Computing and Informatics, Vol. 32, 2013, 203–224

APPLYING SOFT COMPUTING APPROACHES TO PREDICT DEFECT DENSITY IN SOFTWARE PRODUCT RELEASES: AN EMPIRICAL STUDY

Vijai Kumar

Data Communication and Product Engineering Services
Aricent Technologies, Gurgaon, India
e-mail: vkumar.phd@gmail.com

Arun Sharma

Department of Computer Science and Engineering
Krishna Institute of Engineering and Technology, Ghaziabad, India

Rajesh Kumar

School of Mathematics and Computer Applications
Thapar University, Patiala, India

Communicated by Igor Mokriš

Abstract. There is non-linear relationship between software metrics and defects, which results to a complex mapping. Therefore, to focus on the defect density area, it is a critical business requirement of effective and practical approach, which can help find the defect density in software releases. Soft computing provides a better platform to solve the non-linear and complex mapping problem. The aim of this paper is to formulate, build, evaluate, validate and compare two main sections of soft computing, fuzzy logic and artificial neural network approaches in prediction of defect density of subsequent software product releases. In this research, these two approaches are formulated and applied to predict the existence of a defect in file of software release. Both approaches have also been validated against various releases of two commercial software product release data sets. The validation criteria include mean absolute error, root mean square error and graphical analysis. The analysis of the study shows that artificial neural network provides better results compared to Fuzzy Inference System; but applicability of best approach depends on the data availability and the quantum of data.

Keywords: Software metrics, defect, defect density, release, prediction, fuzzy inference system, artificial neural network

Mathematics Subject Classification 2010: 68N30

1 INTRODUCTION

A software system must be changed or enhanced as per business requirement and need of subsequent releases. If we are able to predict the defect prone area of the system/subsystem module, it can impact the schedules, cost and customer satisfaction. There are various investigations in literature to show relationship between the direct measures and defects. Prediction of entities is crucial in software metrics analysis and design. Defect Density (DD) is a software quality attribute, which gives the reliability measure of the software product. DD can be defined as the number of defects per defined area, as a file, KLOC or module. DD can be measured as a sum of defects or faults in a file or module. This helps us find the high-risk prone modules out of the complete system and as per the Pareto principle that greater defects lie in the 20 % of the modules and hence work on them. Research literature concludes that in a large software base, 20 % of files have 80 % of the defects, which indicates there should be more focus on the infected 20 % area and more efficient resources should be put to work on it rather than the equal focus and resource distribution on the entire software base.

The prediction of DD during the subsequent releases of the software system is crucial to produce the system with more reliability and quality. In last three decades, soft computing (SC) has been vastly used in basic sciences and engineering disciplines. Artificial Neural Networks (ANNs), Fuzzy Inference Systems (FIS) and Adaptive Neuro Fuzzy Systems (ANFIS) can be used for universal approximations. In software development process, there are various attributes, which need to be calculated during the Software Development Life Cycle (SDLC). Apart from normal calculation, it is a good practice to find the possibility of occurrence whether it is a good or bad event to take the preventive measure beforehand. After applying the SC approaches in other engineering disciplines, now researchers are trying to use these techniques in software engineering disciplines as well. In literature, it is proven that there are many software attributes, which can be predicted in advance using the SC techniques. The main attributes include DD, effort estimation, software quality, software reusability, software maintainability, etc. The prediction can be done using empirical methods or SC methods viz., Fuzzy Logic (FL), ANN, Neuro-Fuzzy (NF), Genetic Algorithm (GA) and others. The researchers have shown that they are capable of approximating general non-linear relationships to high degree of accuracy [1, 2, 3]. The combination of ANNs and FIS leads to best results as they complement each other [4]. Khoshgotaar et al. [5, 6] used ANNs and fuzzy systems to estimate the quality of a complex telecommunication system as well as for cost estimation which makes ANN and fuzzy to be a good tool in empirical studies in software engineering. If we have the probability of incident, it can be used as a precaution measure for another similar domain project or subsequent releases within a software project or across the other project releases. There are various project metrics entities, which can be predicted using FL and ANN techniques.

The rest of the paper is organized as follows: Section 2 describes the related work in prediction of DD. Section 3 contains the concept and data variables used in two approaches. Section 4 discusses the proposed FIS and ANN based method to predict the defect density. Section 5 describes the comparative results and applications of proposed FIS and ANN approaches. Section 6 provides the insight of the conclusion and future prospects.

2 RELATED WORK

SC techniques are different from conventional computing paradigms; unlike hard computing, it is tolerant of imprecision, approximation, uncertainty and partial truth. That is why SC inherits some of the important properties of human mind. The main principle of SC is to achieve tractability, robustness and low solution cost. Recently the authors have shown a great amount of interest towards the use of SC approaches to solve the uncertain problems in software engineering. To classify modules as fault prone, Khoshgoftaar et al. [5] used ANN with back propagation training algorithm. To measure software maintainability, Aggarwal et al. [7] have used a fuzzy model. In open source study, Gyimothy et al. [8] empirically validated important metrics for fault prediction on open source. Graves [9] has proved that software change history can also be used to predict the DD in modules and the systems. Khatatneh et al. [10] developed a new fuzzy expert system to predict the software failures. Kehan Gao et al. [11] focused to select the attributes for software quality estimation. Comparative study also presented to evaluate the proposed approach for attribute selection. Fenton et al. [12] discussed four general approaches to predicting the number of defects in a system. The author presented the approach of finding the correlation between DD and code metrics. Khoshgoftaar et al. [13] used factor analytic variables to fit regression model to a number of defect data sets. Principal Component Analysis is used in studies to reduce the dimensionality of many related metrics to a small type of set. Ohlsson and Alberg, [14] presented a study at Ericsson where metrics derived from design documents were used to predict fault-prone modules prior to testing. Gyimothy et al. [8] empirically evaluated and validated fault prediction of Chidamber and Kamerer [15] metrics on open source software. They used linear, logistic regression and neural network for machine learning methods for model prediction.

Statistical models have been explored vastly in prediction of software attributes. SC techniques, especially FIS and ANN have changed the focus over the years. FIS and ANN are being successfully applied across various domains like finance, medicine and other engineering areas. Khoshgoftaar et al. [5] worked on a case study of avionics software to predict the testability of each module using static source code metrics and ANN. They were able to predict testability because they are able to model linear relationship using ANN. Aggarwal et al. [16] predicted the maintainability using object oriented metrics and applying ANN. They examined the application of ANN for software quality prediction and taking the input variables as object oriented metrics. In their study, maintenance effort was the dependent variable. Principal components of eight OO metrics were used as independent variables. The results showed a good accuracy with ANN model. Yuan et al. [17] used Fuzzy Subtractive Clustering to predict the number of faults. Aggarwal et al. [18] have developed a fuzzy model for measuring software maintainability. The inputs to the model are comment ratio, average live variable, average life span and average cyclomatic complexity. The output of the model is average corrective maintenance time. They have used Mamdani style inference. Kumar et al. [19] presented the applicability, usability, and extendibility to rank the usage of the existing approaches for component based systems in software industries. The author concluded that SC approaches could work better for component bases systems (CBS) as well.

3 CONCEPT AND VARIABLE SELECTION

3.1 Empirical Data

We considered two projects of different software industry domains. We gathered a large set of data of two projects for three attributes. Using the Rational Rose bug history tool, we captured the data set of 4 000 files. The same process is repeated for two real projects, which fall into the following category

- An optical telecom project, which has component and embedded project (A type).
- A component based application project (B type).

The A project is an optical telecom project, which is used as an optical communication platform across cities. It has some object oriented based system design as well as structural design. It has been developed in 4 years timeframe. The B type was a component based application project. It was developed as an IDE tool, which is used as debugging and development environment tool for the software development project. It contains Java components and files. Therefore, we had a good mix of various domain projects. This results in a good scope of approach validation across various domain projects. The data for two projects consist of many attributes. We extracted three attributes, namely complexity, total lines of code and pre release defects. There are numerical values for each of the attribute in A and B projects.

The attribute values were captured at file, module and package level. In addition to three attributes we gathered the total defects in a particular file of a release of the two selected projects.

The following section gives the details of the attributes and each case contains the following information:

Name: The name of the file module or package, to which this case corresponds. It can be used to identify the source code in the release and may be needed for additional data collection.

Pre release defects: The number of non-trivial defects in a file that were reported in the last six months before release.

Post release defects: The number of non-trivial defects in a file that were reported in the first six months after release.

Total defects: It is the aggregation of the Pre and post release defects in a file for a particular project release.

Complexity metrics: For each case, we computed a file complexity metrics. Metrics that are computed for functions/classes or methods are aggregated by using the average.

Total lines of code: The number of lines of code in a file, module or package.

Number of function calls: The number of function calls in file, module or package.

3.2 Input and Output Variables

The data gathered during the development and testing of the project is used to develop FIS and ANN model, which in turn may be useful in future projects and planning. To find the data relevance we observed that most of the files (approximately 50 %) had no defects. The remaining half has defects ranging from one to 100. Principal component analysis method is used to quantify the data set. The defect distribution among the files and modules leads to interesting research points:

- To identify the files which have the defects? This falls into classification problem.
- To estimate the intensity of the defects, i.e. the number of defects in a file or module.
- To predict the DD for efficient planning of the resources during the subsequent releases or the future projects.

Based on the discussion with software architects, we have identified the set of software metrics which plays an important role and is responsible for estimating the fault in software system. After subsequent discussions, we narrowed down to the three metrics, which was possible member for building a new metrics, which can be used easily in the software industries. PCA technique is used to pick the uncorrelated metrics. The research team considered the following fact while deciding on the attributes:

- Simplified and generally available metrics.
- The attributes, which could be easily collected during the software development process.
- The metrics, which has high impact, either positive or negative, on the number of defects in the system.
- Although there was a large set of software attributes available, we have narrowed down to few for having a simple and the best formulation of the metrics.

Based on the practical experience we finalized the three software metrics for input and DD as the output. The following factors have been identified, which will influence defect density:

PREDD – Pre-RElease Defects in a file and aggregated for module.

TLOC – Total Lines Of Code in a file and aggregated for module.

VG – McCabe cyclamate complexity of a file and aggregated for module.

3.3 Principal Component Analysis

Principal Components Analysis (PCA) is a technique of identifying patterns in data, and quantifying the data to highlight their similarities and differences, since it is difficult to find the pattern out of high dimension, which cannot be found using graphical representation. Recently PCA has been used widely as a powerful tool for analyzing data in image processing field. PCA is used to maximize the sum of squared entry of each factor extracted in turn [20]. In addition to its accuracy, PCA gives the liberty to find the pattern in data without much loss of information. We used the power of PCA to quantify the data set used in ANN training and validation.

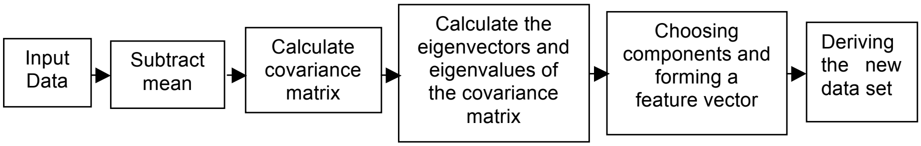
Input Data
Subtract mean
Calculate covariance matrix
Calculate the eigenvectors and eigenvalues of the covariance matrix
Choosing components and forming a feature vector
Deriving the new data set

Fig. 1. Steps followed for PCA

PCA is used to transform raw metrics values of prerelease defects, complexity and lines count to a set of unrelated variables. PCA is also used to maximize the sum of squared loadings of each factor extracted in turn. The PCA aims to drive a new variable (p_n), called Principal Component out of a given set of variables x_m's ($m = 1, 2, \ldots, k$) [19].

$$p_1 = b_{11}x_1 + b_{12}x_2 + \ldots + b_{1k}x_k$$

$$
\begin{aligned}
p_2 &= b_{21}x_1 + b_{22}x_2 + \ldots + b_{2k}x_k \\
&\vdots \\
p_k &= b_{k1}x_1 + b_{k2}x_2 + \ldots + b_{kk}x_k
\end{aligned}
$$

$b_n m$ is calculated in such a way that derived PCA fulfills the following conditions:

- Principal components are uncorrelated, i.e. orthogonal.
- The p_1 has the highest variance and similarly for further components.

3.4 Validation Criteria

Accuracy is percentage of the predicted values that match with the expected values of the number of faults in the file or module. For the present study, we used Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE is the average of the difference between predicted and actual value in all test cases; it is the average prediction error [21]. The formula for calculating MAE is given in the following equation:

$$\text{MAE} = (|A_1 - P_1| + |A_2 - P_2| + \ldots |A_n - P_n|)/n \tag{1}$$

Assuming that the actual output is A, the expected output is P. RMSE is a frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated [21]. It is just the square root of the mean square error, as shown in the following equation:

$$\text{RMSE} = \sqrt{\frac{(A_1 - P_1)^2 + (A_2 - P_2)^2 + \ldots + (A_n - P_n)^2}{n}} \tag{2}$$

The best system is that having the highest accuracy and least values of MAE and RMSE.

4 PROPOSED FIS, ANN APPROACHES AND EXPERIMENTAL DESIGN FOR DEFECT DENSITY PREDICTION

SC is equipped with two main techniques, namely FL and ANN. In the present study we applied these two sections of SC for DD prediction in subsequent software product releases.

4.1 Fuzzy Inference System

Zadeh [22] derived the concept of FL to implement vagueness in linguistic variables. The concept of FL implements and simulated the human knowledge as nature implements it in daily life. FL theory is based on fuzzy sets. A fuzzy set is a set without a crisp, clearly defined boundary, which can contain a part of membership of an element. A membership function, which is called MF, can be any curve that defines how every point of input space is related or mapped to a degree of membership value in-between 0 and 1. Sometimes the input space is referred to as universe of discourse. Although there are various MFs but riangular and trapezoidal ones are most used, as well as simple MFs which are formed with straight lines. FL reasoning is a superset of Boolean logic theory. To interpret the if-then rule involves fuzzification of the input and applying the suitable fuzzy operators [23].

Let X be a set of objects and x belong to X. A classical set S, $S \subseteq X$, can be defined as a set of elements or objects $x \in X$, in such a way that x can be either contained in set S or not.

A fuzzy set S in X can be defined as a set of ordered pairs

$$A = \{(x, \mu S(x))|x \in X\} \tag{3}$$

where $\mu S(x)$ is called MF for the fuzzy set S. A MF maps all elements of X to a membership value in between universe of discourse 0 and 1. Hence, Equation (3) is an obvious extension of classical definition in which the defined function is allowed to have a value between 01 and 1. Figure 2 illustrates the fuzzification process.

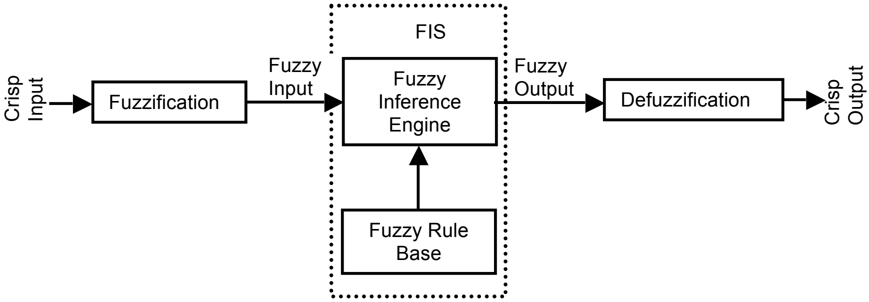

Fig. 2

4.2 Proposed FIS based approach

It is often impossible to estimate some attributes directly. DD is one of them, which can predict using the other available attributes. For example, attributes (say, DD) are affected by many different factors and there is no straightforward method to measure or predict them. To predict the DD, one needs to establish a relationship of direct measures (which affects the DD attribute) with each other and drive the indirect measure as DD. We proposed a FL based approach for predicting DD of different releases of software system based on the earlier release. The proposed method has been validated as an approach for software system across few domains. It has been validated of two domains of complex type projects. We emphasize to predict DD by taking the direct measures as dependent attributes to predict the system reliability and quality. In the proposed approach, we have given the fuzzy based algorithm to predict the DD in a file or module. To calculate the DD in module or software system, we can take a sum of defects in all files or a module or system. Mathematically the relation between the dependent attributes and the defect of a file can be given as follows:

Let x_n, y_n, z_n be the values of the complexity, lines of code and pre release defects, respectively; then

$$D_f = F_m[x_n, y_n, z_n] \tag{4}$$

where F_m is the function of the x_n, y_n, z_n as a value of the dependent attribute in a file. D_f is the defect count in the file f. In the experimental design F_m is considered as fuzzified and fuzzy rule function.

The defect density of a module can be derived as

$$\mathrm{DD}_m = \sum_{f=1}^{m} [D_f] \tag{5}$$

where DD_m is the module DD. Using the above formulation of the problem we can find out the defect count in a file based on the three attributes values of a file. Taking the sum of all defects in each file of a module, the DD can be derived for a particular module. The function F_m is implemented using rules in FIS as described in the factors with defects to achieve the desired goal. DD is an indicator, which depends on several other measures in the software system. We propose that DD of a software system can be a measure of three most influenced factors mentioned above. These combined factors can be used to predict the DD, as it cannot be measured directly. The formulation given in Equations (4) and (5) is implemented using the fuzzy toolbox in MATLAB 7.1. For complexity let X be the universe of discourse and x_n be the value in the fuzzy boundaries $[0, 1]$. Then Equations (4) and (5) are implemented by applying fuzzy inference rules for the function F_m. The MF is used to map the actual attributes values into the fuzzy range. Similarly, for line count and pre release defects, the triangular MF is used for fuzzification of the crisp values.

The proposed FL based model considers all three factors as inputs and provides a crisp value of DD using the rule base. All inputs can be classified into fuzzy sets, namely low, medium and high. The output DD is classified as High, Medium, and Low. All possible combinations (i.e. 27) of inputs are considered to design the rule base. Each rule corresponds to one of the three outputs based on the expert opinions. These rules are defined to implement the FM given in Equation (4) above. Some of the proposed rules are shown as follows:

If complexity of a file is high, Lines of Code (LOC) is low and number of function calls is low then the DD will be medium.

If complexity of a file is high, LOC is low and the number of pre-release defects is medium then the DD will be medium.

If complexity of a file is high, LOC is low and the number of pre-release defect is high then the DD will be high.

All 27 rules are inserted into the proposed model and a rule base is created. Depending on a particular set of inputs, a rule is fired. Using the rule viewer, the output (i.e. DD) is observed for a particular set of inputs using the MATLAB Fuzzy toolbox. Table 1 shows the values of various parameters set for inputs, outputs for fuzzification process.

System	Name = 'DD'
	Type = 'mamdani'
	Version = 2.0
	NumInputs = 3
	NumOutputs = 1
	NumRules = 27
	AndMethod = 'min'
	OrMethod = 'max'
	ImpMethod = 'min'
	AggMethod = 'max'
	DefuzzMethod = 'centroid'

Output	Name = 'output'
	Range = [0 1]
	NumMFs = 3
	MF1 = 'L-DD' : 'trimf', [0 0.2 0.4]
	MF2 = 'M-DD' : 'trimf', [0.35 0.5 0.65]
	MF3 = 'H-DD' : 'trimf', [0.6 0.8 1]

Table 1. Parameter values for system and output

Fuzzication of inputs into output, MF for complexity, surface view for complexity, LOC and DD and rules for defuzzication process are shown in Figures 3, 4, 5 and 6, respectively. In Figure 3, Complexity, TLOC and PDEF are given as input to mamdani type FIS and the out is produced as DD. The value of DD can be estimated by taking all three values for input factors. Figure 4 describes the MF used in the process of fuzzi cation and de-fuzzi cation. Figure 5 shows the surface view of the decision based on the FIS rules. Figure 6 is a snapshot of the rules from MATLAB FIS toolbox rules viewer system.

4.3 Validation and Results of FIS Approach

The proposed approach works to find the DD at file and module level. As discussed above we had to project data for approximately 4 000 files. The metrics values are captured during the development and maintenance phase using the IBM Rational Rose tool. Therefore, we have data of three metrics values of 4 000 files for every project. After applying PCA technique, the data set is compressed to 2 847 data set entries. The proposed methodology is validated by using statistical evaluation

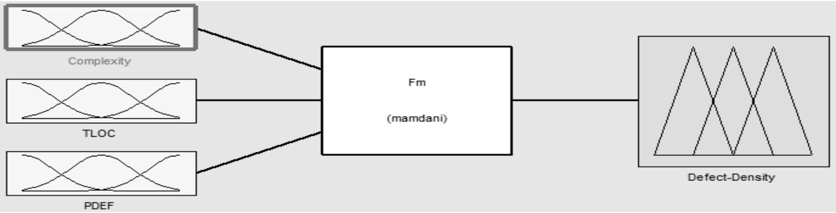

Fig. 3. Fuzzification of inputs into output (DD)

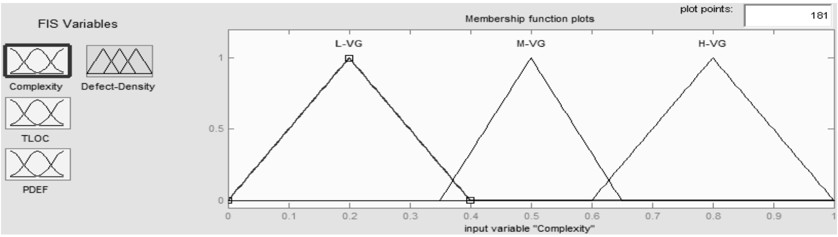

Fig. 4. MF for complexity attribute used in FIS

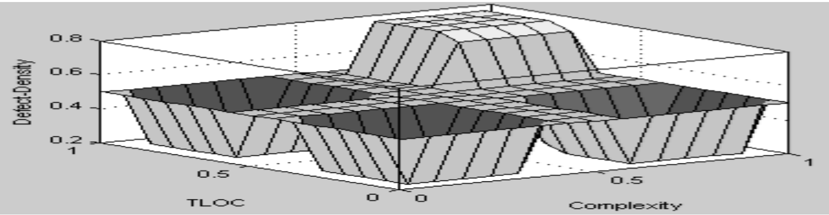

Fig. 5. Surface view for complexity on X-axis, LOC on Y-Axis and DD on Z-Axis by using MATLAB toolbox

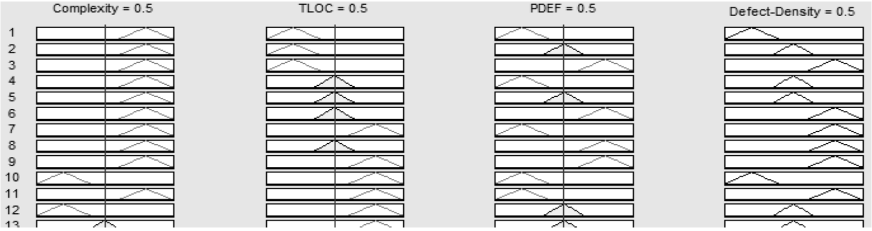

Fig. 6. Snapshot of rule viewer showing some of the rules

of the predicted values vs. actual data values. The proposed FL based approach is applied to predict the DD for subsequent releases of a single project and across the project.

The metrics for all the three factors of DD were measured for both projects. These metric values were then given as input to the proposed approach. The values of DD obtained from the proposed approaches are validated using the statistical methods.

From Table 2, it is clear that values obtained from FL based approach for DD prediction are maximum 77 % and minimum 73 % match to the actual DD of the files. We captured the output of the fuzzy approach and found out the root mean square error with the actual error with actual defects in the file. Comparisons of predicted defect density are made on the basis on accuracy, mean absolute error and root mean-square error.

Validation of the proposed formulation and experiment is done by taking the real project data as described in Section 4.3. Proposed fuzzy based prediction model has been applied on the dataset and the accuracy, MAE as per Equation (1) and RMSE as per Equation (2) are calculated. Accuracy, MAE and RMSE values are shown as follows.

Project	Accuracy	MAE	RMSE
P_1	73.40	0.266	0.515
P_2	77.55	0.225	0.473

Table 2. FIS validation results

From the above Table, it is clear that values obtained from FL based approach for DD prediction are 77 % match to the actual DD of the files. We captured the output of the fuzzy approach and found out the root mean square error with the actual error with actual defects in the file.

Figure 7 gives graphical representation and variance of actual defects and predicted DD using proposed FIS. The graph is prepared from the data of the actual defects from each file of the selected two projects. On the X-axis the low highlighted columns and the dark columns show the actual and predicted DD, respectively when the FIS approach is applied on the data. On the Y-axis, the particular file instance in the selected software system is shown for which there is data for 3 input variables and the DD is derived with FIS. Although the approach is validated on large number of files, this graph depicts the graphical view of actual versus predicted DD for a set of 55 files. This approach is able to predict the DD up to 77 % accuracy and 0.4738 as RMSE.

4.4 Artificial Neural Network

ANNs have been developed as generalizations of mathematical models of biological nervous systems of human brain. A first interest in neural networks emerged after simplified design of neurons by McCulloch and Pitts.

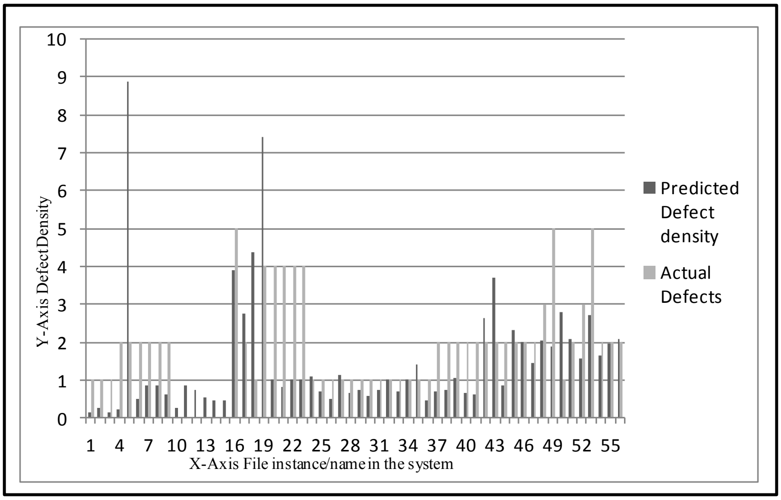

Fig. 7. Predicted vs. actual DD using FIS for validated data set

In feed-forward network, which is being used in our research, the signal travels in feed-forward direction from input to output units. In this type of neural network, no feedback connections are present but the data processing can spread over multiple layers of units [24]. Other recurrent network can contain feedback connections. Sometimes the activation values of the units have to go through a relaxation process so that network can evolve to a stable state. After the relaxation process, these activations do not change further any more.

The architecture of a particular artificial neuron is shown in Figure 8.

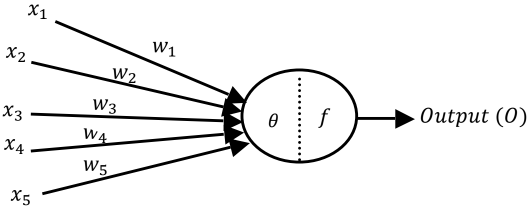

Fig. 8. Artificial neuron

The signal flow from input x_1, x_2, …, x_n is considered to be unidirectional which corresponds to consolidated neuron output signal (O).

The neuron output signal O is given by the following relation:

$$\mathrm{O} = f(\mathrm{net}) = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{6}$$

where w_i is the weight vector, and the function $f(\text{net})$ is referred to as an activation (transfer) function.

The variable net is defined as a scalar product of the weight and input vectors

$$\text{net} = W^T x = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \tag{7}$$

where T is the transpose of a matrix, and, in the simplest case, the output value O is computed as

$$\text{O} = f(\text{net}) = \begin{cases} 1, & \text{if } W^T x \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where θ is called the threshold level; and this type of node is called a linear threshold unit.

4.5 Proposed Artificial Neural Network Approach

ANN has been developed as generalizations of mathematical models of biological nervous systems. In a simplified mathematical model of the neuron, the effects of the synapses are represented by connection weights that modulate the effect of the associated input signals, and the nonlinear characteristic exhibited by neurons is represented by a transfer function. Artificial neural network is applied on the similar data set as we investigated using FL in the above section.

The input values are normalized using Min-max normalization. It performs a liner transformation of original data [25]. If Min_A and Max_A minimum and maximum values of an attribute in data set taken for the study. The formula 9 is used to map X value of A to X' as decimal value within the range of 0 to 1. So after normalization each value X' will be such that X' belongs to $[0, 1]$. Output values are again mapped to the actual number of defects.

The normalization concept can be formulized as follows:

$$X' = \frac{X - \text{Min}_A}{\text{Max}_A - \text{Min}_A} \tag{9}$$

where X' will be any value between 0 and 1 corresponding to a value X of an attribute.

Multilayer feed forward network model is used for modeling. Every node of the hidden layer is connected to input nodes. Input nodes are not directly connected to output nodes. In turn, it proves that ANN does not have any shortcut connection. ANN adjusts the weights to adapt the actual outputs and differences between the desired output and actual output are minimized.

The metrics values have been captured during the development and maintenance phase using the IBM Rational Rose tool. Therefore, we have data of three metrics values of 4 000 files for every two sated project. After applying PCA technique, the data set was compressed to 2 847 data set entries. The data set is divided into two parts with 3:1 ratio. 2 135 records from data set were used for training and 712 records are used for validation of the ANN approach. Then ANN model is developed using training data set. MATLAB is used for simulation of the training. Then validation data set is applied on the trained ANN model to validate and check the accuracy of the developed model.

Network Type	Feed Forward Back Propagation
Training Function	TRAINLM
Adaption Learning Function	LEARGDM
Number of Layers	2
Transfer Function	PURELIN
Performance	Root Mean Square Error (RMSE)
Layers	3
Input units	3
Output units	1
Hidden units	9

Table 3. Attributes for ANN architecture

The network architecture which was empirically determined and ANN architecture attributes are as follows: The neural network with above architecture is trained using the data set and validated for feed forward back propagation algorithm TRAINLM and simulated using MATLAB neural network toolbox. TRAINLM is network training function that updates weight and bias values according to Levenberg-Marquardt optimization. TRAINLM is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms. The input metrics to the ANN are pre release defects, complexity and lines count, and the output is defect density of a file.

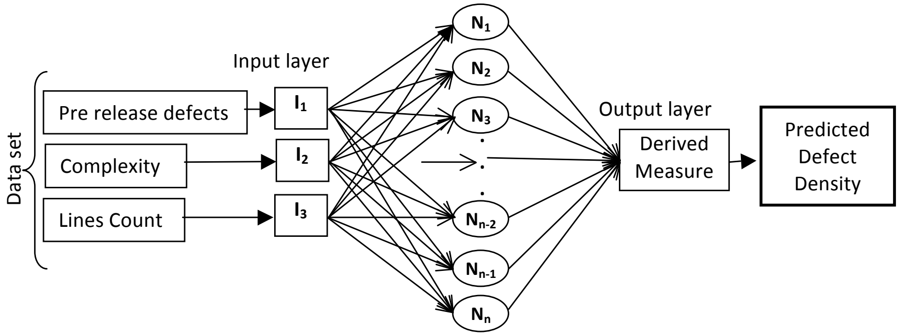

Fig. 9. Architecture of proposed approach

4.6 Validation and Results of ANN Approach

Validation of the proposed formulation and experiment is done by taking the real project data as described in Section 4.3. Proposed ANN based prediction model has been applied on the dataset and acuracy, MAE as per Equation (1) and RMSE as per Equation (2) are calculated. 712 data records are used for validation of ANN approach. It has shown accuracy, MAE and RMSE values as follows.

Project	Accuracy	MAE	RMSE
P_1	80	0.25	0.50
P_2	85	0.15	0.3872

Table 4. ANN validation results

Figure 10 gives the graphical representation and the variance of the actual defects and predicted DD using proposed ANN. This approach is able to predict the DD upto 85 % accuracy and 0.3872 as RMSE.

From the above table, it is clear that values obtained from proposed ANN based approach for DD prediction are 85 % match to the actual DD of the files. We captured the output of the ANN approach and found out the root mean square error with the actual error with actual defects in the file.

The graph in Figure 10 is prepared from the data of the actual defects from each file of the selected two projects. On the X-axis, the low highlighted columns and the dark columns are shown as actual defects and as predicted DD when the ANN approach was applied with mamdani model on the data gathered during the SDLC process. The attributes and architecture are used as shown in Table 3. On the Y-Axis the particular file instance in the selected software system is shown for which there is data for 3 input variables and the DD is derived with ANN. Although the approach is validated on large number of files, this graph depicts the graphical view of actual versus predicted DD for a set of 55 files. The calculated RMSE is 0.3162 which shows a strong implication that ANN provides better prediction than the FIS.

5 COMPARATIVE RESULTS AND APPLICATIONS OF THE PROPOSED APPROACHES

Software defects are the main indicator of software quality. To build and formulate the defect metrics, software defect is considered as dependent variable of three software metrics; these are easily available during SDLC, which increases practicality and applicability of the proposed approaches. The FIS engines are built to decide DD of a file, which can be aggregated, for DD at subsystem level of a product and hence DD as well as quality of a software release. The outcome of the proposed research on metrics design shows that software industries can use FIS and ANN approaches to predict the defect of software release files based of the data availability, which can help maintain the better software product quality and resource management across releases.

This paper presents the investigation using FIS and ANN as well as comparative study of the two approaches. Although ANN shows good results and stands out in

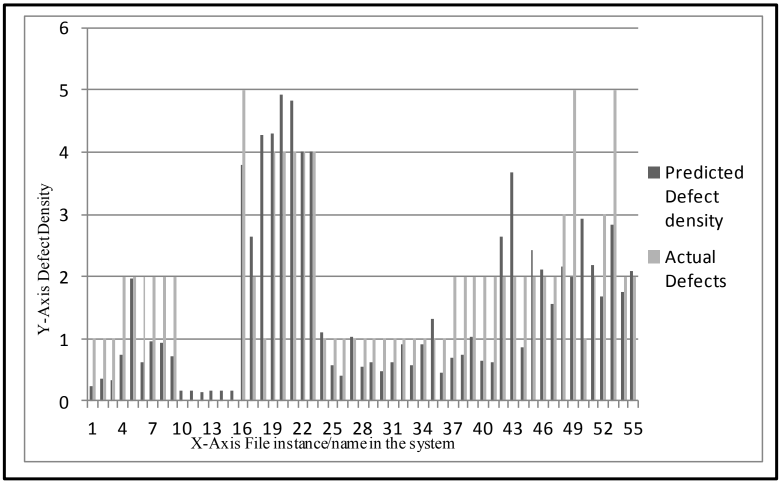

Fig. 10. Predicted vs. actual defect density using ANN for validated data set

case of DD prediction, FIS can also be useful in case on partial data of no data. Therefore if the data set is available and metrics values have been captured during the development process, ANN can be used for better results and in case of less data availability FL is also a useful tool for prediction of DD.

In Figure 11, the state diagram shows the proposed FIS and ANN approach application usage across multiple releases of a project. If we start the process of DD prediction from release R1 of project P_1, the DD prediction can be calculated in each release and use that is next release of software product. After predicting the DD in a particular release by using the proposed FIS or ANN approach, an efficient resource distribution and effort planning can help maintain the product quality and reliability.

The proposed approach has the following applications:

- As a human nature, we generally used to have the low/high/medium type of values for any attribute. One major advantage is that the proposed FIS approach may also work without the data, although we empirically evaluated the proposed model on real project data.
- Using the power of adaptation and learning, ANN can be used, which is simple; but it is expensive to collect the data set during the tight schedule of development.
- After first release or from a similar type of project data set, we will have the three attributes values ready for the system so it will be the best DD predictor tool for the next release; generally we used to have 50 to 100 releases in large and long term software projects.

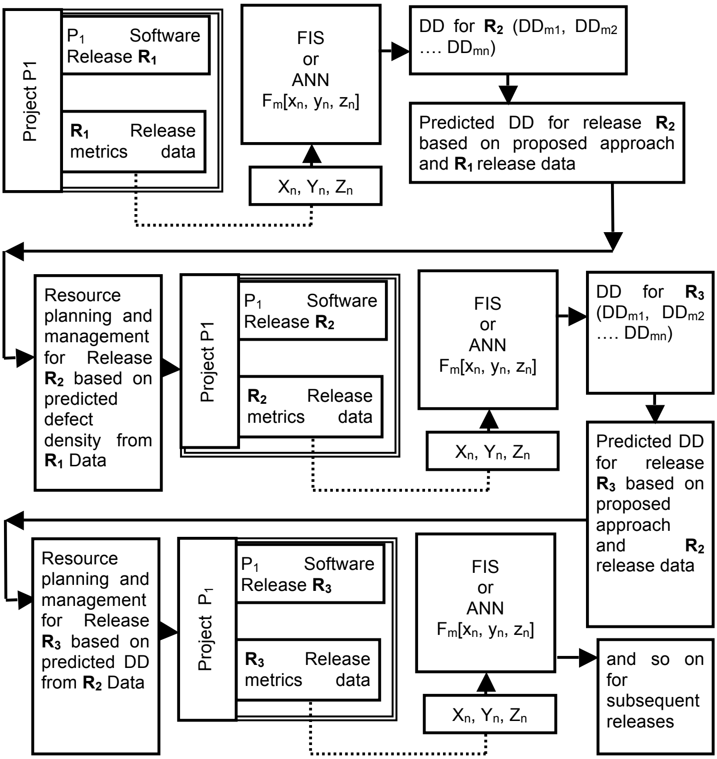
Project P1
P1 Software Release R1
R1 Release metrics data
FIS or ANN Fm[xn, yn, zn]
Xn, Yn, Zn
DD for R2 (DDm1, DDm2 .... DDmn)
Predicted DD for release R2 based on proposed approach and R1 release data
Resource planning and management for Release R2 based on predicted defect density from R1 Data
Project P1
P1 Software Release R2
R2 Release metrics data
FIS or ANN Fm[xn, yn, zn]
Xn, Yn, Zn
DD for R3 (DDm1, DDm2 .... DDmn)
Predicted DD for release R3 based on proposed approach and R2 release data
Resource planning and management for Release R3 based on predicted DD from R2 Data
Project P1
P1 Software Release R3
R3 Release metrics data
FIS or ANN Fm[xn, yn, zn]
Xn, Yn, Zn
and so on for subsequent releases

Fig. 11. Application of proposed approach across releases of Project P_1

- As it has been proved as a good estimator for different domain project, if the system needs to enhance or add a new feature, during the planning of software project the proposed approach will be able to predict the DD for the modules and files.
- The proposed approach has a good granularity as it can predict the DD at file level as well as at module or subsystem level which not only reduces the project planning errors and risk but also provides additional information for resource distribution across the project schedules.
- Even for a new project initiation the proposed approach can be used as a DD predictor for various modules. In fact, we generally used to have some rough figures for the three values for a new project by taking the reference from a similar type of previous projects.
- It can be a good evaluator for component on the shelf (COTS) product. For a COTS product, we will have the values of three input variables in advance; hence, the DD can be estimated using the proposed approach, which helps us

evaluate the COTS product quality. It is obvious that if we have the possibility of more defects then the system developed using the COTS product will also need more DD. We should have the attributes values for any ready COTS product. By that, we can predict the DD and hence use it as effort and schedule estimator as well as a quality indicator.

- As shown in Figure 11, the proposed approach can be used to predict the DD for the next release of the software product. Hence, it can optimize the resource loading, efforts and schedules for the release. It can be a part of regular practice during the software development and maintenance as release management process of software quality assurance.
- ANN approach is adaptive in nature; hence, the method can be trained for different environment based on the data nature and variables

6 CONCLUSION

There has been considerable research investigation to predict DD using various statistical methods and approaches. Either SC has not been completely explored for this area or there is lack of validation and comparative analysis. Therefore, our investigation and metrics design show that SC provides a good indicator in software quality and management. The best technique is concluded based on comparison of the validation results. Here the results of the practice shows that ANN stands better than FIS approach to predict the defected bundle of files. MAE and RMSE calculations indicate that ANN is best in case of accuracy. However, it is worth mentioning here that FIS is also a good method if there is no project data or partial data. It is not always possible to have a data collection to train the neural network. Hence, in practicality, there are pros and cons of both the approaches. Therefore, software industries can use any or combination of FIS and ANN approaches based on their data collection and strategies. The study indicates that now industries should accept and include the power of soft computing in general practice during the development and maintenance process. The combination of FIS and ANN, called neuro-fuzzy, can yield better results, so in future there is a need to explore neuro-fuzzy, fuzzy-genetic and other evolutionary approaches for prediction of defect density and other quality attributes.

REFERENCES

[1] Haykins, S.: A Comprehensive Foundation on Neural Networks. Prentice Hall 1999.

[2] Wang, L. X.—Mendel, J. M.: Fuzzy Basis Function, Universal Approximation, and Orthogonal Least Squares Learning. IEEE Trans. Neural Networks, Vol. 3, 1992, No. 5, pp. 807–814.

[3] Shing, J.—Jang, R.: ANFIS: Adaptive Network Based Fuzzy Inference System. IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, 1993, pp. 665–685.

[4] Sivanadam, S. N.—Sumathi, S.—Deepa, S. N.: Introduction to Neural Networks Using Matlab 6.0. Tata McGraw Hill, NewDelhi 2006.

[5] Khoshgoftaar, T. M.—Allen, E. B.—Hudephol, J. P.—Aud, S. J.: Application of Neural Networks to Quality Modeling of a Very Large Telecommunication System. IEEE Transactions on Neural Networks, Vol. 8, 1997, pp. 902–909.

[6] Tagi, A. I.—Khoshgoftaar, M.—Abran, A.: Can Neural Networks be Easily Interpreted in Software Cost Estimation? IEEE Transactions on Software Engineering, 2002, pp. 1162–1167.

[7] Aggarwal, K. K.—Singh, Y.—Chandra, P.—Puri, M.: Measurement of Software Maintainability Using a Fuzzy Model. Journal of Computer Sciences, Vol. 1, 2005, pp. 538–542.

[8] Gyimothy, T.—Ferenc, R.—Siket, I.: Empirical Validation of Object Oriented Metrics on Open Source Software for Fault Prediction. IEEE Trans. Software Engineering, 2005, Vol. 31, pp. 897–910.

[9] Graves, T. L.—Karr, A. F.—Marron, J. S.Siy, H.: Predicting Fault Incidence Using Software Change History. IEEE Transactions on Software Engineering, Vol. 26, 2000, pp. 653–661.

[10] Khatatneh, K.—Mustafa, T.: Software Reliability Modeling Using Soft Computing Technique. European Journal of Scientific Research, Vol. 26, 2009, No. 1, pp. 154–160.

[11] Kehan, G.—Khoshgoftaar, T. M.—Wang, H.—Seliya, N.: Choosing Software Metrics for Defect Prediction: An Investigation on Feature Selection Techniques. Software Practice and Experience, Vol. 41, 2011, pp. 579–606.

[12] Fenton, N. E.—Neil, M.: A Critique of Software Defect Prediction Models. IEEE Transactions on Software Engineering, Volume 25, 1999, pp. 675–689.

[13] Khoshgoftaar, T. M.—Munson, J. C.: Predicting Software Development Errors Using Complexity Metrics. IEEE J. Selected Area in Comm., Vol. 8, 1990, No. 2, pp. 253–261.

[14] Ohlsson, N.—Alberg, H.: Predicting Error-Prone Software Modules in Telephone Switches. IEEE Trans. Software Eng., Vol. 22, 1996, No. 12, pp. 886–894.

[15] Chidamber, S.—Kemerer, C.: Towards a Metrics Suite for Object Oriented Design. Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA '91), SIGPLAN Notices, Vol. 26, 1991, No. 11, pp. 197–211.

[16] Aggarwal, K. K.—Singh, Y.—Kaur, A.—Malhotra, R.: Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics. Proceedings of World Academy of Science, Engineering and Technology 2006, pp. 144–144.

[17] Yuan, X.—Khoshgoftaar, T. M.—Allen, E. B.—Ganesan, K.: An Application of Fuzzy Clustering to Software Quality Prediction, Proceedings of IEEE Symposium on Application Specific Systems and Software Engineering Technology 2000, pp. 85–90.

[18] Aggarwal, K. K.—Singh, Y.—Chandra, P.—Puri, M.: Measurement of Software Maintainability Using a Fuzzy Model, Journal of Computer Sciences, No. 1, Vol. 4, 2005, pp. 538–542.

[19] Kumar, V.—Sharma, A.—Kumar, R.—Grover, P. S.: Quality Aspects for Component-Based Systems: A Metrics Based Approach. Software: Practices and Experience, John Wiley & Sons 2012, DOI: 10.1002/spe.1153.

[20] Kothari, C. R.: Research Methodology. Methods and Techniques, New Age International Limited 2009.

[21] Challagulla, V. U. B.—Bastani, F. B.—Paul, I. Y.: Empirical Assessment of Machine Learning Based Software Defect Prediction Techniques. Proc. of 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS) 2005, pp. 263–270.

[22] Zadeh, L. A.: Fuzzy Sets. Journal of Information and Control, Vol. 8, 1965, pp. 338–353.

[23] Abraham, A.: Rule Based Expert Systems. Handbook for Measurement Systems Design, Peter Sydenham and Richard Thorn (Eds.), John Wiley and Sons Ltd., London 2005, pp. 909–919.

[24] Abraham, A.: Artificial Neural Networks. Handbook for Measurement Systems Design, Peter Sydenham and Richard Thorn (Eds.), John Wiley and Sons Ltd., London 2005, pp. 901–908.

[25] Han, J.—Kamber, M.: Data Mining: Concepts and Techniques. Harchort India Private Limited 2001.

Vijai Kumar is working as software architect in product engineering services division, Aricent Technologies, Gurgaon, India. He holds an M. Tech. in computer science and engineering from G. J. University Hisar, Haryana, India. He has research and work experience of 10 years in Linux, RTOS, LTE/3G/4G, IP stack, LTE/IMS based UE testing framework, ATCA based systems, High Availability, SIP, Voice Quality Enhancement in GSM and LTE network, Optical Transport Technologies, Device Drivers, Embedded Linux, software and systems engineering, embedded systems, Telecom networks, defense applications and products, Software Defined Radio, MANET. His research interests include software engineering, soft computing and telecom network technologies. He is leading the research activities in his current role.

Arun Sharma is working as Professor and Head of Department of Computer Science and Engineering in Krishna Institute of Engineering and Technology, Ghaziabad, India. He holds an M. Tech. (CSE) from Punjabi University and Ph. D. in software engineering from Thapar University, Patiala, India. He has more than 15 years of teaching and research experience. His research interests include software engineering and database techniques. He has published a number of research papers in reputed journals and conferences including ACM, Elsevier, Springer, Wiley, IJSEKE and others. He is on Editorial Board of several national and International journals. He is an active participant of various conferences as keynote speaker, General Chair and Program Committee member. He is senior member of IEEE and CSI.

Rajesh Kumar is currently an Associate Professor in Thapar University, Patiala. He obtained his M. Sc., M. Phil. and Ph. D. degrees from IIT Roorkee. He has more than 15 years of UG & PG teaching and research experience. He has more than 70 research papers to his credit in various international and national Journals and conferences. He is a member of various professional bodies/societies of repute. His area of research and interest includes mathematical modeling, software engineering, and pattern recognition.