4	5	5	4
Software Empathy	4	4	4	4	4
Tangible Nature	3	3	2	2	3

Hence forth the feedback from the customer is been taken on the pre-decided questioner and then mapped to the feedback weightage. The final result of this process is the overall satisfaction rate or score.

B. Process Related Matrix

The main objective for software development to develop software which is satisfying all customer needs and in parallel it is also important to improve the software development process to achieve higher satisfaction rate during the further development tasks [9]. The software matrix for process contains information related to multiple factors concerning about the process of development in the organization. The following sub-categories are used for specific purposes:

1) Machine Testing – Defect Density Matrix: The defect density is measured during the system testing of the system. The system testing is performed manually or automatically in a simulated real time environment. The information stored in the defect density matrix is a nearly corrected correlation of the defects in the real time. Unless the assumptions of the production system are massively incorrect, the defect density matrix can generate a good prediction.

2) Machine Testing – Defect Pattern Matrix: The total defects detected to be rectified are the overall scenario for the system. This might not be sufficient to predict the number of probable upcoming defects in the under development modules. Hence another matrix plays a role for prediction is Defect Pattern Matrix. This matrix denotes and helps in proper prediction of the defects which may be faced by the development team in under development modules for the same software product.

3) Defect Resolving Matrix: The detection of the defects is the half way task completion for the development team. The further task is to correct or resolve the defects. Another matrix called Defect Resolving or Removal matrix keeps track of the number of defects detected and resolved in the system. This helps in the identification of testing and resolution team efficiencies.

4) Effectiveness Matrix during Defect Removal: The input from the Defect Resolving Matrix is processed to generate the efficiency of Effectiveness Matrix. The effectiveness for each developed module can be calculated as following, considering DR_m is the efficiency for the module "m" as

$$DR_m = \frac{NDR_m}{TD_m} X100\%, \qquad \text{…. Eq 3}$$

Where NDR_m the number of defects is resolved and TD_m is the total number of defects detected in the module.

C. Maintenance Related Matrix:

After the completion of the development cycle, the software product is delivered to the client. Thereafter the maintenance process for the software starts including patch releases and corrective measured based on the customer feedback. Here the information is collected in multiple matrices to improve the defect resolution and customer satisfaction.

1) Matrix for Backlog Index: The backlog index denotes the ratio for number of defects resolved and number of defects reported. The formula for calculating the Backlog Index, B_I as following:

$$B_I = \frac{DR_m}{DD_m}, \qquad \text{…. Eq 4}$$

Where DR_m and DD_m are considered as Defects Resolved and Defects Detected in a month respectively.

2) Matrix for Fix Quality: Another matrix called Fix Quality matrix plays major role in defining the quality of the developed software delivered to the customer. The Fix Quality matrix denotes the ratio of number of total number of defects detected and resolved.

III. DATA MINING FOR DEFECT PREDICTION

Data Mining is a widely adopted technique for analysing various data sets generated from multiple processes [10]. The data mining techniques are very useful for generating information for report generation and prediction processes. Here we analyse multiple outcomes from the parallel researches employing data mining techniques on software matrices for defect probability detection and prediction.

- Software Classification Modelling: In the parallel researches the outcome of quality classification with the help of the existing dataset of software matrices is been demonstrated. However the approach used one software project matrices generated during the development process. To make the classification more robust, the researchers tried incorporating multiple other software project data. However the dataset from different projects are not compatible with the initial dataset. Hence a manual time consuming approach is been carried out to normalize the data. The work of Yi Liu et al. is to be considered as a bench mark for this model of work.

- Association Based Rule Mining Method: Use of Data Mining rules for establishing the correlation and prediction of software defects from the software matrices are also been proposed in parallel researches. The research conclusions are been applied to multiple project data for more efficient detection of software defects. The novel approach proposed by Song et al. is a notable mark in this direction of research and the approach proposed by them is also been compared with the PART or C4.5 algorithms to demonstrate the improvement. However this is to be understood that, generalizing the algorithm for over 150 projects is not a simple task and the approach can focus rather on normalization of the data.

- Software Defect Prediction using Classifiers: In other parallel tracks or researches also demonstrates the use of 22 most popular algorithms for data classification for defect prediction. The outcome of the research demonstrates that the algorithms demand the initial dataset to be in multiple dissimilar formats to be analysed. However the clarification algorithms demonstrate the same efficiency in detecting the defects. The work for Lessmann et al. is a benchmark for the comparative study.

Hence in the parallel researches we denote the significant gap of learning capabilities in the application of data mining algorithms. Also the need for initial data normalization is also ignored in all the mentioned studies.

IV. NEURAL NETWORK BASED APPROACHES FOR DEFECT PREDICTION

Here in this work we analyse the applicability of the artificial neural networks in software defect predictions [8] [11]. A numerous number of researches is been carried out to understand the deficiencies identified in case of data mining approaches. The neural networks are capable of identifying the gaps and rectifying them to a certain extent [12]. Here we understand the results and conclusions of the parallel researches based on applicability of artificial neural networks for software defect detection and predication [13].

- Feedforward Neural Network: The uses of neural networks are popular in case of defect detection for software modules. The most common types of research outcomes demonstrate the use of feedforward neural networks [Figure 1] to detect the defects.

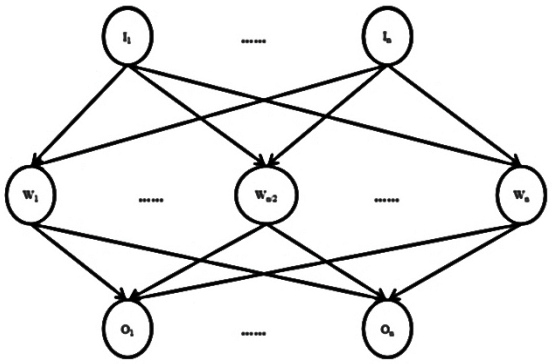

FIGURE 1: SIMPLE FEEDFORWARD NEURAL NETWORK FOR DEFECT DETECTION

The feedforward neural network deploys a mechanism for calculating the outputs with the modified weights assigned for each calculation. Here is the process of calculation

$$O_i = \sum_{i=0}^{n} I_i W_i \ , \qquad \text{…. Eq 5}$$

Where O_i is the output in a single node based on the I_i and W_i, the input and weight respectively.

The weight for the neural network is to be modified in each state as following:

$$W(t+1) = W(t) \pm \Delta W \ , \qquad \text{…. Eq 6}$$

Where the W(t+1) denotes the next weight to be considered as W(t) denotes the present weight assigned to the network and ΔW denotes the shift in weight factor.

The work of Ebru Ardil et Al. cannot be ignored in case of use of feedforward neural network for defect detection.

- Backpropagation Neural Network: The Resilient backpropagation neural network [Figure 2] is the most effectively studied in the parallel research works. The outcome of the works demonstrates the use of neural network for classification of the defects in terms of seriousness or severity.

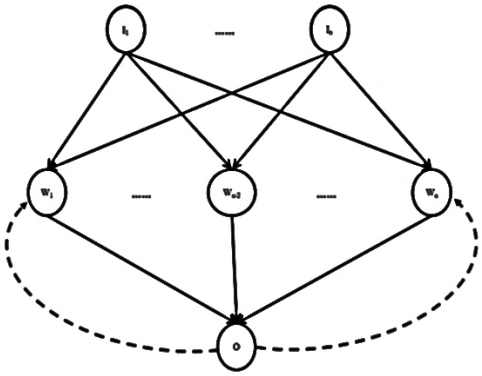

FIGURE 2: BACKPROPAGATION NEURAL NETWORK FOR CLASSIFICATION OF DEFECT CATEGORIES

The back propagation neural networks deploy the technique for calculating the final output based on the modified weight based on the corrective measures taken from weight modification and output.

The calculation for the final output can be understood as,

$$O = \sum_{i=0}^{n} I_i W_i \ , \qquad \text{…. Eq 7}$$

Where I_i and W_i, are the input and weight respectively.

The calculations for the weights can be calculated as,

$$W(t+1) = W(t) \pm \Delta W \pm O(t) \ , \qquad \text{…. Eq 8}$$

Where the W(t+1) denotes the next weight to be considered as W(t) denotes the present weight assigned to the network and ΔW denotes the shift in weight factor and the O(t) denotes the corrective measure from the output in the previous step.

The work of Joanhong et al. is a significant mile stone for comparing multiple outcomes from different neural network types under backpropagation.

- Fuzzy Neural Networks: For tracking the software reliability, the use of Fuzzy Neural Network cannot be ignored. The fuzzy neural networks deploy a mechanism for tracking the reliability of the software based on the Fuzzy rules and the inputs as number of defects per kilo line of codes [14] [15].

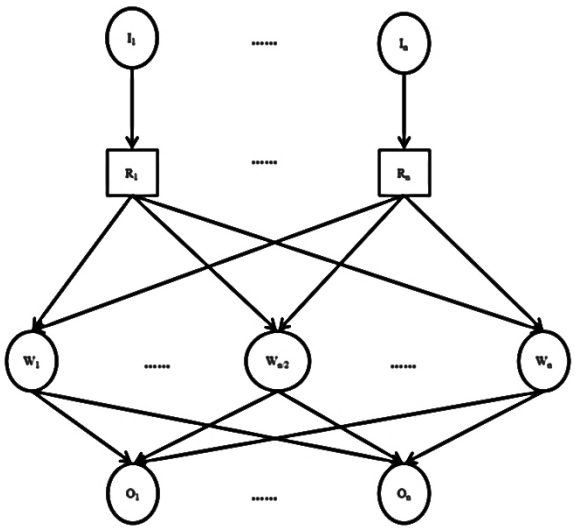

FIGURE 3: FUZZY NEURAL NETWORK FOR RELIABILITY PREDICTION

The calculation for the final output can be understood as,

$$O = \sum_{i=0}^{n} I_i (W_i + R_i), \qquad \text{….} \qquad \text{Eq 9}$$

Where, R_i denotes the Fuzzy Rule from the rule set based on the member functions.

The calculation of the weight remains same for the other types.

In this work we majorly consider the use of the Fuzzy Neural Networks for Defect tracking and prediction.

Hence after the extensive study of data mining and neural network based works, we understand the importance of normalization of data. Henceforth in this work we deploy an approach for data normalization.

V. RESULTS

Hence the applications methodology is been demonstrated in the light of normalization for input attributes for defect prediction.

Before we understand the results and conclusion notes from the work, let us take a look into the dataset used for this work.

During the testing of the modified algorithm based on Fuzzy Neural Networks, the "Class Level Data For KC1 / Software Defect Prediction" is used.

Here we try to understand the dataset [Figure 6], The Defect Level or DL is considered as True or False in case of presence of a defect or no defect respectively.

@attribute PERCENT_PUB_DATA numeric	@attribute minHALSTEAD_VOLUME numeric	@attribute avgNUM_UNIQUE_OPERANDS numeric
@attribute ACCESS_TO_PUB_DATA numeric	@attribute minNUM_OPERANDS numeric	@attribute avgNUM_UNIQUE_OPERATORS numeric
@attribute COUPLING_BETWEEN_OBJECTS numeric	@attribute minNUM_OPERATORS numeric	@attribute avgLOC_TOTAL numeric
@attribute DEPTH numeric	@attribute minNUM_UNIQUE_OPERANDS numeric	@attribute sumLOC_BLANK numeric
@attribute LACK_OF_COHESION_OF_METHODS numeric	@attribute minNUM_UNIQUE_OPERATORS numeric	@attribute sumBRANCH_COUNT numeric
@attribute NUM_OF_CHILDREN numeric	@attribute minLOC_TOTAL numeric	@attribute sumLOC_CODE_AND_COMMENT numeric
@attribute DEP_ON_CHILD numeric	@attribute maxLOC_BLANK numeric	@attribute sumLOC_COMMENTS numeric
@attribute FAN_IN numeric	@attribute maxBRANCH_COUNT numeric	@attribute sumCYCLOMATIC_COMPLEXITY numeri
@attribute RESPONSE_FOR_CLASS numeric	@attribute maxLOC_CODE_AND_COMMENT numeric	@attribute sumDESIGN_COMPLEXITY numeric
@attribute WEIGHTED_METHODS_PER_CLASS numeric	@attribute maxLOC_COMMENTS numeric	@attribute sumESSENTIAL_COMPLEXITY numeric
@attribute minLOC_BLANK numeric	@attribute maxCYCLOMATIC_COMPLEXITY numeri	@attribute sumLOC_EXECUTABLE numeric
@attribute minBRANCH_COUNT numeric	@attribute maxDESIGN_COMPLEXITY numeric	@attribute sumHALSTEAD_CONTENT numeric
@attribute minLOC_CODE_AND_COMMENT numeric	@attribute maxESSENTIAL_COMPLEXITY numeric	@attribute sumHALSTEAD_DIFFICULTY numeric
@attribute minLOC_COMMENTS numeric	@attribute maxLOC_EXECUTABLE numeric	@attribute sumHALSTEAD_EFFORT numeric
@attribute minCYCLOMATIC_COMPLEXITY numeric	@attribute maxHALSTEAD_CONTENT numeric	@attribute sumHALSTEAD_ERROR_EST numeric
@attribute minDESIGN_COMPLEXITY numeric	@attribute maxHALSTEAD_DIFFICULTY numeric	@attribute sumHALSTEAD_LENGTH numeric
@attribute minESSENTIAL_COMPLEXITY numeric	@attribute maxHALSTEAD_EFFORT numeric	@attribute sumHALSTEAD_LEVEL numeric
@attribute minLOC_EXECUTABLE numeric	@attribute maxHALSTEAD_ERROR_EST numeric	@attribute sumHALSTEAD_PROG_TIME numeric
@attribute minHALSTEAD_CONTENT numeric	@attribute maxHALSTEAD_LENGTH numeric	@attribute sumHALSTEAD_VOLUME numeric
@attribute minHALSTEAD_DIFFICULTY numeric	@attribute maxHALSTEAD_LEVEL numeric	@attribute sumNUM_OPERANDS numeric
@attribute minHALSTEAD_EFFORT numeric	@attribute maxHALSTEAD_PROG_TIME numeric	@attribute sumNUM_OPERATORS numeric
@attribute minHALSTEAD_ERROR_EST numeric	@attribute maxHALSTEAD_VOLUME numeric	@attribute sumNUM_UNIQUE_OPERANDS numeric
@attribute minHALSTEAD_LENGTH numeric	@attribute maxNUM_OPERANDS numeric	@attribute sumNUM_UNIQUE_OPERATORS numeric
@attribute minHALSTEAD_LEVEL numeric	@attribute maxNUM_OPERATORS numeric	@attribute sumLOC_TOTAL numeric
@attribute minHALSTEAD_PROG_TIME numeric	@attribute maxNUM_UNIQUE_OPERANDS numeric	@attribute DL {_TRUE,FALSE}

FIGURE 6: KC1 DATASET

The data for the dataset is not pre-normalized and after the normalization should be applied to the algorithms [Figure 7].

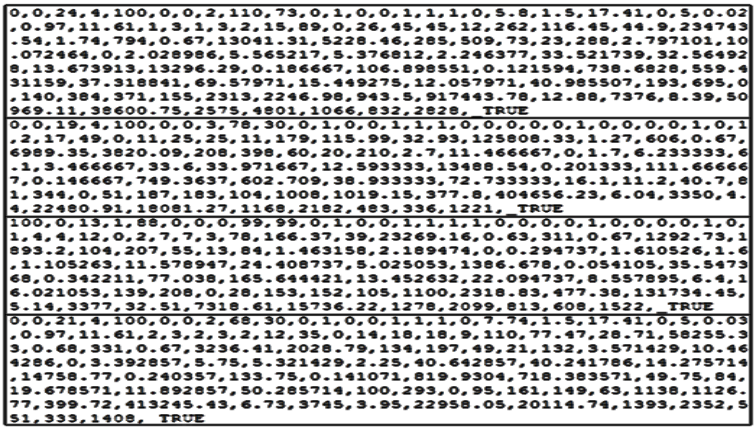

FIGURE 7: NORMALIZED DATA IN THE DATASET

The modified approach is been tested on KC1 dataset and the following improvements is been observed [Table -3].

TABLE IIIII: SATISFACTION WEIGHTAGE MAPPING

Neural Network Model	Initial Accuracy %	Accuracy % with Normalized Data	Improvement	% of Improvement
Random Tree Classifier	94.56	95.5	0.94	0.99
Regression Classifier	96	96.3	0.3	0.31
CART or Classification And Regression Tree	96.8	96.9	0.1	0.1
Multi-Layer Perceptron Neural Network	94.3	95.4	1.1	1.17

Hence it is clarify observable that the normalization of the initial dataset can improve the results. We also notice the improvement is higher for a Multi-Layer Perceptron neural network.

VI. CONCLUSION & FUTURE WORK

In the view of the need for normalization of the data in case of software defect prediction, we proposed a combination of automatic pre-processing module for the neural networks with or without fuzzy rule based networks. During the research work, we have used the KC1 dataset to predict the software defects based on the popular neural algorithms like RTC, RC, CART and MLP. We notice that the performance of most of the neural network algorithms can be improved with the pre-processed and normalized datasets. However we can realize that the multi-layer perceptron can make much more improvements with the normalized datasets.

Hence with the scope for more improvements, we understand the need for further improvements in case of multi-layer perceptron neural networks.

Henceforth we reset the scope for future work need to be carried out in this direction. Firstly, the test results needs to be validated on multiple software project datasets, which can be obtained from open and public data sources and Secondly, an improvement study needs to be carried out for multi-layer perceptron neural networks apart from the normalization of initial datasets. In the upcoming research findings we propose to address these issues.

REFERENCES

[1] Y. Chen, X. Shen, P. Du and B. Ge titled “Research on software defect prediction based on data mining” published at Computer and Automation Engineering on 2010 at Singapore.

[2] M. Ravat and S. Dubey titled “Software defect prediction models for quality improvement: a literature study” published at International Journal of Computer Science on 2012.

[3] T. Hall, S. Beechams, D. Bowes and S. Counsel titled “A systematic review of fault prediction performance in software engineering” published at Software Engineering IEEE Transactions on 2011.

[4] K. Elish and M. Elish titled “Predicting Defect-Prone Software Modules Using Support Vector Machines” published at Journal of Systems and Software on 2008.

[5] C. Catal and B. Diri titled "A systematic review of Software Fault Prediction Studies" published at Expert Systems with Applications Elsevier on 2009.

[6] C. Catal titled "Software fault prediction: A literature review and current trends published at Expert Systems with Applications Elsevier on 2011.

[7] A. G. Koru and H. Liu titled "Building Defect Prediction Models in Practice" published at Software IEEE Transactions on 2005.

[8] A. Porter and R. Selby titled "Empirically Guided Software Development Using Metric-Based Classification Trees" published at Software IEEE on 1990.

[9] M. Mertic, M. Lenic, G. Stiglic and P. Kokol titled "Estimating Software Quality with Advanced Data Mining Techniques" published at Software Engineering Advances International Conference on 2006 at Tahiti.

[10] C. Chang, C. Chu and Y. Yeh titled "Integration In-Process Software Defect Prediction with Association Mining to Discover Defect Pattern" published at Information and software technology on 2009.

[11] Y. Singh, A. Kaur and R. Malhotra titled "Software Fault Proneness Prediction Using Support Vector Machines" published at Proceedings of the World Congress on Engineering on 2009 at London, U. K.

[12] H. A. Al-Jamimi and L. Ghouti titled "Efficient Prediction of Software Fault Proneness Modules using Support Vector Machines and Probabilistic Neural Networks published at Software Engineering IEEE on 2011.

[13] H. Can, X. Jianchun, Z. Ruide and L. Juelong titled "A New Model For Software Defect Prediction using Particle Swarm Optimization and Support Vector Machine" published at Control and Decision Conference on 2013 at Guiyang.

[14] K. Punitha and S. Chitra titled "Software Defect Prediction Using Software Metrics –A Survey" at Information Communication and Embedded Systems on 2013 at Chennai.

[15] G. K. Armah, G. Luo and K. Qin titled "Multi_Level Data Pre-Processing for Software Defect Prediction published at Information Management, Innovation Management and Industrial Engineering on 2013 at Xi'an.

[16] P. Wang, C. Jin and S. Jin titled "Software Defect Prediction Scheme Based on Feature Selection" published at Information Science and Engineering on 2012 at Shanghai.

[17] B. Ma, K. Dejaeger, J. Vanthienen and B. Baesens titled "Software Defect Prediction Based on Association Rule Classification" at International Conference on E-Business Intelligence on 2010 at China.

AUTHOR INTRODUCTION

Rajeshwar Rao. K (ISTE, CSI Life Member), working as .Prof in Department of CSE, SIET , Hyderabad , Having rich teaching experience and interested research domains : Data mining, Cloud computing, Computer Networks , Data Analytics.