Global Journal of Computer Science and Technology: G Interdisciplinary
Volume 15 Issue 2 Version 1.0 Year 2015
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Evolutionary Computing based an Efficient and Cost Effective Software Defect Prediction System

By Racharla Suresh Kumar & Prof. Bachala Sathyanarayana
Sri Krishnadevaraya University, India

Abstract- The earlier defect prediction and fault removal can play a vital role in ensuring software reliability and quality of service. In this paper Hybrid Evolutionary computing based Neural Network (HENN) based software defect prediction model has been developed. For HENN an adaptive genetic algorithm (A-GA) has been developed that alleviates the key existing limitations like local minima and convergence. Furthermore, the implementation of A-GA enables adaptive crossover and mutation probability selection that strengthens computational efficiency of our proposed system. The proposed HENN algorithm has been used for adaptive weight estimation and learning optimization in ANN for defect prediction. In addition, a novel defect prediction and fault removal cost estimation model has been derived to evaluate the cost effectiveness of the proposed system. The simulation results obtained for PROMISE and NASA MDP datasets exhibit the proposed model outperforms Levenberg Marquardt based ANN system (LM-ANN) and other systems as well. And also cost analysis exhibits that the proposed HENN model is approximate 21.66% cost effective as compared to LM-ANN.

Keywords: *software defect prediction, artificial neural network, adaptive genetic algorithm, levenberg marquardt, object oriented software metrics, cost estimation.*

GJCST-G Classification: D.4.8

Evolutionary Computing based an Efficient and Cost Effective Software Defect Prediction System

Racharla Suresh Kumar [α] & Prof. Bachala Sathyanarayana [σ]

Abstract- The earlier defect prediction and fault removal can play a vital role in ensuring software reliability and quality of service. In this paper Hybrid Evolutionary computing based Neural Network (HENN) based software defect prediction model has been developed. For HENN an adaptive genetic algorithm (A-GA) has been developed that alleviates the key existing limitations like local minima and convergence. Furthermore, the implementation of A-GA enables adaptive crossover and mutation probability selection that strengthens computational efficiency of our proposed system. The proposed HENN algorithm has been used for adaptive weight estimation and learning optimization in ANN for defect prediction. In addition, a novel defect prediction and fault removal cost estimation model has been derived to evaluate the cost effectiveness of the proposed system. The simulation results obtained for PROMISE and NASA MDP datasets exhibit the proposed model outperforms Levenberg Marquardt based ANN system (LM-ANN) and other systems as well. And also cost analysis exhibits that the proposed HENN model is approximate 21.66% cost effective as compared to LM-ANN.

Keywords: software defect prediction, artificial neural network, adaptive genetic algorithm, levenberg marquardt, object oriented software metrics, cost estimation.

I. Introduction

With the increase in information technologies and associated software applications, the inevitable requirement of software reliability has alarmed scientific societies, industries as well as academician to develop certain optimal paradigm to ensure defect free software applications for long run reliability.

Furthermore, the cost factor for software products and services also suggests the defect free software solutions, so as to eliminate probability of faults in future and iterative maintenance. In order to accomplish these objectives, the efficient software defect prediction (SDP) systems are of great significance. In order to ensure optimal software reliability, the defect prediction has become an inevitable part of software development life cycle (SDLC) that intends to eliminate the probability of software failure in run time. The earlier defect prediction can enable software professional to identify fault-prone modules and thus can debug the defects to ensure quality of service provisioning. In recent years the application of open source software has increased tremendously and professional prefer to customize software modules and implement as per need. Still, these modules are prone to defect in real time scenarios, thus demanding for fault prediction and verification [1, 2, 3, 4] before introducing product to the users. The SDP might be functional on the basis of certain software metrics [3, 4, 5] like changes in source code, earlier defect or fault etc. Typically, software metrics do represent certain quantitative factor that characterizes the properties of software source code, which can be employed to predict fault proneness of software during function. On the other hand, in recent years majority of software applications are being developed using Object-Oriented (OO) paradigm. The object oriented paradigm enables certain metrics that that can be employed to examine the quality of software application and associated fault proneness. Some of the predominantly proposed software metrics are MOOD [6], QMOOD [7], Bieman and Kang [8], Briand et al. [9], Etzkorn et al. [10], Halstead [11], Henderson-sellers [12], L and H metrics suite [13], McCabe [14], Tegarden et al. [15], Lorenz and Kidd [16] and CK metric suite [17]. The implementation of object oriented metrics enables software practitioners to examine quality of software in terms of precision, accuracy, fault-resilience, reliable functionality, adaptability, supportability, usability, portability, and cost effectiveness etc. In fact, it makes testing enhanced for large scale software applications. This is the matter of fact that a number of researches have been made for defect prediction. Some of the predominantly employed SDP techniques are based on machine learning and artificial neural network [18, 19, 20, 21, 22], clustering techniques, statistical method, data mining based fault identification, random forest [23, 24, 25] approaches etc. However, the emerging software complexities, critical software applications, reliable service assurance, quality oriented service provisioning, and cost effective or economical solutions etc., motivate researchers to develop certain

*Author α: Research Scholar, Department of Computer Science, Sri Krishnadevaraya University, Andhra Pradesh, India.
e-mail: suresh_sku@yahoo.com*
*Author σ: Professor, Department of Computer Science, Sri Krishnadevaraya University, Andhra Pradesh, India.
e-mail: bachalasatya@yahoo.com*

cost effective defect prediction solution. In recent years, primarily, support vector machine (SVM) and artificial neural network (ANN) approaches are being explored for SDP utilities. The emergence of artificial intelligence based applications have motivated researchers to explore ANN based defect prediction that works based on the human brain functions, while encompassing multiple neurons and directed edges possessing certain weights values between input and output layers. In fact, ANN is a complex non-linear mapping process that employs output as the input for learning certain complex non-linear input-output relationship between input and output layers. In function ANN encompasses data sets to optimize key factors such as weight parameters, risk minimization mechanism for stopping training once the learning error enters in expected margin level. Although, ANN has established itself as a potential candidate for prediction and classification applications, still its limitations in terms of slow learning ability, local minima and convergence can't be ignored. In order to enhance the performance of ANN based defect prediction some researchers [26, 27] have suggested evolutionary computing paradigm that could enable optimal classification and prediction without introducing any computational complexity and premature convergence.

Considering efficiency of evolutionary computing techniques such as Genetic Algorithm (GA) in this paper a robust Adaptive genetic algorithm based ANN learning algorithm has been developed, which has been used for software defect prediction. In addition, to enhance the performance of GA for huge data elements and efficient performance, the genetic parameters (crossover and mutation probability) have been selected dynamically that makes overall system much robust as compared to conventional approaches. In order to examine the performance of the proposed HENN system, a Levenberg Marquardt based ANN (LM-ANN) algorithm has been developed and the comparative performance analysis with the object oriented software metrics, CK metrics [17] has revealed that the proposed HENN algorithm provides better fault detection as compared to LM-ANN. Furthermore, the fault removal cost analysis for both the algorithms has stated that the proposed system is cost effective and can be used for real time defect prediction utilities.

The remaining sections discusses, related work in Section II, the research contributions and problem definitions for the proposed software defect prediction model are presented in III, which has been followed by proposed HENN and LM-AMM based SDP model discussion and implementation in Section IV. Section V presents the results and analysis and conclusion has been discussed in Section VI. The references used in this paper are given at the last of the manuscript.

II. Related Work

Software reliability is of course an inevitable need for quality service provisioning. The reliability oriented software defect prediction (SDP) has motivated researchers to develop optimal system for cost efficient defect prediction. Researchers examined the relationship between object oriented software metrics and associated faults [28, 29, 30, 31, 32, 33] by means of machine learning algorithms and detected fault proneness of software. To achieve better prediction some other approaches such as decision trees, naïve Bayes, and 1-rule [34] based fault detection scheme were developed, where the standard datasets such as NASA MDP was used to examine classification accuracy of the SDP approaches. Chug et al [35] demonstrated fault identification using data mining and employed conventional J48, Random Forest, and Naive Bayesian Classifier (NBC) schemes for performance comparison but still couldn't employ the benefits of advanced classification approaches. To optimize conventional random forest based defect prediction Pushphavathi et al [36] incorporated a hybrid random forest (RF) and Fuzzy C Means (FCM) clustering model for software defect prediction. Unfortunately, these approaches could not address the issue of unbalanced datasets, which motivated researchers to come up with Adaboost. Nc [37] which implemented a number of class imbalance approaches, re-sampling, threshold variations, and ensemble algorithms. Exploring insight, this approach can be found to be complicate and not a cost effective solution for large scale dynamic data. Researchers used SVM based defect prediction scheme [38, 39] and a dynamic SVM model was proposed that intended to detect faults in source code by means of error data and faulty code execution. In [40, 41]an ANN based defect prediction model was developed. A defect severity model using conventional back-propagation learning based ANN was developed in [42]. Similarly in [43] a Radial Basis ANN was used for SDP. ANN based SDP for Halstead data metrics has been done in [44]. In [45] the Bayesian Regularization (BR) technique based ANN model was developed for software fault detection. Almost all ANN based defect prediction model employs conventional learning and weight estimation techniques that confines applicability with huge datasets with dynamic functional environment. The conventional learning and weight estimation approaches can't eliminate the key issues of local minima and convergence issue that limit the performance of generic ANN. The enhancement of learning scheme and further optimization through certain evolutionary computing approaches can make ANN robust for SDP applications. In fact, cost feasibility is one of the key factors that decide employability of certain SDP model, but till no any research work has addressed the issue of cost estimation of the defect prediction model. This paper has considered these limitations as motivation and has developed an evolutionary computing A-GA based SDP model which has been compared with Levenberg Marquardt based ANN and respective fault removal cost estimation has been done.

III. Our Contribution

In SDLC the fault resilience and reliability is of great significance. The implementation of efficient SDP strengthensearly fault detection and thus it enables software practitioner to remove faults to ensure reliability and QoS of the software solution. The predominant question in this paper is whether the implementation of Adaptive Genetic Algorithm can enable efficient and cost effective SDP solutions? In this paper, object oriented software metrics [17] has been considered for defect prediction and using proposed SDP models, the fault proneness of metrics data has been retrieved, whether the data is faulty or non-faulty. In order to perform classification of faulty and non-faulty data, initially the conventional ANN learning scheme with Leven berg Marquardt (LM) algorithm [45] has been developed and respective performance towards software defect prediction with NASA defect datasets has been done. This is the matter of fact that LM based ANN performs better as compared to other approaches such as back-propagation or feed-forward learning based NN, still it suffers due to prime limitations of ANN, such as local minima and weight update issues. Thus, considering higher employability of artificial intelligence techniques and respective limitations for critical software applications, in this paper an evolutionary computing based optimization scheme called Genetic Algorithm has been used for weight estimation during ANN learning. Further to ensure optimal performance of GA, in this paper a novelty has been introduced in terms of adaptive GA parameter (Crossover and Mutation probability) selection. The proposed Adaptive Genetic Algorithm (A-GA) performs adaptive weight estimation and learning optimization so as to ensure optimal fault classification and accuracy. The A-GA optimization scheme alleviates the issue of premature convergence and local minima. Such enhancement has lead better classification and accuracy for fault detection in huge datasets.

In order to examine the performance of the proposed SDP model, the object oriented software metrics (here, CK metrics [17]) has been considered. The implemented metrics characterizes various software features. In this paper, six predominant software metrics have been considered in fault identification. The considered metrics are WMC, NOC, DIT, CBO, RFC, and LCOM. The individual metrics has been feed as the input of the ANN and performing learning with the proposed HENN model the classification for faults has been done. The discussion of the proposed A-GA based ANN (HENN) has been discussed in the next section of the presented manuscript. In this paper, in order to examine the cost effectiveness of the developed SDP models, certain cost efficiency model can be used [46, 47, and 48] and with certain standard threshold the applicability of the proposed SDP model for large scale software data can be examined. The performance analysis of the proposed model has been done in terms of accuracy, precision, recall, F-Measures and fault removal cost efficiency. The discussion of the proposed SDP models and its implementation is discussed in the following sections.

IV. System Model

In this section, the proposed Levenberg Marquardt learning based ANN and our proposed HENN based software defect prediction schemes and its algorithmic implementation have been discussed.

a) Artificial Neural Network based Software Defect Prediction

This is the matter of fact that the Artificial Neural networks (NN) have seen an explosion of interest over the years, and it has been implemented across a range of problem domains, specifically classification and prediction. In fact, the major problems dealing with prediction and classification, ANN is considered to be the dominating solution. For SDP scenario, ANN can be used with different learning schemes like Gradient Descent (GD), Gauss Newton, and Levenberg Marquardt (LM) etc. Unfortunately majority of existing learning paradigm are ineffective to alleviate the key limitations of ANN such as local minima and convergence issue. Even though, researches have revealed that Levenberg Marquardt (LM) can be a potential candidate for ANN learning due to its stable nature and flexible implementation. In this paper, in addition to LM-ANN algorithm, an evolutionary computing technique called Adaptive Genetic Algorithm (A-GA) has been used for dynamic weight estimation for prediction enhancement. In the proposed ANN model and ultimately intended SDP system, it has been intended to find relation between object oriented software metrics and fault prone classes of the six CK metrics; WMC, NOC, DIT, RFC, CBO, LCOM, which has been considered as independent variable. The fault data has been taken as the dependent data. Figure-1 illustrates the architecture of our proposed ANN model comprising three layers i.e., input layer, hidden layer and output layer. Here, 6 input nodes have been defined that takes six CK matrix [17] having multiple classes as individual input. Since, in the proposed ANN model, the expected outputs are either FAULTY or NO-FAULTY, therefore only one output node is needed. Here, we have considered 8 hidden layers so as to avoid unwanted computational complexity. Thus in the defined ANN architecture, 56 weights $(input\ node +$

$Output\,Node) * hidden\,node)$ are required to be estimated for fault prediction and classification. At the input layer, the linear activation function has been used that enables the output of the output layer same as the input of the input layer($O_o = I_i$).

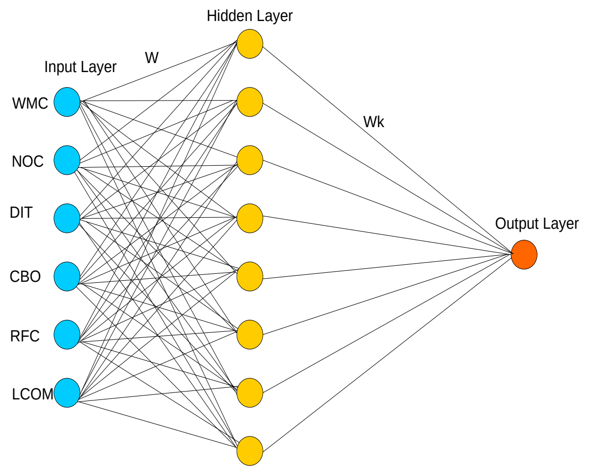

Figure 1 : ANN model for Defect prediction

In our model, the sigmoid function has been used at the hidden layerO_h and thus the output of the hidden nodes O_h with inputI_h would be $O_h = \frac{1}{1+e^{-I_h}}$. The final output at the output node come of output nodes O_o can be obtained as mathematically by $O_o = \frac{1}{1+e^{-O_i}}$.

Generally, the ANN model is defined in terms of a function$Y' = f(W, X)$where Y'states for the output vector and$W$ and Xrepresent the weight vector and the input vector respectively. In learning process, the weight factor $W$is updated iteratively so as to minimize the Root Mean Square Error (RMSE), which can be estimated by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i' - y_i)^2 \qquad (3)$$

Where $y$depicts the actual output andy_i' represents the expected output.

In order to make computation efficient and to process multidimensional data with ANN, it is inevitable to perform the normalization. In the proposed ANN based SDP models; the data normalization has been done using Min-Max approach, which is discussed as follows:

i. *Data normalization*

In this paper, normalization has been performed on the defect datasets that strengthens the proposed ANN based software detect prediction systems for better readability and classification. In the proposed SDP model, the data normalization has been done over the range of [0, 1] so as to adjust the defined range of input feature value and avoid the saturation of neurons. There a number of normalization approaches such as Min-Max normalization, Z-Score normalization and decimal scaling etc. We have normalized the defect data using Min-Max normalization scheme that performs a linear transformation on the original data and then maps individual data x_i of attribute X to the normalized value x'_iin the range of [0, 1]. The normalization using Min-Max approach has been done using following equation:

$$Normalized(x_i) = x_i'' = \frac{x_i - min(X)}{max(X) - min(X)} \qquad (4)$$

where $max(X)$ and $min(X)$ are the maximum and minimum values of the attribute X respectively. In the proposed SDP model, performing data normalization the ANN model has been implemented for fault classification.

In ANN based systems, the efficient weight estimation and learning approach is of great significance. Till a number of approaches have been developed for learning optimization in ANN based artificial intelligence applications. Some of the predominant approaches are: Gauss Newton, Gradient descent, Levenberg Marquardt (LM) etc. Interestingly LM can work as both gradient descent as well as gauss Newton. Some researchers also have advocated that LM can outperform other existing learning schemes in ANN. Thus considering significance of LM for effective learning for SDP, in this paper initially LM based ANN (LMANN) has been developed for SDP model. The discussion of the proposed LMANN model for SDP application is given as follows:

b) *Levenberg Marquardt (LM) Learning based ANN for Software Defect Prediction*

The prime scope for ANN optimization is the enhancement of its weight estimation and respective learning optimization. Therefore, considering these factors, a number of algorithms have been proposed for weight update in ANN learning (Table-1). In this paper, considering the higher efficiency of Levenberg Marquardt (LM) algorithm, we have used this algorithm for weight update (W) during ANN training for defect prediction.

Table 1 : Specifications of varied Weight Update algorithms

Algorithm	Weight Update Rules	Convergence	Computation Complexity
EBP Algorithm	$W_{k+1} = W_k - \alpha g_k$	Stable, Low	Gradient
Newton Algorithm	$W_{k+1} = W_k - {H_k}^{-1} g_k$	Unstable, Fast	Gradient and Hessian
Gauss-Newton Algorithm	$W_{k+1} = W_k - \left({J_k}^T J_k\right)^{-1} J_k e_k$	Unstable, Fast	Jacobian
Levenberg-Marquardt Algorithm	$W_{k+1} = W_k - \left(J_k^T J_k + \mu I\right)^{-1} J_{k^e k}$	Stable, Fast	Jacobian
NBN Algorithm	$W_{k+1} = W_k - {Q_k}^{-1} g_k$	Stable, Fast	Quasi Hessian

Levenberg Marquardt (M) algorithm performs localization of the bare minimum value of multivariate function in a repetitive manner, which is expressed as the sum of squares of non-linear real-valued functions. Similar to GD algorithm, in HENN, LM algorithm updates the weights during NN learning process. Considering the performance novelty, the proposed LM algorithm comprises the functional ability of Steepest Descent and Gauss Newton method. The proposed LM algorithm can update the weight vector by following expression:

$$W_{k+1} = W_k - (J_k^T J_k + \mu I)^{-1} J_{k^e k} \qquad (1)$$

Where W_{k+1} is the updated weights, W_k is the current weights, I represents the identity or unit matrix, J is the Jacobian matrix andμ, the combination coefficient is always positive. With μ as very small, it functions as Gauss Newton method while making μ as very large makes it functional as Gradient descent method. The Jacobian matrix derived as given as:

$$J = \begin{bmatrix} \frac{d}{dW_1}(E_{1,1}) & \frac{d}{dW_2}(E_{1,1}) & \cdots & \frac{d}{dW_N}(E_{1,1}) \\ \frac{d}{dW_1}(E_{1,2}) & \frac{d}{dW_2}(E_{1,2}) & \cdots & \frac{d}{dW_N}(E_{1,2}) \\ \vdots & \vdots & & \vdots \quad \vdots \\ \frac{d}{dW_1}(E_{P,M}) & \frac{d}{dW_2}(E_{P,M}) & \cdots & \frac{d}{dW_N}(E_{P,M}) \end{bmatrix} \qquad (2)$$

Where N refers the weight counts and the input patterns are P. The output patterns are indicated by M. The overall training function by the proposed LM algorithm is presented in the following figure.

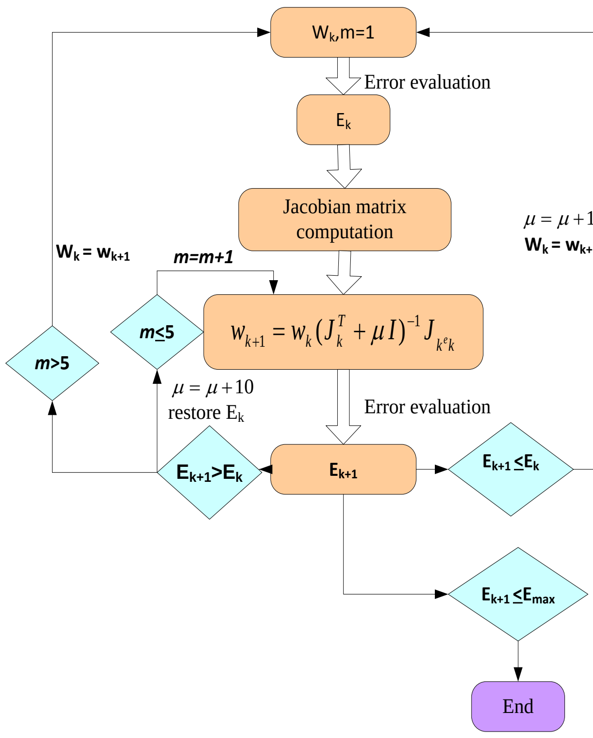

Figure 2 : Levenberg–Marquardt algorithm based HENN training: W_k is the current weight, $W_{k+1}$is the next weight, E_{k+1} is the current total error, and $E_k$is the final error

In the proposed SDP model, in the initial phase the LM algorithm has been used to estimate the weights for the learning scheme. Figure 3 represents the adaptive weight estimation approach using LM algorithm. The weights are updated dynamically so as to reduce RMSE and satisfying the stopping criteria, the classification has been done for fault prediction. On the basis of fault classification, the confusion matrix has been obtained which has been employed further to examine performance of the proposed SDP model.

This is the matter of fact that LM-ANN has been employed for varied classification utilities but considering the specific requirements of fault prediction and robust function with huge data sets in real time software utilities, the local minima problem and convergence issues of ANN can't be ignored. Thus, considering these limitations, in this paper, the evolutionary algorithm Adaptive-Genetic Algorithm (A-GA) has been used for parameter optimization that can strengthen the function of the proposed system to yield more precise, accurate and efficient outputs. The implementation of A-GA for ANN based SDP utility has been discussed in the following section

c) *HENN: Hybrid Evolutionary Computing Based Neural Network for Software Defect Prediction*

In recent years a number of optimization schemes have been developed on the basis of the concept of human evolution and Genetic Algorithm (GA) is one of the predominant one. GA is an adaptive search approach based on the evolutionary concepts of natural selection that intends to find certain optimal or near optimal solutions. In fact, the basic concept of GA is based on the philosophy of natural selection and Darwin principle of the survival of fittest. In function, GA at first performs random population generation, where population represents certain set of solutions. In fact, these solutions are nothing else but a chromosome possessing a form of binary strings where all the comprising parameters are supposed to be encoded. Performing population generation, GA calculates the fitness value, also known as fitness function for the individual chromosome. The fitness value represents a user-defined function that provides the estimation results for individual chromosome, and thus a higher fitness value signifies the chromosome to be the dominant one. On the basis of retrieved fitness values, the offspring are generated by means of genetic operators called crossover and mutation. Implementing genetic operators the population generation continues until the stopping criteria is achieved. Here, it must be noted that after every generation, chromosomes having fitness value more than defined threshold are considered for next generation otherwise are mutated out of competition.

As depicted in Figure-1, the developed HENN model [59] encompasses $i-h-o$ network configuration having$i$ input layer, h hidden layer and O output layer or nodes. In the proposed ANN model, all the six CK metrics under consideration have been fed as input to the individual input nodes, where the individual metrics can have multiple classes depending on the size of software and dimensions. As already discussed with the considered 6-8-1 ANN configuration, the total number of weights, N to be calculated are:

$$N = (i+O)*h \tag{5}$$

In the proposed model the individual weight is considered as a gene in the chromosomes and is a real number. Consider$l$, the gene length or the number of digits be$l$, then the length of the chromosome L_{Chrom} can be obtained using following equation:

$$L_{Chrom} = N*l = (i+O)*h*l \tag{6}$$

In the proposed A-GA based scheme all chromosomes are considered as the population and for each chromosomes the fitness values and weights are estimated. In our proposed model, the weights (W_k) has been obtained using following equation:

$$W_k = \begin{cases} \text{if } 0 \le x_{kl+1} < 5 \\ -\dfrac{x_{kl+2*}10^{l-2} + x_{kl+3*}10^{l-3} + \cdots + x_{(k+1)l}}{10^{l-2}} \\ \text{if } 5 <= x_{kl+l} <= 9 \\ +\dfrac{x_{kl+2*}10^{l-2} + x_{kl+3*}10^{l-3} + \cdots + x_{(k+1)l}}{10^{l-2}} \end{cases} \tag{7}$$

To perform A-GA based weight estimation in ANN, the fitness values for individual chromoseomes are needed to be obtained. The algorithm developed for fitness value estimation is given in the following figure (Figure-3).

Algorithm for Fitness Estimation

Input:$\bar{I}_i = (I_{1i}, I_{2i}, I_{3i}, \cdots, I_{li})$

Output:$\bar{T}_i = (T_{1i}, T_{2i}, T_{3i}, \cdots, T_{ni})$

Where $\bar{I}_i, \bar{T}_i$state the input and output pairs of the $i - h - o$ configuration of neural network.

Phase-1 :Calculate weights $\bar{W}_i$for$C_i$by:

$$W_k = \begin{cases} \text{if } 0 \le x_{kd+1} < 5 \\ -\dfrac{x_{kd+2*}10^{d-2} + x_{kd+3*}10^{d-3} + \cdots + x_{(k+1)d}}{10^{d-2}} \\ \text{if } 5 <= x_{kd+d} <= 9 \\ +\dfrac{x_{kd+2*}10^{d-2} + x_{kd+3*}10^{d-3} + \cdots + x_{(k+1)d}}{10^{d-2}} \end{cases}$$

Phase-2:Assuming$\bar{W}_i$be a constant weight, perform trainingof N input instances and calculate output O_i

Phase-3:CalculateMSE E_j for all input instance j, $E_j = (T_{ji} - O_{ji})$

Phase-4: Calculate RMSE of chromosomeC_i

$$E_i = \sqrt{\frac{\sum_{j=1}^{j=N} E_j}{N}}$$

Where N is the number of training data

Phase-5: Calculate the fitness value for chromosomeC_i

$$F_i = \frac{1}{E_i} = \frac{1}{\sqrt{\frac{\sum_{j=1}^{j=N} E_j}{N}}}$$

Figure 3 : Fitness generation using A-GA

Genetic algorithm (GA) has been considered as a potential global optimization approach for major applications; still this approach can be further optimized to alleviate issues of premature convergence. In this paper, in order to alleviate these issues, the genetic parameters, cross over probability (P_c) and mutation probability (P_m) has been selected dynamically so as to get optimal or sub-optimal solution efficiently without converging. To update P_c and $P_m$the following mathematical equations has been used:

$$(P_c)_{k+1} = (P_c)_k - \frac{C_1 * n}{5}$$

$$(P_m)_{k+1} = (P_m)_k - \frac{C_2 * n}{5} \qquad (8)$$

where $(P_c)_{k+1}$ and $(P_m)_{k+1}$denote the updated crossover probability and mutation probability respectively. The other variables $(P_c)_k$ and $(P_m)_k$are the current crossover and mutation probability, C_1 and C_2 can be any positive constant and$n$represents the number of chromosome having similar fitness value. In the proposed HENN model, the A-GA continues functioning till 95% of chromosomes are having similar fitness value. Once the stopping criterion is achieved the A-GA terminates and the final output at output layerO_o is obtained.If the final estimated output is more than 0.5, it signifies class as FAULTY otherwise NON-FAULTY. On the basis of retrieved FAULTY and NON-FAULTY data, a confusion matrix is obtained, which is further used for performance assessment. Figure-4 represents the flow diagram of the proposed HENN based SDP model.

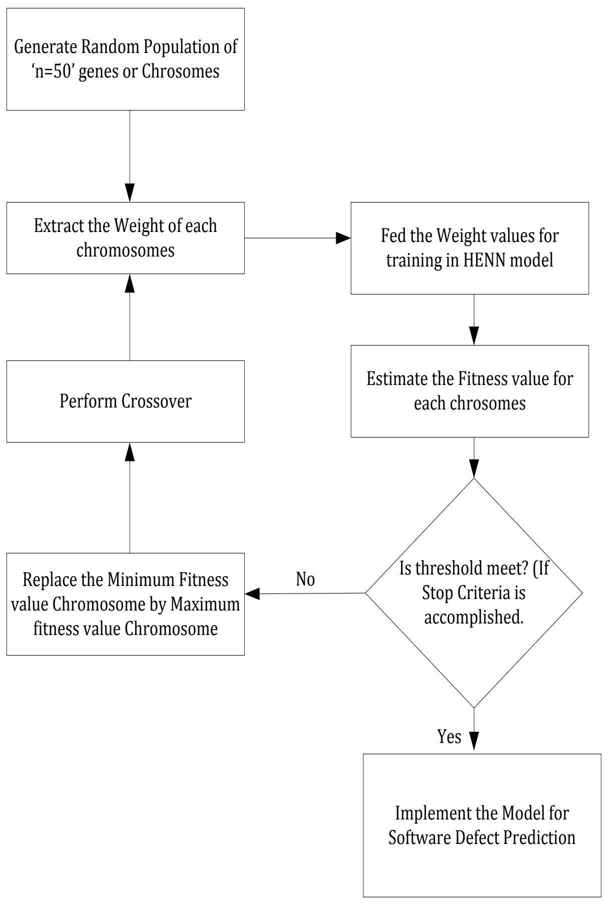
Generate Random Population of 'n=50' genes or Chrosomes
Extract the Weight of each chromosomes
Fed the Weight values for training in HENN model
Perform Crossover
Estimate the Fitness value for each chrosomes
Replace the Minimum Fitness value Chromosome by Maximum fitness value Chromosome
No
Is threshold meet? (If Stop Criteria is accomplished.
Yes
Implement the Model for Software Defect Prediction

Figure 4 : Proposed HENN Scheme for Software Defect Prediction

The overall discussion of the proposed HENN model is given as follows:

- *HENN-SDP Simulation*

Since, the proposed HENN model operates on the basis of genetic algorithm principle; it also encompasses processes such as, population generation, selection, crossover, fitness estimation, and mutation. A brief discussion of the implemented HENN simulation model is given as follows:

Step-1 Population Initialization: In our model randomly 50 chromosomes are selected randomly to perform competition. These randomly selected chromosomes perform crossover with defined crossover and mutation probability.

Step-2 Weight Estimation: HENN estimates weight W_k for each selected chromosomes as input to the hidden layer and hidden layer to the output layer using equation (7).

Step-3 Fitness Estimation: On the basis of weight estimated, the fitness value is obtained for individual chromosome with an intention to minimize the root mean square error (RMSE) obtained at the output node of ANN.

Step-4 Chromosome Ranking and Mutation: On the basis of fitness values for the individual chromosomes, the ranking is performed which is followed by mutation of the chromosomes having lower fitness values and chromosomes with higher ranking replaces chromosomes with lower fitness.

Step-5 Crossover: In the proposed HENN model, the two point crossover is performed with the selected chromosomes. Here to enhance computational efficiency the GA parameters, P_c and P_m are varied adaptively, as per equation(6). Initially, P_c and P_m have been assigned as 0.6 and 0.1 respectively and n refers the number of chromosome having similar fitness value.

- *Stopping Criteria:* The process of weight estimation using HENN algorithm continues till the stopping criteria is not achieved and the 95% chromosomes in gene pool achieves unique fitness value, as beyond it the fitness level of chromosomes get saturated.

Step 6 Fault Classification: Considering step-3, and stopping criteria, with the optimal RMSE, the final output at output layer of ANN is obtained that more than 0.5 signifies towards FAULTY class otherwise NON-FAULTY.

Step 7 Confusion Matrix: On the basis of FAULTY and NON-FAULTY label of comprising classes, a Confusion Matrix is derived that is used for performance evaluation.

Thus, implementing the above mentioned approaches, the proposed HENN model performs Software Defect Prediction.

This is the matter of fact that a number of SDP systems have been developed but only prediction accuracy and precision can't be the justification for a system to be employable in real time scenarios. Industries demands for certain cost effective and efficient system for defect prediction. A system with higher computational efficiency with minimal cost of fault detection and removal can be of great significance and can be suggested to be used in real time SDP applications.

Thus, considering the need of a novel cost analysis mechanism, in this paper a novel cost estimation approach has been developed which has been used to assess the computational (Fault detection and removal) cost analysis for both our proposed HENN based SDP as well as reference, LM-ANN based SDP model. The discussion of the proposed cost estimation model is given as follows:

d) Software Fault Estimation and Removal Cost analysis

In this paper, a novel cost estimation approach has been developed that estimates the cost of fault detection and removal, as the efficiency to be considered as a criterion that decides whether the system should be used or not in real time applications. The proposed cost estimation model has been derived from [46]. In the developed cost estimation approach, certain constraints have been assumed such as, varied testing phases might take different cost for certain fault removal as different softwares are developed in varied software platform and with varied development standards, and it is impractical to perform comprising unit testing on all the associated modules [47]. In the proposed cost estimation model, the identification efficiency model proposed in [48] has been incorporated that suggests following efficiencies to be used for cost estimation model.

Table 2 : Cost Estimation for different testing approaches (Staff hour per faults)

Testing	Min	Max	Median
Unit	1.5	6	2.5
System	2.82	8.37	6.2
Field	3.9	27.24	27

In this paper, the following notations have been used to formulate mathematical model for fault estimation and removal cost.

Table 3 : Cost Estimation Metrics

$\mathrm{Cost}_{\mathrm{Estm\_SDP}}$	Estimated fault removal cost of the software when fault prediction is performed
$\mathrm{Cost}_{\mathrm{Estm\_WSDP}}$	Estimated fault removal cost of the software without using fault prediction approach
$\mathrm{Cost}_{\mathrm{Norm}}$	Normalized Estimated fault removal cost of the software when fault prediction is utilized
C_i	Initial setup cost of used fault-prediction technique
C_u	Normalized fault removal cost in unit testing
C_S	Normalized fault removal cost in system testing
C_f	Normalized fault removal cost in testing
M_p	percentage of classes unit tested
FP	Number of false positive
FN	Number of false negative
TP	Number of true positive
TN	Number of true negative
TC	Total number of classes
FC	Total number of faulty classes
δ_u	Fault identification efficiency of unit testing
δ_s	Fault identification efficiency of system testing

The derived cost estimation expressions are given as follows:

$$\text{Cost}_{\text{Estm\_SDP}} = C_i + C_u * (FP + TP) + \delta_s * C_s * (FN + (1 - \delta_u) * TP) + (1 - \delta_s) * C_f * (FN + (1 - \delta_u) * TP) \qquad (9)$$

$$\text{Cost}_{\text{Estm\_WSDP}} = M_p * C_u * TC + \delta_s * C_s * (1 - \delta_s) * FC + (1 - \delta_s) * C_f * (1 - \delta_u) * FC \qquad (10)$$

$$\text{Cost}_{\text{Norm}} = \frac{Cost_{Estm\_SDP}}{Cost_{Estm\_WSDP}} = \begin{cases} < 1, Significant\ SDP\ System \\ \geq 1 \quad \text{Not Suitable} \end{cases} \qquad (11)$$

Here, $\text{Cost}_{\text{Estm SDP}}$ represents the estimated fault removal cost for software with fault prediction scheme, $\text{Cost}_{\text{Estm\_WSDP}}$ is the fault removal cost without using any SDP system. The variable $\text{Cost}_{\text{Norm}}$ refers the normalized cost with the SDP models. As illustrated in above expression, the minimal normalized cost signifies better employability of a defect prediction system. In this paper, the cost analysis for both the proposed HENN as well as Levenberg Marquardt based ANN (LMANN) has been done. The results obtained are given in Table 7.

V. Result and Analysis

This section discusses the experimental setup, benchmark fault data, results and performance analysis.

In this paper, the overall algorithms for artificial neural network, Levenberg Marquardt based ANN, Adaptive Genetic Algorithm and its implementation with ANN for defect prediction, etc have been developed using MATLAB2012b software model. In addition, the toolboxes of machine learning and artificial neural network have been considered to perform simulation. In order to examine the performance of the proposed HENN model, object oriented software metrics suite, CK Metrics [17] has been considered, which has been derived from the fault data taken from PROMISE [49] and NASA MDP [50] fault data repository. The software metrics from the fault datasets (*JEdit, Ant, Camel* and *IVY*)have been derived using Chidamber and Kemerer Java Metrics tool (CKJM) tool that extracts software metrics by executing byte code of compiled Java cases and assigns a definite weight of the comprising classes having feature vectors. In this paper, six predominant CK metrics have been considered as depicted in the Table-4.

Table 4 : Object Oriented Software Metrics (CK Metrics [17])

WMC	Overall complexities of the methods in comprising classes
NOC	Number of sub-classes subordinate to a class in the class hierarchy
DIT	Maximum height of the class hierarchy
CBO	Number of other classes to which it is allied with
RFC	A set of approaches that can be executed in response to a message received by an object of that class
LCOM	Dissimilarity measurement of varied methods in a class using instanced attributes/variables

In our work, the six software metrics have been considered as the independent data while the fault data has been taken as dependent variable.

The considered data *JEdit, Ant, Camel* and *IVY* comprise static code measures along with varied modules sizes, defective modules and defect rates. In the proposed SDP models the respective extracted weights and features of the data classes have been taken as input to the ANN as illustrated in Figure-1. On the basis of final outcome of the both SDP models, LM-ANN as well as HENN for individual datasets, the confusion matrix has been obtained. A confusion matrix comprises two rows and columns representingtrue positive (TP), false negatives (FN), false positive (FP) and true Negative variables. The variables in confusion matrix represent the faulty and non-faulty data and its severity. As depicted in Table-5, TP depicts modules which are classified as FAULTY, FN represents the modules which are FAULTY but are classified incorrectly as NON-FAULTY. Similarly, FP represents the modules which are non-faulty but are classified as faulty.

Table 5 : Confusion Matrix

	Predicted Defective	Predicted Defect Free
FAULTY	True Positive	False Negative
NON-FAULTY	False Positive	True Negative

In this paper, the performance of the proposed HENN as well as LM-ANN SDP models has been examined in terms of fault prediction accuracy, precision, F-measure, recall, specification and fault detection and removal cost. The mathematical expression for considered performance parameters are given in Table-6.

Table 6 : Performance Parameters

Construct	Mathematical Expression
Recall	$TP/(TP+FN)$
Precision	$TP/(TP+FP)$
Specification	$TN/(TN+FP)$
F-measure	$2.\frac{Recall.Precision}{Recall+Precision}$
Accuracy	$(TN+TP)/(TN+FN+FP+TP)$

a) Result Analysis

The following section represents the results obtained from the proposed HENN based SDP model and a reference model based on Leven berg Marquardt based ANN. Here, from the results obtained it can be found that the proposed HENN based SDP model performs better than Leven berg Marquardt algorithm based ANN (LMANN). Here, it can be found that the average fault prediction accuracy of the proposed HENN model is 87.23%, on contrary, the LM-ANN based SDP models delivers 75.48% and hence the proposed system outperforms the existing and till most efficient ANN model, LMANN. In addition, the analysis results states that the proposed system provides 98.2% precision, 92.74% F-measure, 88.55% of recall, which is 87.7% 85.7%, and 85.4% for LMANN based SDP system, respectively. The following figures (Figure 5-8) represent the average performance of the proposed system with four benchmark datasets (*JEdit, Ant, Camel* and *IVY*). The performance results for the developed SDP models with individual datasets are given in Table-7. Considering cost effectiveness of HENN and LMANN based SDP models, Figure 9 depicts that the proposed HENN based system is most cost efficient as compared to LMAMM, and hence it can be implemented for real time applications intending software defect prediction and removal.

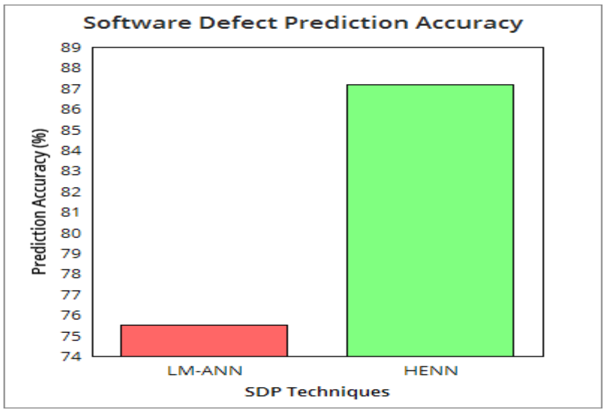

Figure-5 : Accuracy analysis of software defect prediction

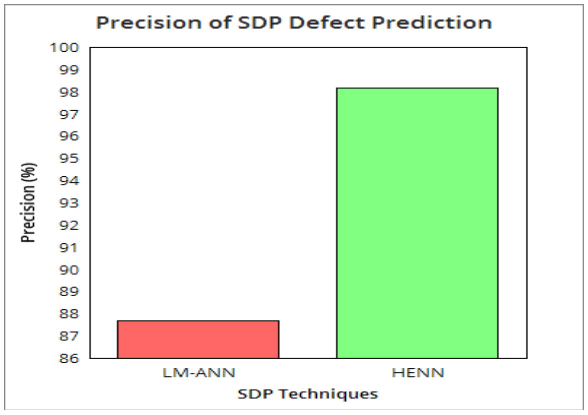

Figure-6 : Precision analysis of Software defect prediction precision

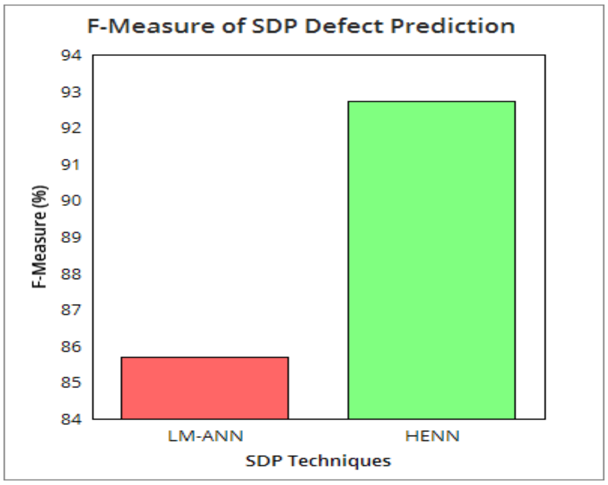

Figure-7 : F-Measure analysis of software defect prediction

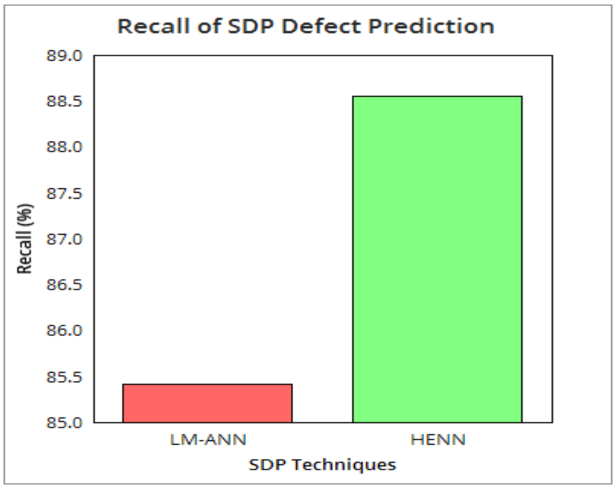

Figure-8 : Recall Analysis of Software defect prediction

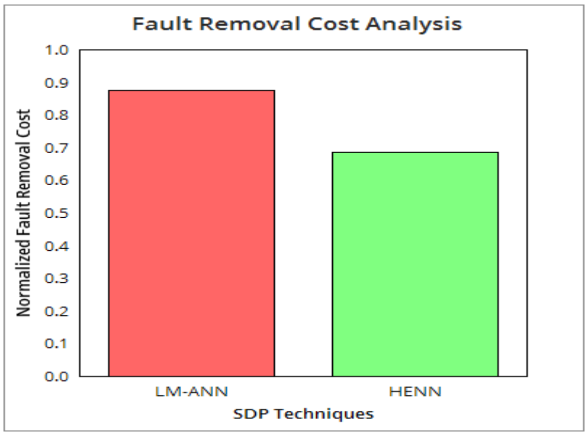

Figure -9 : Fault detection and removal cost analysis

The cost analysis results depict that the proposed HENN based SDP model is approximately 21.66% cost efficient as compared to LMANN based SDP system.

Table 7 : Performance analysis of the proposed HENN model and LM-ANN based SDP system

Data	Modules	Tech.	Accuracy	Precision	F-Measure	Recall	Specification	Norm. Fault Removal Cost (Norm.)
JEDIT	492	HENN	0.9799	1	0.9897	1	0.9756	0.2406
		LMANN	0.8394	0.8503	0.9119	0.9832	0.0526	0.2927
ANT	744	HENN	0.8145	0.9343	0.8867	0.8438	0.6346	0.9149
		LMANN	0.7675	0.9879	0.8684	0.7748	0	0.9763
IVY	352	HENN	0.8835	0.9936	0.9380	0.8883	0.3333	0.7115
		LMANN	0.6278	0.6955	0.7681	0.8577	0.0404	0.8936
CAMEL	965	HENN	0.8114	1	0.8952	0.8102	1	0.8771
		LMANN	0.7845	0.9743	0.8792	0.8011	0	1.3401

Table 7 depicts that the proposed defect prediction approach is highly robust and efficient as compared to Levenberg-Marquardt based ANN system, which is supposed to be the most effective ANN system till. The proposed HENN model has exhibited better cost effectiveness for the fault detection and removal than LMANN. Further to explore effectiveness of the proposed HENN model as compared to other existing systems, a comparison has been done (Table-8) and results revealed that the proposed system can be the best optimal solution for defect prediction for object oriented software applications.

Table 8 : Performance comparison for different SDP schemes

SDP Techniques	Accuracy (%)	Precision (%)	F-Measure (%)
LLE-SVM[51]	81.1	82.5	80.4
SVM [51]	69.4	68.1	69.7
SVM [52]	55.3	88.0	83.2
Natural Gas [57]	94.2	-	-
Symbolic Regression [57]	89.50	-	-
RBP-NN [57]	80.0	-	-
LP [52]	86.6	86.6	87.4
Naive Based [52]	85.6	83.1	83.9
CPSO[53]	69.2	67.6	-
T-SVM [54]	75.8	84.1	80.9
GANN[53]	73.4	81.6	-
AdaBoost [53]	79.1	82.3	-
Random Forest [58]	91.4	-	-
k-NN [56]	91.8	-	-
C4.5 [56]	88.3	-	-

J 48 [56]	90.9		
Levenberg-Marquardt-NN [56]	88.0	-	-
NNEP-Evolutionary [53]	88.8	81.2	-
PSO [55]	78.7	-	-
PSO-NN [57]	97.7	-	-
HENN SDP	**97.9**	1	**98.9**

VI. Conclusion

In order to ensure optimal software reliability and quality of service the earlier prediction of faults and its removal is of great significance. In addition, the cost effective solution for defect prediction and fault removal has motivated industries as well as academician to develop a novel SDP solution that could ensure cost effective and optimal defect prediction solutions. In this paper, an object oriented software matrix based defect prediction model has been developed.

Considering the limitations of artificial intelligence techniques such as artificial neural network, in this paper an evolutionary computing technique named Adaptive Genetic Algorithm (A-GA) has been developed for ANN dynamic weight estimation and learning optimization. The proposed Hybrid Evolutionary computing based Neural Network (HENN) based system has been employed for SDP system. Furthermore, Levenberg Marquardt algorithm based ANN algorithm (LMANN) has been developed for defect prediction. Considering cost effectiveness of the defect prediction systems, a novel mathematical model has been derived and the cost analysis results confirms that the proposed HENN model is cost effective as well as performs better as compared to other existing systems. The simulation results obtained with PROMISE and NASA MDP datasets exhibits that the proposed model performs on average 87.23% accuracy and the best classification accuracy obtained is 97.99% with 100% precision. The proposed model delivers 98.97% of F-measure. The cost analysis exhibits that the proposed HENN model is approximate 21.66% cost effective as compared to LMANN. The comparative analysis in this paper reveals that the proposed HENN model performs better as compared to other existing techniques. This paper could perform cost analysis of only HENN and LMANN, hence in future other defect prediction models can also be examined for their cost effectiveness for real time applications.

References Références Referencias

1. Zuse H., "A Framework of Software Measurement, Walter de Grutger Publish" 1998.
2. University of Texas, Software Quality Institute Report, May 2002.
3. Rosenberg, L., S. B., Sheppard, "Metrics in Software Process Assessment, Quality Assurance and Risk Assessment", 2nd International Symposium on Software Metrics, London, October, 1994.
4. Boehm, B. W., Software Engineering Economics, Prentice-Hall, 1981.
5. L.C. Briand, W.L. Melo, J. Wu st, "Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects," IEEE Trans. Software Eng., vol. 28, no. 7, pp. 706-720, July 2002.
6. F. B. E. Abreu, R. Carapuca, "Object-Oriented software engineering: Measuring and controlling the development process," in Proceedings of the 4th International Conference on Software Quality, vol. 186, 1994.
7. J. Bansiya, C. G. Davis, "A hierarchical model for Object-Oriented design quality assessment," ACM Transactions on Programming Languages and Systems., vol. 128, pp. 4–17, August 2002.
8. B. K. Kang and J. M. Bieman, "Cohesion and reuse in an Object-Oriented system," in Proceedings of the ACM SIGSOFT Symposium on *software reusability*, pp. 259–262, Seattle, March 1995.
9. L. C. Briand, J. Wust, J. W. Daly, D. V. Porter, "Exploring the relationships between design measures and software quality in Object-Oriented systems," *The Journal of Systems and Software*, vol. 51, pp. 245–273, May 2000.
10. L. Etzkorn, J. Bansiya, and C. Davis, "Design and code complexity metrics for Object-Oriented classes," *Object-Oriented Programming*, vol. 12, no. 10, pp. 35–40, 1999.
11. M. Halstead, *Elements of Software Sciencel*. New York, USA: Elsevier Science, 1977.
12. B. Henderson-Sellers, *Software Metrics*. UK: Prentice-Hall, 1996.
13. W. Li and S. Henry, "Maintenance metrics for the Object-Oriented paradigm," in *Proceedings of First International Software Metrics Symposium*, pp. 52–60, 1993.
14. T. J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, vol. 2, pp. 308–320, December 1976.
15. D. P. Tegarden, S. D. Sheetz, D. E. Monarchi, "A software complexity model of Object-Oriented systems," Decision Support Systems, vol. 13, no. 3, pp. 241–262, 1995.
16. M. Lorenz and J. Kidd, Object-Oriented Software Metrics. NJ, Englewood: Prentice-Hall, 1994.

17. S. R. Chidamber and C. F. Kemerer, “A metrics suite for Object-Oriented design,” IEEE Transactions on Software Engineering on June 1994, vol. 20, pp. 476–493.
18. Kutlubay O., A. Bener, “A Machine Learning Based Model for Software Defect Prediction,” working paer, Boaziçi University, Computer Engineering Department 2005.
19. Bo. Yang, Xiang Li, “A study on software reliability prediction based on support vector machines”, The Annual IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1176-1180, 2-4 Dec. 2007.
20. Sandhu, Parvinder Singh, Sunil Kumar, Hardeep Singh, “Intelligence System for Software Maintenance Severity Prediction”, Journal of Computer Science, Vol. 3 (5), pp. 281-288, 2007.
21. Gondra, "Applying machine learning to software fault-proneness prediction," Journal of Systems and Software, vol. 81, no. 2, pp. 186-195, Feb. 2008.
22. Q. P Hu, Y. S. Dai, M. Xie, S. H. Ng., “Early software reliability prediction with extended ANN model,” Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC’06), Vol. 2, pp. 234-239, September 2006.
23. Chug, A., Dhall, S., "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm, "Confluence 2013: The Next Generation Information Technology Summit, pp.173-179, 26-27 Sept. 2013.
24. Armah, G.K., Guangchun Luo, Ke Qin, "Multilevel data preprocessing for software defect prediction," Information Management, Innovation Management and Industrial Engineering (ICIII), 2013 6th International Conference, vol.2, pp.170-174, 23-24 Nov. 2013.
25. Mohamad Mahdi Askari, Vahid Khatibi Bardsiri, “Software Defect Prediction using a High Performance Neural Network”, International Journal of Software Engineering and Its Applications, Vol. 8, No. 12, pp. 177-188, 2014.
26. M. Harman, “Why the Virtual Nature of Software makes it Ideal for Search Based Optimization”, Fundamental Approaches to Software Engineering, 2010.
27. C. Grosan, and A. Abraham, “Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews”, Studies in Computational Intelligence, vol. 75, pp. 1-17, 2011.
28. Saida Benlarbi, Khaled El Emam, Nishith Geol (1999), “Issues in Validating Object-Oriented Metrics for Early Risk Prediction”, by Cistel Technology 210 Colonnade Road Suite 204 Nepean, Ontario Canada K2E 7L5.
29. Lanubile F., Lonigro A., Visaggio G. “Comparing Models for Identifying Fault-Prone Software Components”, Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering, pp. 12-19, June 1995.
30. Fenton, N. E. and Neil, M., “A Critique of Software Defect Prediction Models”, Bellini, I. Bruno, P. Nesi, D. Rogai, University of Florence, IEEE Trans. Softw. Engineering, vol. 25, Issue no. 5, pp. 675-689, 1999.
31. Giovanni Denaro, ”Estimating Software Fault-Proneness for Tuning Testing Activities” Proceedings of the 22nd International Conference on Software Engineering, Limerick, Ireland, June 2000.
32. Manasi Deodhar, “Prediction Model and the Size Factor for Fault-proneness of Object Oriented Systems”, MS Thesis, Michigan Tech. University, Dec. 2002.
33. Bellini, P., “Comparing Fault-Proneness Estimation Models”, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), pp. 205-214, 2005
34. Khoshgoftaar, T.M., K. Gao and R. M. Szabo, “An Application of Zero-Inflated Poisson Regression for Software Fault Prediction. Software Reliability Engineering”, ISSRE 2001. Proceedings of 12th International Symposium, pp: 66 -73, 27-30 Nov. 2001.
35. Chug, A., Dhall, S., "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm," Confluence 2013: The Next Generation Information Technology Summit, pp.173-179, 26-27 Sept. 2013.
36. Pushphavathi, T.P.; Suma, V.; Ramaswamy, V., "A novel method for software defect prediction: Hybrid of FCM and random forest," Electronics and Communication Systems (ICECS), 2014 International Conference, vol., no., pp.1,5, 13-14 Feb. 2014.
37. Wang, S.; Yao, X., "Using Class Imbalance Learning for Software Defect Prediction," Reliability, IEEE Transactions, vol.62, no.2, pp.434-443, June 2013.
38. Brun, Y. and D. E. Michael, “Finding Latent Code Errors via Machine Learning over Program Executions”, Proceedings of the 26th International Conference on Software Engineering, May, 2004.
39. F. Xing, P. Guo, M. R. Lyu, “A novel method for early software quality prediction based on support vector machine,” Software Reliability Engineering, International Symposium, pp. 213–222, 2005.
40. Cai K Y, 0n the Neura1 Network Approach in Software Reliability Modeling, Journal of Systems and Software, pp 47-62, 2001.
41. S.A. Rojas and D. Fernandez-Reyes, “Adapting multiple kernel parameters for support vector machines using genetic algorithms,” The 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 626-631, September, 2005.

42. Jianhong, Z., Sandhu, P.S., Rani, S., "A Neural network based approach for modeling of severity of defects in function based software systems," International Conference on Electronics and information Engineering, vol.2, pp.568- 575, 1-3 Aug. 2010.
43. Jindal, R., Malhotra, R. and Jain, A., "Software defect prediction using neural networks," in 3rd International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), pp.1-6, 8-10 Oct, 2014.
44. Yousef A.H., "Extracting software static defect models using data mining", Ain Shams Engineering Journal, Vol. 6, pp. 133–144, 2015.
45. Mahajan R., Gupta S., Bedi R.K., "Design Of Software Fault Prediction Model Using BR Technique", Procedia Computer Science, Vol. 46, pp. 849 – 858, 2015.
46. W. S, "A literature survey of the quality economics of defect-detection techniques," in Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering (ISESE), pp. 194–203, 2006.
47. R. Huitt and N. Wilde, "Maintenance support for object-oriented programs," IEEE Transactions on Software Engineering, vol. 18, no. 12, pp. 1038–1044, 1992.
48. J. C, "Software quality in 2010: a survey of the state of the art," in Founder and Chief Scientist Emeritus, 2010.
49. http://mdp.ivv.nasa.gov/.
50. http://promisedata.googlecode.com/svn/trunk/defect/
51. Shan C., Chen B., Hu C., Xue J., Li N., "SOFTWARE DEFECT PREDICTION MODEL BASED ON LLE AND SVM" Communications Security Conference; pp 1-5, 22-24 May 2014.
52. Xia Y., Yan G., Jiang X., Yang Y., "A new metrics selection method for software defect prediction," Progress in Informatics and Computing (PIC), International Conference, pp.433-436, 16-18 May 2014.
53. Malhotra, R., Pritam, N., Singh, Y., "On the applicability of evolutionary computation for software defect prediction," Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference, pp.2249-2257, 24-27 Sept. 2014.
54. Chug, A., Dhall, S., "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm, "Confluence 2013: The Next Generation Information Technology Summit, pp.173-179, 26-27 Sept. 2013.
55. Verma, R., Gupta, A., "Software defect prediction using two level data pre-processing," Recent Advances in Computing and Software Systems (RACSS), International Conference, pp.311-317, 25-27 April 2012.
56. Singh M., Salaria D.S., "Software Defect Prediction Tool based on Neural Network", International Journal of Computer Applications, Volume 70– No.22, pp-0975 – 8887, May 2013.
57. Shrivastava A., Shrivastava V., "A Hybrid Model of Soft Computing Technique for Software Fault Prediction", International Journal of Current Engineering and Tech. Vol. 4, No. 4, Aug 2014.
58. Askari, M.M, Bardsiri, V.K., "Software Defect Prediction using a High Performance Neural Network", International Journal of Software Engineering and Its Applications, Vol. 8, No. 12, pp. 177-188, 2014.
59. Racharla Suresh Kumar, Bachala Satyanarayana, "Adaptive Genetic Algorithm Based Artificial Neural Network for Software Defect Prediction", Global Journal of Computer Science and Technology : D, Vol. 15, Issue No. 1,Version 1.0, pp. 23-32, 2015.