J. Software Engineering & Applications, 2010, 3: 331-340
doi:10.4236/jsea.2010.34038 Published Online April 2010 (http://www.SciRP.org/journal/jsea)

Test Effort Estimation Using Neural Network

Chintala Abhishek*, Veginati Pavan Kumar, Harish Vitta, Praveen Ranjan Srivastava

Department of Computer Science and Information System, Birla Institute of Technology and Science, Pilani, India.
Email: {*chabhishek123, pavanon9, harishvitta, praveenrsrivastava}@gmail.com

Received January 5th, 2010; revised February 21st, 2010; accepted February 25th, 2010.

ABSTRACT

In software industry the major problem encountered during project scheduling is in deciding what proportion of the resources has allocated to the testing phase. In general it has been observed that about 40%-50% *of the resources need to be allocated to the testing phase. However it is very difficult to predict the exact amount of effort required to be allocated to testing phase. As a result the project planning goes haywire. The project which has not been tested sufficiently can cause huge losses to the organization. This research paper focuses on finding a method which gives a measure of the effort to be spent on the testing phase. This paper provides effort estimates during pre-coding and post-coding phases using neural network to predict more accurately.*

Keywords*: *Test Effort Estimation, Neural Network, Use Case Points, Halstead Model

1. Introduction

Software engineering [1] is a field that provides standardized approaches for the development, operation, and maintenance of software. Software Engineering as a discipline the need arose when there was software crisis [1]. The need for producing software of high quality and to have a control on the effort both in terms of the money and the person-hours gave software Engineering a higher prominence. It defines a process which helps for project management [1].

A crucial aspect of software Engineering is software testing [1]. Software testing is a phase of software development which deals with testing the developed product or project. A project /product which have been developed without sufficient testing might contain major bugs which can render the entire project useless and also cause losses of critical data.

Software testing [1] by definition is the process of validating and verifying a software product or a project or an application. It should be tested on the aspects of: meeting the requirements of the user, functionality, and characteristics of the developed software.

Generally software test life cycle involves several stages and it can be classified into three major phases.

Initial phase: This phase involves with identifying which aspects of the designed are to be tested followed by the creation of a test phase strategy.

Intermediate phase: This phase involves the development step in which the procedures and the scenarios all are defined. This is followed by the execution step which deals with implementing the developed plan and reporting any error found.

Termination phase: This is longest phase involves several activities, once the testing is finished a report indicating the fitness of the project/product to be released is created. Then the analysis is carried out with the client to deal with the problems faced during its real time implementation. Then the detection of any further existing defects is carried out. If there are any modifications done then the entire component is retested to determine any side effects that could have occurred because of changes in previous step (Regression testing). If the system meets the exit criteria the testing phase is terminated.

In the ideal scenario it is desirable to have exhaustive testing as this ensures that there are no bugs or errors. This is not possible even with a project of very less complexity. Thus the need for having an efficient testing strategy arises. Software testing phase needs to be planned to be carried out efficiently.

Artificial neural network [2-4] is a soft computing technique that tries to achieve the functionality of biological neural network. It consists of group of artificial neurons that work on mathematical model to process the information and to solve highly complex problems. It involves a network of simple processing elements called as neurons that are connected. The connections between the neurons help in realizing a complex functionality. As mentioned above Software

testing is a challenging field and this paper proposes and efficient methodology to estimate test effort estimation with more accuracy using artificial neural network.

The paper is written with the general introduction of the Software testing in Introduction, followed by the description about the background work (Section 2). Section 3 deals with actual problem while section four is fully devoted on proposed approach of the paper. Section 4 deals with the application of the proposed model and finally in Section 6, the results obtained are discussed.

2. Background Work

Estimation accuracy can be achieved by choosing an accurate model for measuring. This section provides with the information that has been gathered, on which the work is based upon.

2.1 Use Case Point [5]

The effort to be estimated for the pre-coding phase is based on the use case point analysis [5]. Nageswaran [5] proposes a strategy which calculates effort based on the unadjusted use case weight (UUCW), unadjusted actor weight (UAW) and the technical and environmental factors (TEF). Those factors are calculated based on the classification of actors and usecases into simple, average, complex and very complex classes. The obtained unadjusted use case point (AUCP) is multiplied by a factor to obtain the effort. The effort obtained in this is not accurate with the expected level of accuracy in estimation. Our proposed model is totally inspired by Nageswarn work. This paper provides an improvement over the method proposed by Nageswaran [5]. Nageswaran [5] model can be stated as:

During this phase the project manager has the design document based on which he can make an estimate of the effort that needs to be allocated to the testing phase.

The proposed method suggest the usage of adjusted unadjusted usecase weight, unadjusted actor weight (UCW), technical and environmental factor (TEF) as a measure for the test effort estimation. Back propagation in neural network is used for training the network.

The inputs are taken for a particular project based on the design document. The UUCW is calculated as

Usecase component:

UUCW = (No. of usecases of type simple*1 + No. of usecases type average*2 + No. of usecases of type complex*3 + No. of usecases of type very complex*4)

The usecase information **Table 1** is used for distinguishing and assigning the values.

Actor components:

The actor information is obtained from the **Table 2**.

TEF components:

The technical and environmental factors are assigned as indicated by the **Table 3**.

Table 1. Usecase weight assignment table [5]

Usecase type	Description	Weight
Simple	<=3	1
Average	4-7	2
Complex	>7	3

Table 2. Actor weight assignment table [5]

Actor type	Description	Weight
Simple	GUI	1
Average	Interactive	2
Complex	Low interaction	3

Table 3. TEF weight assignment factors [5]

Factor	Description	Assigned value
F1	Test tools	5
F2	Documented inputs	5
F3	Development environment	2
F4	Test environment	3
F5	Test ware reuse	3
F6	Distributed system	4
F7	Performance objectives	2
F8	Security	4
F9	Complex interface	5

UAW and TEF are calculated as:

UAW = $\sum$ Actor weight*number of actors

TEF = $\sum$ Assigned Weight*assigned value

2.2 Halstead Model [6]

The background study involved studying Halstead model [6]. A brief explanation of it is given here. It makes use of some primitive measures to determine the length and the volume of the program [6]. It makes use of the factors such as total number of operators (n_1), total number of operands (n_2), total number of their operator occurrences (N_1) and the total number of operand occurrences (N_2). He also proposes a formula for measuring the development effort and development time using such measures.

The length N is estimated according to Halstead as:

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

The program volume is given by the formula

$$V = N*\log_2(n_1+n_2)$$

A volume ratio is defined by him, represented by L, its value should not be more than one. It is represented by the formula

$$L = 2/n_1*n_2/N_2$$

The effort is given by the formula

Effort = $((n_1 * N_2)/ (\text{float } (2 * n_2)) * N *\log(n, 2)$

This is the effort as estimated by the Halstead model

and is obtained in elementary mental discriminations.

2.3 Cognitive Complexity [7]

Kushwaha [7] suggests that effort can estimated based on the total weighted information count of line of code and software and basic control structures. This method involves a complex estimation function. The effectiveness of which has not been established for large projects.

2.4 Effort Estimation Using Soft Computing Techniques [8]

Sandhu [8] shows that soft computing technique–neuronfuzzy can be applied for effort estimation by establishing its accuracy by comparing it with various other models. The estimation was done on NASA project data.

Neurofuzzy was able to estimate the nonlinear function with more accuracy. This paper helps in suggesting that effort estimation based on soft computing is indeed a right direction of accurate estimation.

Introduction to neural network:

2.5 Neural Network [2-4]

The neural network structure is used for solving complex problems [2-4]. The Backpropagation methodology is used for training the neural network. A set of input training data and the expected output is created and the network is trained with the training set. The network is trained over multiple iterations. Over the multiple iterations the network tries to converge towards the expected output and thus training itself with the required training function. The trained network is provided with the inputs from a test set and it gives the output which is the estimated output.

2.6 Neural Network Structure [9]

The neural network structure is realized using the freeware Neuroph neural network framework [9]. EasyNeurons [9] is the GUI application for it. It is a java library. The multilayer perceptron model is used for creating a neural network, as this would be the appropriate network structure which would help in realizing the problem. In a multilayer network there will be one input layer, atleast one hidden layer and one output layer. Backpropagation is used as the training methodology *i.e.*, the learning rule. It is a supervised learning algorithm. It is a learning methodology through which the network trains itself through multiple iterations over the test data. It does so by reducing an error function. The network eventually converges towards accurate values as it is trained with more and more training data.

The activation function used here is the Tanh function. The activation function is an abstraction of the action potential. It represents whether the cell should fire or not. The Tanh function is normalized .It is real valued differential curve, as represented in the **Figure 1**.

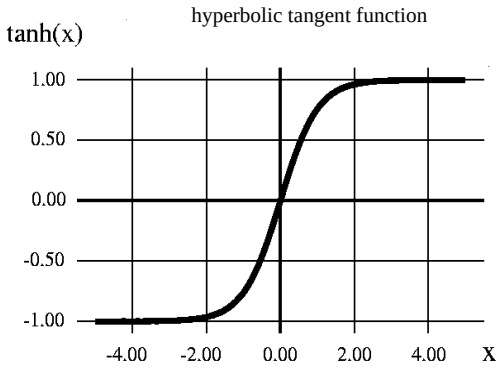

Figure 1. Tanh function

A brief description of the conventional pre and post coding effort estimation models is given here:

2.7 Conventional Methods for Pre Coding Effort Estimation [9]

1) The testing phase effort is not generally calculated. Once the product is designed the rest of the resources in terms of the budget and time are allocated to the testing phase. This methodology can be applied for mission critical system testing, as any compromise in the quality of the product would lead to huge losses [1].

2) Another method which is used for planning the testing phase effort is the percentage of the total development effort to be spent on testing. This also doesn't provide with efficient planning of resources.

2.8 Conventional Methods for Post Coding Effort Estimation [9]

1) Based on Software size:

The software size is available from the code and a productivity figure is applied to it. It involves the multiplication of number of function points and effort per function point. This approach is too simplistic, it involves estimations based on other project data which can lead to errors and it includes rigorous data maintenance [1].

2) Delphi Technique:

This technique involves a group of experts answering a questionnaire and arriving at a converging solution to the problem. The technique is time and resource consuming and generally doesn't lead to accurate predictions [1].

3) Test case enumeration based estimation:

It involves the enumeration of the entire test cases and the effort for each test case is estimated and beta distribution is applied over it. It is time consuming process.

3. Actual Problem

The test effort estimation is a big challenge in project

planning. There are no models presently available that can estimate the test effort accurately. The effort that needs to be spent on the testing phase needs to calculate precisely. The effort needs to be estimated both before the coding phase and after the coding phase. A comparison of the observed efforts should not be large, which is an indication of effective model. The problem is to propose a model which estimates the effort accurately. The proposed model should not be dependent on the type of project.

4. Proposed Approach

4.1 Architecture

The architecture involves two components: pre and post effort estimation components and learning rule used here is Back propagation algorithm as shown in the **Figure 2**.

The pre coding effort estimation consists of the three inputs components which get inputs from the design document. The three components are:

Actor components: This takes information about the actors involved in the system.

Usecase component: This takes information about the usecase involved in the design document.

TEF component: This takes the information regarding the technical and environmental factors involved in the system.

Further description about these components is given ahead in the paper.

The post coding effort estimation takes input from the code document and it has three components. They are:

Variables component: This takes the information regarding the variables involved in the system.

Complexity component: This takes the information regarding the complexity of the system.

Criticalness component: This takes information regarding the criticalness of the system.

These are further discussed ahead in the research paper. The activation function used is tanh. 'I' represent the inputs given to the system. 'X' represents the values after the application of activation function and 'w' represents the weights assigned.

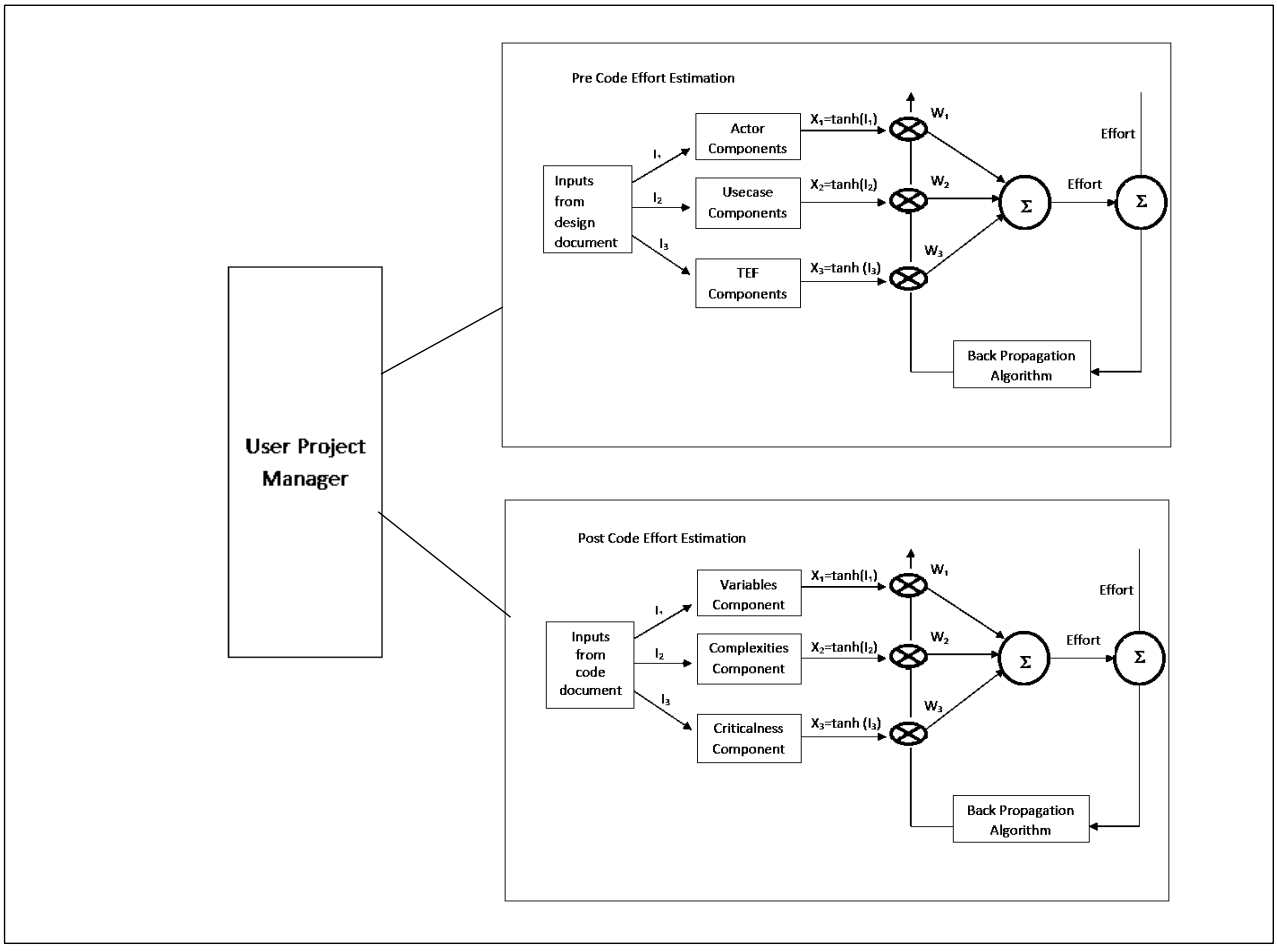

Figure 2. Architecture of the proposed system

4.2 Pre Coding Phase Effort Estimation

The proposed Pre coding effort estimation is based on the model proposed by Nageswaran [5].

Upon which this paper proposes a new improvement *i.e.*

The obtained values of UAW, UUCW, TEF and estimated test effort are trained to the network. The network trains itself to predict the values of weights and threshold values for the activation levels. The network is trained through test data over multiple iterations.

Then the network is provided with the information for the project for which an estimate needs to obtain. The information is derived from the design document. The network provides with the effort in terms of the person-months.

4.3 Neural Network Structure for Pre Coding Effort Estimation

1) Designing the network:

The network structure chosen for this phase involves three layers. One input layer through which the UAW, UUCW, TEF are given as the inputs to the network. The hidden layer consists of three nodes which are used for realizing the effort estimation function. The output layer consists of one node. The output of which gives the effort for the phase.

2) Training the network:

The network is trained with the test data that has been obtained from various sources. The test data is taken from Estimator Pal [9] and Use case Point [5] both of which contain the test data taken from a real time project, also we are using some of the real data for training purpose. This data would be helpful in training the neural network. UUCW, UAW, TEF are calculated for various projects and their test effort is provided as the training data. The network is trained for the data with maximum error rate of 0.2. The network gets trained with the provided test data over few thousands of iterations.

3) Testing the network:

The use case, actor, technical and environmental factors for the project whose test effort needs to be evaluated is taken as the input and is provided to the network which in turn provides the users with effort in person-months.

The **Figure 3** shows the neural network structure for the pre coding phase effort estimation model. It is developed in the easyneurons environment. It shows the thresholds, activation values for input, hidden, output nodes for the structure.

4.4 Post Coding Phase Effort Estimation

During this phase the project manager uses the coding document to make an estimation of the test effort.

The proposed method is based on the fact that the test effort is based on the number of inputs, number of outputs, and the complexity of the code and the criticalness of the code.

Different weightage factors are given a value each.

Variables component: As the number of inputs increases the number of test cases also increases. Different measures are given for different types of inputs. It can be observed from the **Table 4**. The method proposed makes use of the fact that a character data type doesn't need more than single test data, while an integer data would require more test cases and array variable would require even more test cases for testing [1]. Thus the assigned weights increase proportionately. var[i] takes the values of number of occurrences of each variable in the order mentioned in the **Table 4**. Var_comp[i] is the assigned weights which are taken from the **Table 4**. Thus the variable var_val is the summation of product of the number of occurrences of variables and their assigned weights.

Complexity component:

The complexity of the code is a measure of the number of test cases required for testing. Thus **Table 5** giving a measure for the complexity of the code is used. The assigned weight increases proportionately as the complexity of the code increases.

Criticalness component:

The number of test cases increases proportionately with increase in the criticalness of the system, the measure can be obtained from **Table 6**. The criticalness of the code is an indication of the importance of the code. If it is a general purpose code it is assigned a very less value (most of the project classifies under it). However if it is an essential mission critical code then the test effort increases proportionately as the number of test cases increases rapidly and thus the criticalness factor is assigned a very high value. As illustrated in the **Table 6** below.

A variable σ has been defined as an intermediate variable in measuring the effort. It is the product of var_val value, complexity value and the criticalness value.

Table 4. Complexity assignment table for variables

Input type	Assigned weight
Integer	3
Array variable	4
Character	1

Table 5. Complexity weight assignment for code

Complexity of the code	Assigned weight
O(n)	1
O(log n)	2
O(nlog n)	3
$O(n^2)$	4
$O(n^3)$	5
$O(n^4)$	6

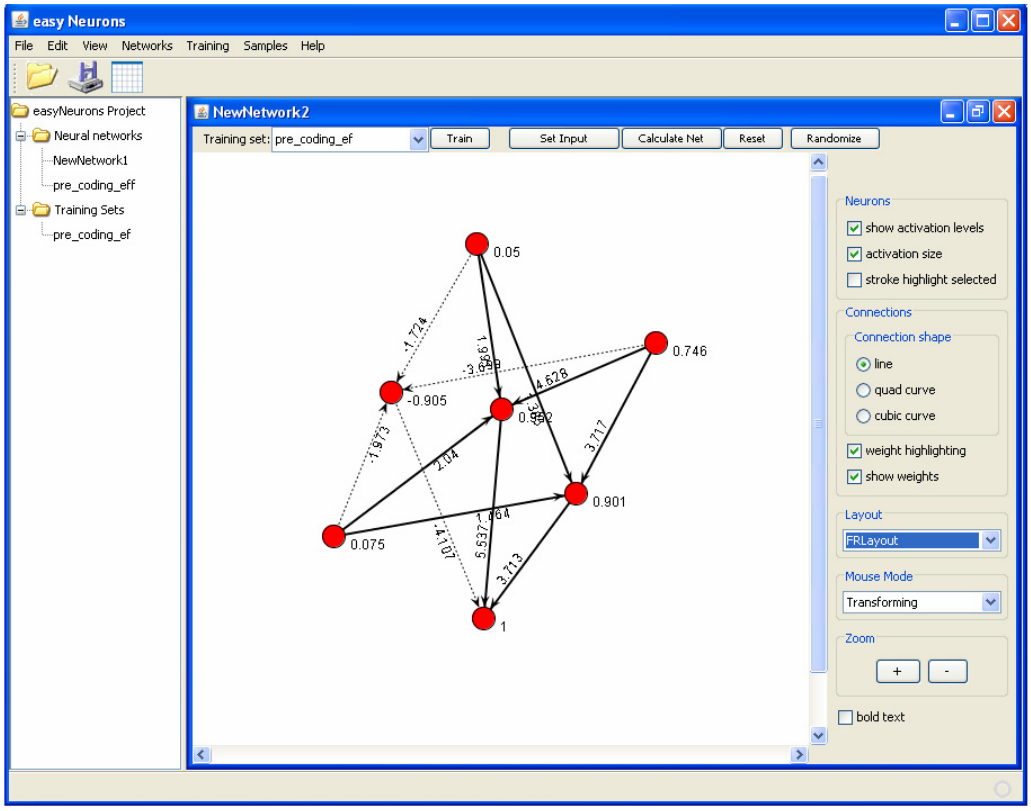

Figure 3. Neural network structure

Table 6. Criticalness assignment table

Criticalness of the code	Assigned weight
General purpose code	1
Higher critical code	2
Mission critical code	3

$$\text{Var\_val} = \sum (\text{var[i]}*\text{var\_comp[i]})$$

$$\sigma = \text{var\_val}*\text{complexity}*\text{criticalness}$$

$$\text{Effort} = (\sigma + 13.5)*10/3 \quad (1)$$

The equation is arrived based on the halstead effort estimation model. The effort is estimated on a large number of test cases (the test cases here being the source codes of quick sort, bubble sort, gcd program etc.,) the halstead effort is estimated for the test cases, the effort is obtained in elementary mental discriminations. For the same test cases the value of σ is computed and a large pool of values for the comparison of the proposed variable and the halstead estimated effort is obtained. The constant 13.5 and the multiplying factor 10/3 have been arrived from this large pool of values and their comparisons.

A relation is obtained for the obtained σ values and the estimated values. Thus Equation (1) has been derived.

The obtained values var_val and σ and estimated test effort according to the proposed model passed as training set to the network. The network trains itself to predict the values of weights and threshold values for the activation levels. The network is trained through test data over multiple iterations.

Then the network is provided with the information for the project for which an estimate needs to be obtained. The information is derived from the source code document. The various parameters are estimated from the source code like the variable occurrences, complexity of the code etc. The network provides with the effort in terms of the elementary mental discriminations (as the formula was derived using the Halstead model). The network gets trained with the proposed effort estimation function for the post coding phase.

4.5 Neural Network Structure for Post Coding Effort Estimation

1) Designing the network:

The designing of the network involves the selection of the network architecture. The architecture is chosen in such a way that it is in accordance with the proposed effort estimation function. The proposed effort estimation function for the post coding phase implies the design of the network structure with two input nodes, two hidden nodes and one output node. The two input nodes are provided with the values of var_val and σ at input

layer. The network gives effort in terms of the EMDs on the output layer which consists of only one node, the output node.

2) Training the network:

Training the network involves the compilation of the test data: the test data has been obtained by manually calculating the proposed model effort, var_val, σ, halstead effort for a substantial number of program codes. The training set is provided to the network designed as above. The acceptable error rate is set to 0.1. The network is trained with the compiled test data and the network converges over a period of thousands of iterations.

3) Testing the network:

The proposed model accepts the number of variables and their occurrences, complexity of the code, criticalness of the code as the input and it computes the values of var_val, σ and provides it to the network. The network calculates the estimated effort according to the proposed evolved model and produces an output in terms of elementary mental discriminations (EMD).

Figure 4 shows the neural network structure which has been obtained using the easyneurons freeware application. The figure shows the network structure, the thresholds, and the activation levels on various nodes.

The model developed takes the inputs from the users (project managers) estimates the intermediate values, passes it to neural network structure which was realized and retrieves the information from it and passes it to the model which then evolves the data to provide with the estimated effort as the final output.

5. Application of Proposed Model to Test Cases

The proposed model which has the effort estimation in pre coding phase in person-months and in post coding phase in elementary mental discriminations has been applied to various project data. The data has been obtained from Estimator Pal, Usecase point [14] which has a detailed design report. It has also been applied to other minor projects.

The post estimation model is very cumbersome. It has been applied to obtain the proposed estimated value as well as the value that is obtained from Halstead model.

6. Results and Discussion

The model has been applied to various projects as mentioned above. The following are the results obtained.

Figure 5 gives the comparison of pre code effort esti-

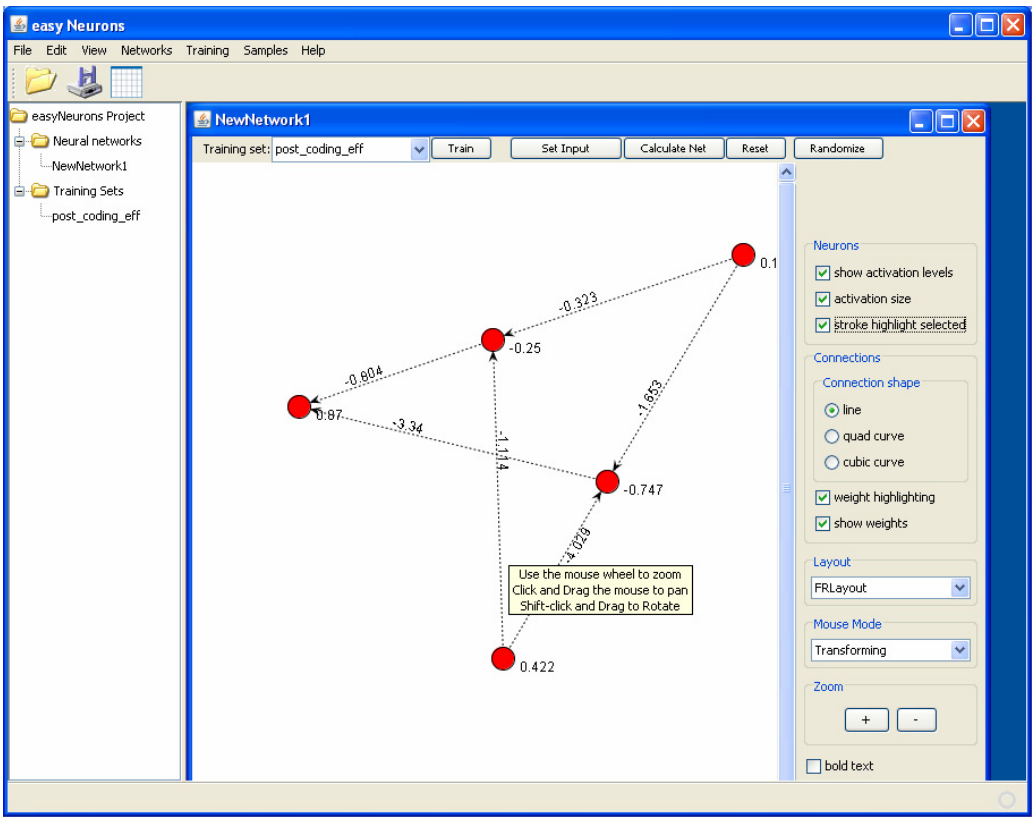

Figure 4. Neural network structure

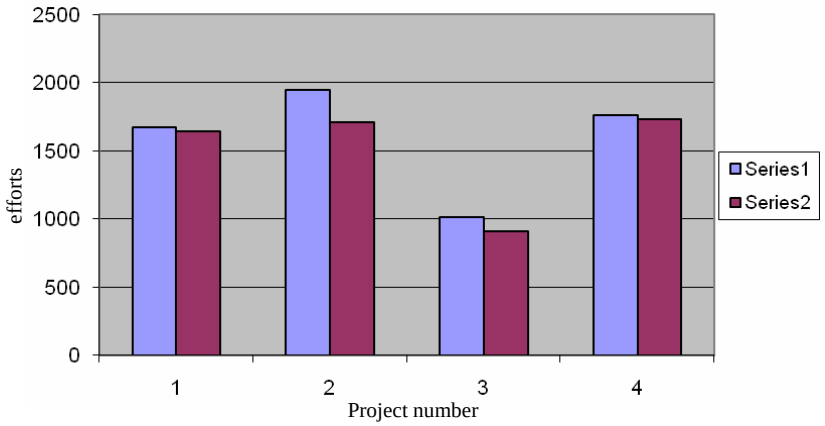

Figure 5. Comparison of pre code effort estimation

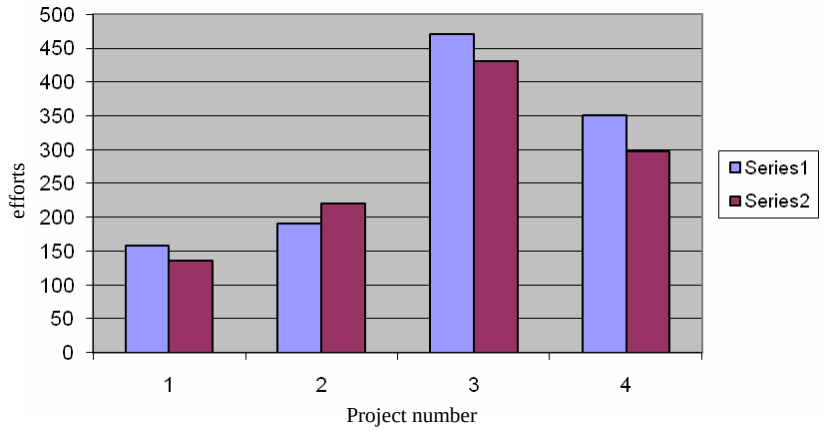

Figure 6. Comparison of post code effort estimations

mations. The X-axis represents the number of the test case and the Y-axis represents the effort interms of person-months. Series1 represents the test effort for pre-coding effort estimation based on the proposed model, while the Series2 represents the pre code effort estimation based on a traditional method. The method to which the proposed method is being compared to is [5] effort estimation based on usecase.

Careful analysis of the results obtained provides the information that the proposed estimation has a deviation of about 8% over the traditional method that has been chosen. This deviation is not much considering the fact that the effort estimated by the traditional method has also not being found to be accurate when applied to real time projects. The method based on usecase points [5] and several other traditional methods haven't produced an accurate estimate of the test effort. The proposed method has been applied on real time data from few of the projects that have been specified above and it has been found to produce an estimate of about 8% deviation from the mentioned effort.

The interpretation of the results obtained and mentioned in the above graph indicate another fact, that the estimated effort has been found to be always on the higher side of traditional method. The deviation found here is found to be on the positive side.

When the results were analyzed with the real time data the proposed model has been found to be more accurate than the traditional method that has been chosen. The proposed model estimated the effort more accurately.

Figure 6 shows the comparison of effort estimation for post coding phase. The X-axis represents the number of the project and the Y-axis gives the effort estimation in terms of elementary mental discriminations. Series1 represents the test effort estimation based on the proposed model which was evolved from the halstead model, cyclomatic model [7] and the application of neural network. Series represents the test effort estimation based on the Halstead model [6].

It can be observed from the graph and the analysis of the results which were obtained by applying the proposed model over several projects that there is about 10% deviation in the test effort estimation for halstead model. The model has been applied to various projects mentioned as above for post effort estimation.

The deviation has also been found to be varying and it has been seen that it is both on the positive side and negative side of the halstead effort. It can be observed

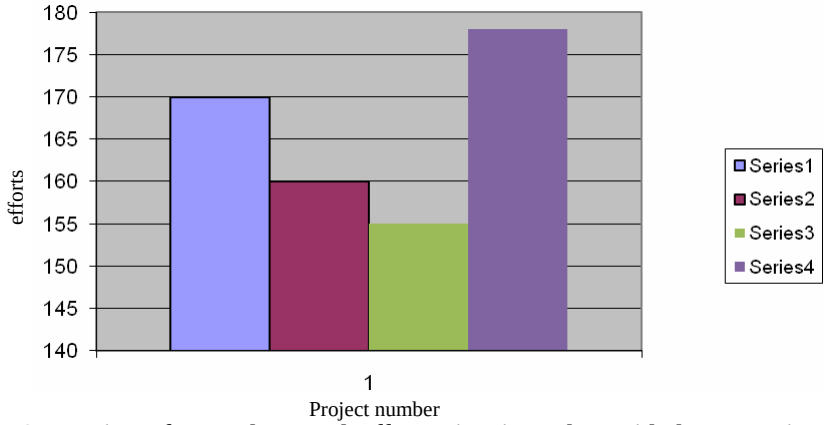

Figure 7. Comparison of pre and post code effort estimations, along with the conventional estimates

that the cyclomatic complexity model [7] and the halstead model [6] haven't been able to estimate the effort accurately. In general there has not been any model that could estimate the effort estimation accurately.

There is no accurate effort estimation for post coding phase. The proposed model has produced results which are in synchronization with the actual effort estimations and found to be more accurate.

In **Figure 7** the comparison of pre and post code effort estimations is given. The model developed has been applied to some student projects and the graph is plotted. In the figure the X axis represents the project number and Y-axis represents the effort. Series1 indicates the pre coding test effort for the proposed model and Series2 represents traditional method pre coding effort estimation, Series3 represents the post coding test effort for the proposed model and Series4 represents the traditional method post coding test effort estimation. It is showing a variation of about 8% over large number of projects. Thus it confirms the fact the estimated efforts both in pre and post coding phase have higher accuracy than the conventional models which as shown earlier show large deviation.

7. Conclusions

The models used for the traditional pre coding effort estimations use the usecase point or the function point.

The paper has covered brief details of the various traditional methods for effort estimations both in pre coding phase and in post coding phase. It then had the introduction of various keywords which are a part of the proposed model.

The proposed effort estimation models for pre coding phase based on usecase point and soft computing technique- neural network has been applied to improve upon the accuracy. The method that has been followed and the metric proposed have an advantage that it produces accurate results. For the post coding effort estimation the proposed model estimated the effort based on and used neural network to improve upon accuracy and the results have been found to show that the proposed estimation is in synchronization with the traditional effort estimation models.

The future scope for the proposed model is based in the direction that the model developed needs to be applied to large number of test cases *i.e.*, real time projects as the proposed model has a unique feature of learning through usage. The model converges towards more accurate values as it used over time. The model developed can be evolved even further in the view that more number of parameters which have a minor effect on the effort estimation be also considered for effort estimation and the model can be evolved.

REFERENCES

[1] R. S. Pressman, "Software Engineering – A Practitioner's Approach," 5th Edition, McGraw Hill, New York, 2002.

[2] B. T. Rao and B. Sameet, "A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network," *International Journal of Computer Science and Network Security*, Vol. 9, No. 6, June 2009, pp. 126-131.

[3] H. Zeng and D. Rine, "Estimation of Software Defects Fix Effort Using Neural Network," *IEEE* 28*th Annual International Computer Software and Applications Conference* (*COMPSAC*'04), Los Alamitos, 28-30 September 2004, Vol. 2, pp. 20-21.

[4] K. K. Agarwal, P. Chandra, *et al.*, "Evaluation of Various Training Algorithms in a Neural Network Model for Software Engineering Applications," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 4, July 2005, pp. 1-4.

[5] S. Nageswaran, "Test Effort Estimation Using Use Case Points (UCP)," 14*th International Software/Internet Quality Week*, San Francisco, 29 May-1 June 2001.

[6] T. E. Hastings and A. S. M. Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation," *IEEE Transactions on Software Engineering*,

Vol. 27, No. 4, April 2001, pp. 337-350.

[7] D. S. Kushwaha and A. K. Misra, "Software Test Effort Estimation," *ACM SIGSOFT Software Engineering Notes*, Vol. 33, No. 3, May 2008.

[8] P. S. Sandhu, P. Bassi and A. S. Brar, "Software Effort Estimation Using Soft Computing Techniques," World Academy of Science, Engineering and Technology, 2008, pp. 488-491.

[9] M. Chemuturi, "Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators," J. Ross Publishing, Lauderdale, July 2009.

[10] Free Software Foundation, "Neuroph Framework," Version 3, June 2007.

[11] M. Braz and S Vergilio, "Software Effort Estimation Based on Use Cases," 30*th Annual International Computer Software and Applications Conference* (*COMPSAC*'06), Chicago, 17-21 September 2006, Vol. 1, pp. 221-228.

[12] G. Banerjee, "Use Case Points – An Estimation Approach," Unpublished, August 2001.

[13] J. Kaur, S. Singh and K. S. Kahlon, "Comparative Analysis of the Software Effort Estimation Models," World Academy of Science, Engineering and Technology, Vol. 46, 2008, pp. 485-487.

[14] N. Nagappan, "Toward a Software Testing and Reliability Early Warning Metric Suite," 26*th International Conference on Software Engineering* (*ICSE*'04), Shanghai, 2004, pp. 60-62.

[15] C. Huang, J. Lo, S. Kuo, *et al*., "Software Reliability Modeling and Cost Estimation Incorporating Test-Effort and Efficiency," 10*th International Symposium on Software Reliability Engineering*, Boca Raton, 1-4 November 1999, pp. 62-72.

[16] O. Mizuno, E. Shigematsu, Y. Takagi, *et al*., "On Estimating Testing Effort Needed to Assure Field Quality in Software Development," 13*th International Symposium on Software Reliability Engineering* (*ISSRE*'02), Annapolis, 12-15 November 2002, p. 139.

