Early Software Reliability Prediction Using ANN for Process Oriented Development at Prototype Level

K. Krishna Mohan[1], A. K. Verma[2], A. Srividya[3]
[1]Reliability Engineering Group, [2]Department of Electrical Engineering, [3] Department of Civil Engineering
Indian Institute of Technology Bombay, Mumbai – 400076, INDIA
kkm@ee.iitb.ac.in, akv@ee.iitb.ac.in, asvidya@civil.iitb.ac.in

***Abstract*-It is important to take into account the proven processes like Rational Unified Process (RUP) to mitigate risks and increase the reliability of systems while building distributed based applications. This paper presents an algorithm using feed-forward neural network for early qualitative software reliability prediction. The inputs for neural networks consist of techno-complexity, practitioner's level, creation effort, size and leakage defects. The number of defects detected in each cycle can be predicted by using Artificial Neural Network (ANN).**

Key words: Software Reliability prediction, Artificial Neural Network, Process oriented development, Leakage defects.

I. INTRODUCTION

This paper provides an approach for early qualitative and software reliability prediction [1] using Artificial Neural Network (ANN) and the adoption of RUP in building the applications, focusing on various areas of software development. Reliability prediction has to be done early in the Software Development Life Cycle (SDLC) at the prototype level before the actual development process. In general, any predictions about real time implementation are solely based on the prototype studies. A unique take on strengthening the role of the prototype itself, without actually realizing it, would be to arrive at predictions using historical information from similar PoCs or the permeating experience of those involved in projects of comparable nature. Using ANN should make this crucial bypassing feasible, which has been illustrated in this paper.

II. SOFTWARE RELIABILITY PREDICTION USING ANN

This section explains the use of ANN for predicting the defects before taking up a project implementation well before the beginning the project. The output variable refers to the expected number of defects in the software before the beginning of the project and the software development stages considered are requirements, design, coding, unit testing and IST testing. The application of ANN has been gaining importance in the area of software reliability [2]. The neural networks have the advantages of adaptive learning, non-linear generalization, fault tolerance, resistance to noisy data, and parallel computation abilities. ANN models contain three steps: data normalization, network training, and fault prediction. In this paper, techno-complexity (technology + complexity), practitioner's level (experience + product familiarity), creation effort, size and leakage defects are used as input features. The data sets obtained from a high maturity software development organization are selected for collecting the data in the form of a defect consolidation log table.

III. DEFECT PREDICTION

A neural network can recognize various conditions or states of a complex system once it has been trained. Feed-forward neural network is set up and back propagation algorithms are used for the training. Input layer in the network is set up with one node for each input feature. Hence, architecture with single hidden layer is selected. The number of neurons in the hidden layer is decided using a trial and error method. Starting from two, the number of neurons is increased by one in each trial until the required accuracy is achieved with quick convergence. The neuron in the output layer represents the predicted number of defects. To illustrate the proposed model, the defects are predicted for an electronic fund transfer module. Based on the data collected from the defect consolidation log for a typical module, of the fifteen modules used in the study for each phase, 9 modules are used for training, and 6 modules are used for testing. The input features are normalized in the range of 0.0 to 1.0 for all the features. The network is trained using different back propagation algorithms and it is observed that Levenberg–Marquardt (LM) [3] algorithm converges quickly. Hence, this algorithm is used for the training. Log sigmoid and tan sigmoid activation functions are used for neurons in the hidden layer and the output layer, respectively. Using trial and error method, the number of neurons in the hidden layer is obtained as 9. The weights and biases of the network are initially selected randomly. The stopping criteria used for training process is the achievement of one of the conditions namely mean square error of 10^{-2}, gradient of 10^{-10} or 100 epochs. In the RUP modules considered in this paper, the execution of each module is done iteratively in three cycles. Each cycle consists of five stages namely requirements, design, coding, unit testing and IST testing. The actual number of defects observed in each phase is shown in Table 1. The effectiveness of the proposed neutral network is evaluated in terms of Mean Magnitude of Relative Error (MMRE). The MMRE is defined as:

MMRE = |(predicted defects-actual defects)/actual defects |.

Discussions on the acceptable limits of MMRE for qualitative software defect predictions are given in [4]. It can be observed that MMRE of the predictions shown in Table 1, using the proposed ANN model with five input features are found to be in acceptable limits. It was observed that the network with leakage defects as an input feature gives better predictions. The actual and predicted defects for all the modules in various phases in cycle 1 are shown in Fig. 1. The difference between the actual number of defects and the neutral network predictions are marginal in most of the cases.

IV. CONCLUSIONS

Usage of Neutral Network acts as a conclusive weighing factor in deciding upon the taking up of a project implementation. In effect, a feasibility study has been conducted in this paper, which acts as a significant pointer towards the capture of qualitative reliability (in the form of expected number of defects) well before the beginning of the project. The proposed model has been applied to real time project it is and observed that the MMRE values are within the acceptable limits.

Phases of RUP	Average number of defects	MMRE of ANN Predictions
Cycle 1		
Requirements	3	0.23
Design	5	0.31
Coding	33	0.07
Unit testing	117	0.16
IST Testing	20	0.28
Cycle 2		
Requirements	2	0.14
Design	5	0.47
Coding	19	0.05
Unit testing	34	0.17
IST Testing	10	0.14
Cycle 3		
Requirements	2	0.10
Design	2	0.22
Coding	2	0.04
Unit testing	4	0.59
IST Testing	2	0.13

Table1 Actual number of defects and errors in ANN predictions

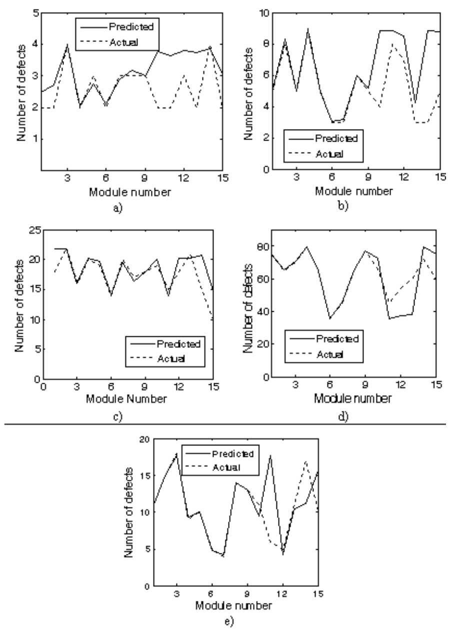

Fig.1. Actual and predicted number of defects for various phases in cycle 1
a) Requirements b) Design c) Coding d)Unit testing e) IST testing

REFERENCES

[1] Smidts, C., Stutzke, M. & Stoddard, R.W. 1998, "Software reliability modeling: An approach to early reliability prediction", IEEE Transactions on Reliability, 47(3),(1998), pp. 268-278.

[2] Cai, K.Y., Cai, L., Wang, W.D., Yu, Z.Y. & Zhang, D, 'On the neural network approach in software reliability modeling,' Journal of Systems and Software, 58(1), (2001), pp. 47-62.

[3] C. Bishop, Neural networks for pattern recognition, Oxford University Press, New York, (1995).

[4] A.K.Verma, R.Anil, Om Prakash Jain, Fuzzy logic Based Maturity Rating for Software Performance Prediction", International Journal of Automation and Computing", 4(4), 2007, pp.406-412.