Application of Artificial Neural Network for Procedure and Object Oriented Software Effort Estimation

Jagannath Singh, and Bibhudatta Sahoo

Abstract—Software effort estimation guides the bedding, planning, development and maintenance process of software product. Software development uses different paradigm like: procedure oriented, object oriented, Agile, Incremental, component based and web based etc. Different companies use different techniques for their software project development. The available estimation techniques are not suitable for all types of software development techniques. So there is a need of estimation technique that can be applied on all type of software. This paper we are evaluating the application of artificial neural networks in prediction of effort in conventional and Object Oriented Software development approach. We have used feed-forward neural network created using MATLAB10 (NN tool kit) and applied on two different types of datasets, one for conventional software and another for object oriented software. The simulation results were studied and we found that artificial neural network model works very accurately on both types of software development techniques.

***Keywords*—**Effort Estimation, Artificial Neural Network, NNtool, MMRE, Class Points, Types of software.

I. Introduction

SOFTWARE project managers require reliable methods for estimating software project costs, and it is especially important at the early stage of software cycle. Because these estimates are needed for budding and budgeting. Software development involves a number of interrelated factors which affect development effort and productivity. Accurate forecasting has proved difficult since many of these relationships are not well understood. Improving the estimation techniques those are available to project managers would facilitate more effective control of time and budgets in software development.

Software development technique keeps on changing. It was started with conventional procedure oriented design, then comes object oriented design, followed by component based, web based, incremental and now-a-days agile technique is very popular in the software development companies. There are number of competing software cost estimation methods available for software developers to predict effort required for software development. Some techniques, including FPA, COCOMO model and original regression model, are not effective, because they are not suitable for all types of software. The estimation models were designed keeping in view only one type of software development methodology. So the estimation model giving excellent results may not be suitable for other types of software. For example Function Point Analysis (FPA), which was designed for conventional software and gives good results for those, but it cannot be applied to the object oriented software. Similarly Use Case Points method was designed for object oriented software, may not work effectively for other types of software.

Due to the problem addressed above the software companies are interested into those estimation techniques which are suitable for all types of software and give more accurate results. G.E. Wittig [5] had used artificial neural network for estimation of the effort for conventional software and found that it is giving good results. S.Kanmani, et al. [17] applied artificial neural network for predicting the effort based on class points for object oriented software. In this paper we will test the applicability of ANN based estimation method on two types of software, conventional and object oriented software and we will check how it perform on two different kind of datasets. So that we can say that the ANN based model may be applied to any kind of software.

In this paper we will first describe the meaning of effort estimation and the different types of estimation techniques. Then we will give reason for why to use artificial neural networks for effort estimation. Next section will cover detail study of artificial neural networks and their types. Our main motive is to check whether artificial neural networks based prediction method is applicable on different type of software and how it performs on different software datasets. So we will give brief definition of different types of software and comparisons of their characteristics. Section 5 will cover the review of related works those motivated us for our purposed

Manuscript received October 9, 2001. (Write the date on which you submitted your paper for review.) This work was supported in part by the U.S. Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here). Paper titles should be written in uppercase and lowercase letters, not all uppercase. Avoid writing long formulas with subscripts in the title; short formulas that identify the elements are fine (e.g., "Nd–Fe–B"). Do not write "(Invited)" in the title. Full names of authors are preferred in the author field, but are not required. Put a space between authors' initials.

Jagannath Singh is with the National Institute of Technology, Rourkela , Odisha, PIN-769008(Phone: 011-661-2462358; fax: 011-661-2462351; e-mail: jagannath.singh@gmail.com).

Bibhudatta Sahoo is with National Institute of Technology, Rourkela, Odisha, PIN-769008, INDIA. (e-mail: bdsahu@nitrkl.ac.in).

work. In the last section we will give our simulation and results.

II. Software Effort Estimation

Software estimates are the basis for project bidding, budgeting and planning. These are critical practices in the software industry, because poor budgeting and planning often has dramatic consequences. When budgets and plans are too pessimistic, business opportunities can be lost, while over-optimism may be followed by significant losses [1].

Software effort estimation is the process of predicting the most realistic use of effort required to develop or maintain software. Effort estimates are used to calculate effort in work-months (WM) for the Software Development work elements of the Work Breakdown Structure (WBS).

Software estimation can be modeled as the three stages, 1st stage involves *size estimation*, 2nd stage includes *effort estimation*, and *time estimation*, followed by the 3rd stage as *cost estimation*, and *staffing estimation*. Figure 1 shows the interaction between these modules in a typical software estimation process in Software Development Life Cycle [10].

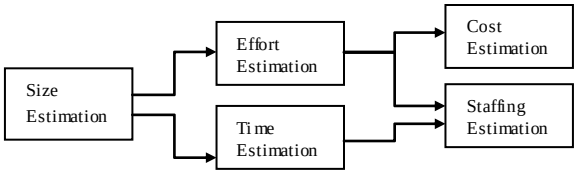

Fig. 1 Sequence of estimates in Software Development Life Cycle

According to the last research reported by the Brazilian Ministry of Science and Technology-MCT, in 2001, only 29% of the companies accomplished size estimation and 45.7% accomplished software effort estimate [2], so effort estimation has motivated considerable research in recent years.

Effort Estimation Techniques

Due course of time there are so many techniques evolved for effort estimation. Basically those can be categorized into three types-

1. Judgment Based Estimation
2. Algorithm Based Estimation
3. Analogy Based Estimation

1. Judgment Based Estimation- This is the most traditional and most popular estimation technique. In this all the estimations are made by human beings and dependent upon personal experience. It is of two types-

Expert Judgment: The mostly widely used cost estimation technique. It is an inherently top-down estimation technique. It relies on the experience, background, and business sense of one or many key people in the organization. This technique is risky because someone may overlook at some factors that make the new projects significantly different. Also the experts making estimate may not have the experience in similar projects.

Delphi Method: The technique is designed as a group communication process which aims to achieve a convergence of opinion on a specific real-world issue. Here a group is made out of most experience people in the organization. It overcomes the disadvantages of Expert Judgment.

2. Algorithm Based Estimation- As the new technologies evolves and the software size became huge, the Judgment based estimation fails to predict correctly. So need of some formula based estimation techniques comes into picture. It is also of two types.

COCOMO: The Constructive Cost Model (COCOMO) is an algorithmic estimation model developed by Barry W. Boehm. The most fundamental calculation in COCOMO is the use of Effort equation to estimate the number of Person-Months required to develop a project.

$$Effort = A*(size)^{B}$$

Where A is proportionality constant and B represents economy. Values of A and B depends upon the type of projects. A project can be classified into three types-Organic projects those involve small teams with good experience, Semi-detached projects those having medium teams with mixed experience working and embedded projects which are developed within a set of tight constraints.

TABLE I
VALUES OF 'A' AND 'B' IN COCOMO

Software Project	A	B
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Function Point Analysis (FPA): Function Point Analysis was developed by Alan Albercht of IBM in 1979. Function point metrics provide a standardized method for measuring the various functions of a software application. Depending upon this we can estimate our effort. Here all the functions are categorized in five types. Those are, Internal Logical File (ILF), External Interface File (EIF), External Input (EI), External Output (EO), and External Inquiry (EQ). For each category values assigned are low, medium or high. Besides the above mentioned domain values, fourteen complexity factors like Bach up and recovery, Data Communication etc. are given certain values as per software requirement and final estimate is calculated. Function Points are simple to understand, easy to count, require little effort and practice. [18]

3. Analogy Based Estimation- Analogy-based estimation has recently emerged as a promising approach, with comparable accuracy to algorithmic methods in some studies, and it is potentially easier to understand and apply. An estimate of the effort to complete a new software project is made by analogy with one or more previously completed projects. Ease of use may be an important factor in the successful adoption of

estimation methods within industry, so analogy-based estimation deserves further scrutiny. There are many techniques those comes under analogy based estimation-

Case Base Reasoning: CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems.

Artificial Neural Networks: Artificial Neural Networks (ANNs) is used in effort estimation due to its ability to learn from previous data. It is also able to model complex relationships between the dependent (effort) and independent variables (cost drivers). In addition, it has the ability to generalize from the training data set thus enabling it to produce acceptable result for previously unseen data.

III. Artificial Neural Networks

Artificial Neural Network (ANN) is a massively parallel adaptive network of simple nonlinear computing element called Neurons, which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strengths [3, 13, and 14].

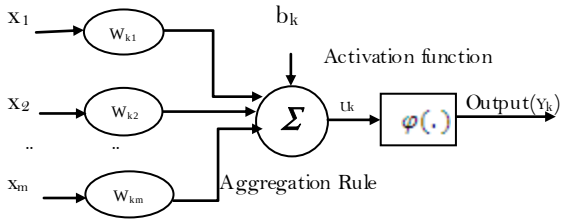

Fig. 2 Architecture of an artificial neuron

ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs.

In mathematical notation, any *neuron-k* can be represented as follows:

$$U_k = \sum_{j=1}^{m} W_{kj} X_j \quad \text{and} \quad Y_k = \varphi(U_k + b_k)$$

where $x_1, x_2, \ldots, x_m$ are the input signals, $w_{k1}, w_{k2}, \ldots, w_{km}$ are the synaptic weights of the corresponding neuron, u_k is the linear combiner output, b_k is the bias, $\varphi()$ is the activation function and y_k is the output signal of the neuron.

After an ANN is created it must go through the process of learning or training. The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called *training* a network. There are two approaches for training– *supervised* and *unsupervised* .In *supervised* training; both the inputs and the outputs are provided. The network then processes the inputs, compares its resulting outputs against the desired outputs and error is calculated. In *unsupervised* training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data [3].

Artificial Neural Networks Architecture:

Depending upon the architecture the ANNs can be categorized into following types, as shown below [19]:

(1)Feed-forward networks- A *feed-forward* ANN is the architecture in which the network has no loops.

A feed-forward networks can again categorized into following-

(a) Single-layer perceptron- A single layer perceptron consists of a single layer of output nodes, the inputs neurons are connected directly to the outputs neurons via a series of weights.

(b) Multilayer perceptron- In multi-layer perceptron an additional layer of neurons present between input and output layers. That layer is called *hidden* layer. Any number of hidden layers can be added in an ANN depending upon the problem domain and accuracy expected.

(c) Cascade network- Cascade network is a feed-forward neural network where the first layer will get signal from input. Each subsequent layer will receive signal from the input and all previous layers.

(2) Elman Networks- It is a feed-forward network with partial recurrence. Elman NN is a special type of recurrent neural network where an additional set of "context units" is connected with the input layer. These are also connected with the hidden layer with connection weight one. The context unit works as memory for the network.

(3)Recurrent/ Feed-back networks- In a recurrent (*feed-back*) ANN is an architecture in which loops occurs in the network.

Recurrent networks can have following types-

(a) Competitive networks- This neural networks is designed based on Competitive learning. This rule is based on the idea that only one neuron from a given iteration in a given layer will fire at a time. Weights are adjusted such that only one neuron in a layer, for instance the output layer, fire. Competitive learning is useful for classification of input patterns into a discrete set of output classes.

(b) Kohonen's neural networks- The Kohonen's neural networks differ considerably from feed-forward neural networks because it doesn't have any activation function and also it doesn't have a bias weight. It uses unsupervised learning for classification of input pattern presented to it.

(c) Hopfield networks- It is a fully recurrent network. The network is based on Hebb's Rule for learning. It can recall a memory, if presented with a corrupt or incomplete version of data.

(d) ART models- This neuron networks works according to adaptive resonance theory (ART).The theory has led to neural models for pattern recognition and unsupervised learning.

These models are capable of learning stable recognition categories. ART networks are fully-connected networks, in that all possible connections are made between all nodes.

IV. ANN IN EFFORT ESTIMATION

Artificial Neural Network is used in effort estimation due to its ability to learn from previous data. It is also able to model complex relationships between the dependent (effort) and independent variables (cost drivers). In addition, it has the ability to generalize from the training data set thus enabling it to produce acceptable result for previously unseen data. Most of the work in the application of neural network to effort estimation made use of feed-forward multi-layer Perceptron, Back-propagation algorithm and sigmoid function. However many researchers refuse to use them because of their shortcoming of being the "black boxes" that is, determining why an ANN makes a particular decision is a difficult task. But then also many different models of neural nets have been proposed for solving many complex real life problems [4].

To use ANN for effort estimation we have to follow these steps-

Steps in effort estimation

1. *Data Collection:* Collect data for previously developed projects like LOC, method used, and other characteristics.
2. *Division of dataset:* Divide the number of data into two parts – training set & validation set.
3. *ANN Design:* Design the neural network with number of neurons in input layers same as the number of characteristics of the project.
4. *Training:* Feed the training set first to train the neural network.
5. *Validation:* After training is over then validate the ANN by giving the validation set data.
6. *Testing:* Finally test the created ANN by feeding test set data.
7. *Error calculation:* Check the performance of the ANN. If satisfactory then stop, else again go to step (3), make some changes to the network parameters and proceed.

Once the ANN is ready we can give the parameter of any new project, and it will output the estimated effort for that project.

V. TYPES OF SOFTWARE

Software types can be categorized into:

1. Conventional/Procedure Oriented Software: Here the total project is divided into modules. All modules are developed separately. Then they are integrated to build complete software .Design & analysis documents are DFD and State-chart diagram. Metrics used are LOC, FPs, and COCOMO cost drivers.

2. Object Oriented Software: Project is developed taking OBJECT as main entity. OO features available, inheritance, abstraction .It is a top-down approach. Design & analysis documents are UML diagrams. Metrics used are no. of classes, weighted method/class (WMC), UML points...

3. Agile Software: Iterative and incremental development. Requirements and solutions evolve throughout the life cycle. Very popular now-a-days. All design documents are written in form of stories. Story points and sprints are used as metrics.

4. Component Based Software: In CBSD the total software is developed in forms of *components*. "A component is a coherent package of software that can be independently developed and delivered as a unit, and that offers interfaces by which it can be connected, unchanged, with other components to compose a larger system." Here the metrics used are component intensity, concurrency, fragmentation, etc.

5. Web Based Software: A web application is an application that is accessed over a network such as the Internet or an intranet. The term may also mean a computer software application that is coded in a browser-supported language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable.

TABLE II
KEY FEATURES OF DIFFERENT SOFTWARE

Conventional	Object Oriented Software	Agile Software	Component Based Software	Web Based Software
-procedure -record -module -procedure call -LOC -FPs	-method -object -class -message -Use case points -CPs	-business value delivered -story points -sprint	-Intensity -Concurrency -Fragmentation -Component -Project Experience	-http count -page count

VI. RELATED WORKS

We have reviewed works based on use of artificial neural networks in prediction of effort for Conventional Software and then for Object Oriented Software. First we present the conventional software development effort estimation, followed by object oriented software.

A. Conventional Software Review

G. E. Wittig, et al. [5] used a dataset of 15 commercial systems, and used feed-forward back-propagation multilayer neural network for his experiment. He had tried for numbers of hidden layers from 1-6, but found the best performance for only one hidden layer. Sigmoid function was used. He found that for smaller system the error was 1% and for larger systems error was 14.2% of actual effort.

In a paper by Ali Idri, et al. [4] he has used COCOMO-81 dataset and three layered back-propagation ANN, applying 13 cost drivers as inputs. Development effort taken as output. By

taking 13 neurons in hidden layer and after 300,000 iterations he found, average MRE is 1.50%.

F. Barcelos Tronto, et al. [2], also used COCOMO-81 dataset, but he has taken only one input, i.e TOTKDSI (thousands of delivered source instructions). All the input data were normalized to [0, 1] range. Here a feed-forward multilayer back-propagation ANN was used with the 1-9-4-1 architecture. The performance in MMRE found was 420, where as that of COCOMO and FPA was 610 and 103 respectively.

Jaswinder Kaur, et al. [6] implemented a back-propagation ANN of 2-2-1 architecture on NASA dataset consist of 18 projects. Input was KDLOC and development methodology and effort was the output. He got result MMRE as 11.78.

Roheet Bhatnagar, et al. [7] used MATLAB NN toolbox for effort prediction. He had used a dataset proposed by Lopez-Martin, which consists of 41 projects data. He has designed a 3-3-1 neural network, applied the Dhama Coupling (DC), McCabe Complexity (MC) and Lines of Code (LOC) as inputs. Development time was the only one output. From the experiment he found that the percentage of error during training, validation and testing was between +14.05 to -25.60, +12.76 to -18.89 and +13.66 to -15.75 respectively.

K.K. Aggarwal, et al. [8] had investigated for finding the best training algorithm. Here ISBSG repository data was used on a 4-15-1 feed-forward ANN. Four inputs were taken-FP, FP standard, language and maximum team size. SLOC was the only output. He had tried all training algorithm and concluded that 'trainbr' is the best algorithm. 'traingd' was found to be the next best algorithm.

TABLE III
RELATED WORKS

Author	Learning Algorithm	Dataset	No. of Projects	No. of Inputs	ANN Configuration
G. E. Wittig[5]	Back-propagation	Commercial Systems	15	-	[23-4-1]
Ali Idri [4]	Back-propagation	COCOMO	63	13	[13-13-1]
I.F. Barcelos Tronto [2]	Back-propagation	COCOMO	63	1	[1-9-4-1]
Jaswinder Kaur[6]	Back-propagation	NASA	18	2	[2-2-1]
Rpheet Bhatnagar[7]	Back-propagation	Lopez-Martin	41	3	[3-3-1]
K.K. Aggarwal [8]	Back-propagation	ISBSG	88	4	[4-15-1]

B. *Object- Oriented Software Review*

Object oriented technology is becoming very popular now-a-days, because of the features offered by it like Encapsulation, Inheritance, Polymorphism, etc. Modern software development technologies such as .Net and Java are rich of features hose are capable of developing highly maintainable, reusable, testable and reliable software [19]. Simple FPA and COCOMO model will not work for estimation of OO software.

Gennaro Costagliola, et al. [15] had proposed the concept of Class Point. In this approach he had presented a FPA like approach for OO software project. He had used two measurements of size, CP1 & CP2. CP1 is for estimation at the beginning stage of development and CP2 is for later refinement when more information is available. He had considered three metrics No. of External Methods (NEM), No. of Service Routines (NSR) and No. of Attributes (NOA) to find the complexity of a class. Here he had proposed 18 system characteristics to find Technical Complexity. From the experiment over 40 project dataset he found that the aggregated MMRE of CP1 is 0.19 and CP2 is 0.18.

Then Wei Zhou and Qiang Liu [16] in the year 2010 have extended the above paper in two ways. First they have added a size measurement named CP3 based on CPA. Second, in-order to improve the precision of estimation, they have taken 24 system characteristics instead of 18 in the previous one. From this they found that where the original CP1 gives MMRE 0.22, this give 0.19 and incase of CP2 it was 0.18, now it is 0.14.

S.Kanmani, et al. [17] has used the same CPA with a little change, by using neural network in mapping the CP1 and CP2 into effort. In the base paper of Gennaro [15], he had used regression method to find the values of the constants that can be multiplied and added with computed CP1 and CP2 to find the effort. Here in this paper Kanmani has used neural network to find those values. The aggregate MMRE is improved from 0.19 to 0.1849 for CP1 and from 0.18 to 0.1673 for CP2.

VII. PERFORMANCE CRITERIA

There are many performance criteria to evaluate the accuracy of any estimation. The Mean Magnitude Relative Error (MMRE) is a widely-accepted criterion in the literatures which is based on MRE. Root Mean Square Error (RMSE) is the next most popular evaluation criteria. In some of the papers Pred ( ) and BRE are also used for measuring the accuracy.

A. *Mean Magnitude Relative Error (MMRE)*

MMRE is frequently used to evaluate the performance of any estimation technique. It measures the percentage of the absolute values of the relative errors, averaged over the N items in the "Test" set and can be written as [6]:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} \left| (y_i - \hat{y}_i) \right| / y_i$$

where y_i represents the i^{th} value of the actual effort and $\hat{y}_i$ is the estimated effort. The MRE calculates each project in a dataset while the MMRE aggregates the multiple projects. The model with the lowest MMRE is considered the best.

B. *Root Mean Square Error (RMSE)*

RMSSE is another frequently used performance criteria which measures the difference between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error, as shown in equation given below [6]:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

C. Balance Relative Error (BRE)

BRE is another evaluation criterion for accuracy [9]:

$$BRE(\%) = 100 * |y_i - \hat{y}_i| / \min(y_i, \hat{y}_i)$$

D. Pred(l)

Another measure of Pred(*l*) was also adopted to evaluate the performance of the established software effort estimation models. It provides an indication of overall fit for a set of data points, based on the MRE values for each data point [19]:

$$\Pr ed(l) = k/N$$

Where N is the total number of observations and *k* is the number of observations with MRE less than or equal to l.

VIII. Experimental Setup For Estimation

Data Preparation

For simulation of ***conventional*** software projects we have used a standard dataset proposed by Lopez-Martin et.al. [7]. It consists of 41 system development projects data, where the Development Time (DT), Dharma Coupling (DC), McCabe Complexity (MC) and the Lines of Code (LOC) metrices were registered, as shown in table-IV. Since all the programs were written in Pascal, the module categories mostly belong to procedures and functions. The development time of each of the forty-one modules were registered including five phases: requirements understanding, algorithm design, coding, compiling and testing

In the next step we have experimented for effort estimation of ***object-oriented*** software. Here we have used data from 40 Java systems developed during two successive semesters. Detail data set is given in table-V.

ANN Preparation

In this experiment we have created one feed-forward neural network using MATLAB10 NN toolkit [12]. For the neural networks [3-5-1] architecture is used, i.e. 3 neurons in input layer, 5 neurons in hidden layer and 1 neuron in output layer, in table-VI. Training algorithm used is 'trainlm'. For training the dataset is divided into three divisions-for training 80%, for validation 10% and for testing 10%. Learning rate is set at 0.01.Stopping criteria were set by number of epochs as 1000 and goal as 0.00.

The ANN shown in fig.3 is created by MATLAB tool, will be applied to convention software dataset first and then the same one will be applied on object oriented software dataset.

TABLE IV
NASA Data set

MC	DC	LOC	DT
1	0.25	4	13
1	0.25	10	13
1	0.333	4	9
2	0.083	10	15
2	0.111	23	15
2	0.125	9	15
2	0.125	9	16
2	0.125	14	16
2	0.167	7	16
2	0.167	8	18
2	0.167	10	15
2	0.167	10	15
2	0.167	10	18
2	0.2	10	13
2	0.2	10	14
2	0.2	10	15
2	0.2	15	13
2	0.25	10	12
2	0.25	10	12
3	0.083	17	22
3	0.125	11	19
3	0.125	15	18
3	0.125	15	19
3	0.143	13	21
3	0.143	14	20
3	0.143	14	21
3	0.143	15	19
3	0.143	15	20
3	0.167	13	15
3	0.167	14	13
3	0.2	18	19
3	0.25	9	13
3	0.25	12	12
3	0.25	17	12
4	0.077	16	21
4	0.077	31	21
4	0.111	16	19
4	0.2	24	18
5	0.143	22	24
5	0.143	22	25
5	0.2	22	18

TABLE V
DATA SETS

Project No.	Effort	CP1	CP2	NEM	NOA	NSR
1	286	103.18	110.55	142	170	97
2	396	278.72	242.54	409	292	295
3	471	473.9	446.6	821	929	567
4	1016	851.44	760.96	975	755	723
5	1261	1263.12	1242.6	997	1145	764
6	261	196.68	180.84	225	400	181
7	993	718.8	645.6	589	402	944
8	552	213.3	208.56	262	260	167
9	998	1095	905	697	385	929
10	180	116.62	95.06	71	77	218
11	482	267.8	251.55	368	559	504
12	1083	687.57	766.29	789	682	362
13	205	59.64	64.61	79	98	41
14	851	697.48	620.1	542	508	392
15	840	864.27	743.49	701	770	635
16	1414	1386.32	1345.4	885	1087	701
17	279	132.54	74.26	97	65	387
18	621	550.55	418.66	382	293	654
19	601	539.35	474.95	387	484	845
20	680	489.06	438.9	347	304	870
21	366	287.97	262.74	343	299	264
22	947	663.6	627.6	944	637	421
23	485	397.1	358.6	409	451	269
24	812	676.28	590.42	531	520	401
25	685	386.31	428.18	387	812	279
26	638	268.45	280.84	373	788	278
27	18.3	2090.7	1719.25	724	1633	1167
28	369	114.4	104.5	192	177	126
29	439	162.87	156.64	169	181	128
30	491	258.72	246.96	323	285	195
31	484	289.68	241.4	363	444	398
32	481	480.25	413.1	431	389	362
33	861	778.75	738.7	692	858	653
34	417	263.72	234.08	345	389	245
35	268	217.36	198.36	218	448	187
36	470	295.26	263.07	250	332	512
37	436	117.48	126.38	135	193	121
38	428	146.97	148.35	227	212	147
39	436	169.74	200.1	213	318	183
40	356	112.53	110.67	154	147	83

TABLE VI
ANN SUMMARY

Architecture	
Layers	3
Input Neurons	3
Hidden Layer Neurons	5
Output Neurons	1
Training	
Training Function	Tansig
Algorithm	Back Propagation
Training Function	TrainLM
Parameters	
Learing Rate	0.01
Epochs	1000
Error	0.00

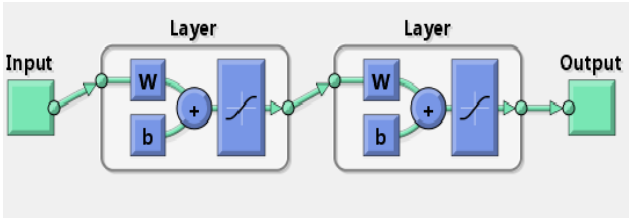

Fig. 3 The Feed-Forward ANN using MATLAB

IX. EXPERIMENTAL RESULTS

Conventional Software

The feed-forward NN was trained for 10 iterations and then the development time was predicted. We have considered ten randomly chosen projects for performance comparison.

TABLE VII
COMPARISON OF MULTIPLE REG. AND ANN

Project No.	Actual DT	Multiple Regression	MRE	Neural Network	MRE
3	9	8.2	0.09	9.31	0.03
4	15	18.2	0.21	16.01	0.07
5	15	16.62	0.11	14.85	0.01
14	13	14.34	0.10	14.57	0.12
19	12	12.7	0.06	12.53	0.04
20	22	19.91	0.10	21.06	0.04
21	19	18.83	0.01	20.64	0.09
34	12	14.41	0.20	12.9	0.08
35	21	22.22	0.06	20.58	0.02
39	24	21.81	0.09	24.2	0.01
MMRE			0.10		0.05

Table-VII presents comparison of prediction using multiple regression and neural network and fig.4 shows the MRE performance of both the techniques.

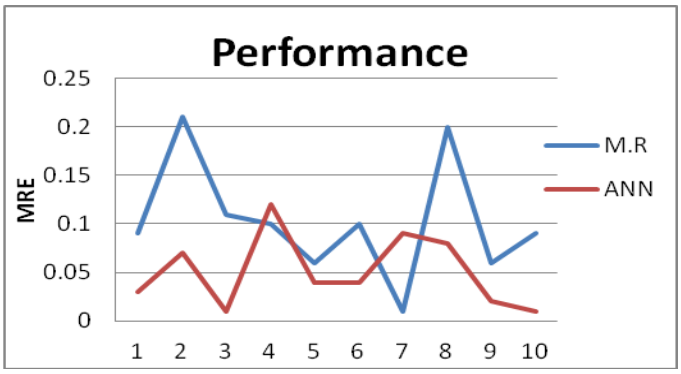

Fig. 4 MRE of Multiple Regression & ANN

Object-Oriented Software

After feeding data of 40 projects into the ANN and completing the training, then the ANN is used for prediction of results. We have partitioned the data into 4 sets taking 10 project data in each one. Then these data sets were given to the ANN for calculating the efforts. The table-VIII & table-IX given below compares the prediction of our ANN with those of CP1 and CP2.

TABLE VIII
COMPARISON OF CP1, CP2 & ANN

	Actual Effort	CP1	MRE	CP2	MRE	ANN	MRE
1	286	328.83	0.15	340.57	0.19	321.5	0.12
2	396	476.81	0.20	460.95	0.16	446.2	0.13
3	471	641.35	0.36	647.05	0.37	485.4	0.03
4	1016	959.62	0.06	933.75	0.08	1012.8	0.00
5	1261	1306.66	0.04	1373	0.09	1268.8	0.01
6	261	407.65	0.56	404.68	0.55	396.7	0.52
7	993	847.46	0.15	828.54	0.17	986.3	0.01
8	552	421.66	0.24	429.96	0.22	480.4	0.13
9	998	1164.94	0.17	1065.11	0.07	1016.5	0.02
10	180	340.16	0.89	326.45	0.81	163	0.09
MMRE			0.28		0.27		0.11

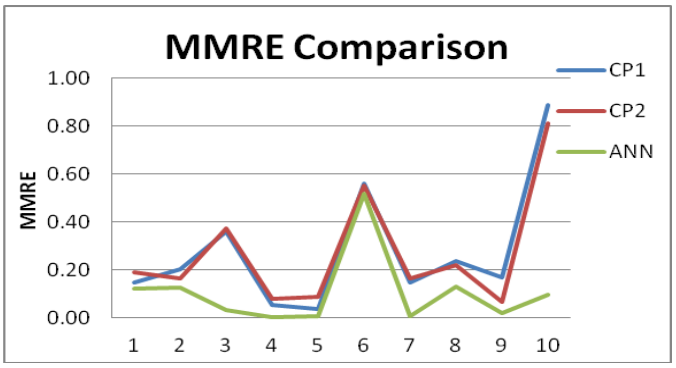

FIG. 5 MMRE OF CP1, CP2 & ANN

TABLE IX
AGGREGATED MMRE OF CP1, CP2 & ANN

	Aggregate MMRE
CP1	0.192
CP2	0.165
ANN	0.097

X. CONCLUSION

Estimation is one of the crucial tasks in software project management. There are different estimation techniques are present, but those are suitable for any one type of software development. We want to verify the applicability of ANN based estimation for software development effort, in all types of software. In this paper we have tested with conventional and object oriented software. We have observed that it gives good results on both types of software in comparison to multiple regression and class points analysis.

In further extension of this paper, we will test estimation with ANN design tool for effort estimation of Agile and Web-based software.

REFERENCES

[1] Stein Grimstad, Magne Jorgensen, Kjetil Molokken-Ostvold ,"Software effort estimation terminology: The tower of Babel", Elsevier, 2005.

[2] I.F. Barcelos Tronto, J.D. Simoes da Silva, N. Sant. Anna, "Comparison of Artificial Neural Network and Regression Models in Software Effort Estimation", INPE ePrint, Vol.1, 2006.

[3] Simon Haykin, "Neural Networks: A Comprehensive Foundation", Second Edition, Prentice Hall, 1998.

[4] Ali Idri and Taghi M. Khoshgoftaar& Alain Abran,"Can Neural Networks be easily Interpreted in Software Cost Estimation", IEEE Transaction, 2002, page:1162-1167.

[5] G.E. Wittig and G.R Finnic, "Using Artificial Neural Networks and Function Points to estimate 4GL software development effort", AJIS,1994, page:87-94.

[6] Jaswinder Kaur, Satwinder Singh, Dr. Karanjeet Singh Kahlon, PourushBassi,"Neural Network-A Novel Technique for Software Effort Estimation", International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010, page:17-19.

[7] Roheet Bhatnagar, Vandana Bhattacharjee and Mrinal Kanti Ghose, "Software Development Effort Estimation –Neural Network Vs. Regression Modeling Approach", International Journal of Engineering Science and Technology,Vol. 2(7), 2010,page: 2950-2956.

[8] K.K. Aggarwal, Yogesh Singh, Pravin Chandra and Manimala Puri, "Evaluation of various training algorithms in a neural network model for software engineering applications" , ACM SIGSOFT Software Engineering , July 2005, Volume 30 Number 4 , page: 1-4.

[9] Mrinal Kanti Ghose, Roheet Bhatnagar and Vandana Bhattacharjee , "Comparing Some Neural Network Models for Software Development Effort Prediction" , IEEE, 2011.

[10] Jagannath Singh and Bibhudatta Sahoo, "Software Effort Estimation with Different Artificial Neural Network", IJCA Special Issue on 2nd National Conference- Computing, Communication and Sensor Network (CCSN) (4):13-17, 2011.

[11] Satish Kumar, "Neural Networks: A Classroom Approach", Tata McGraw-Hill, 2004.

[12] Howard Demuth and Mark Beale, "Neural Network Toolbox-For use with MATLAB",User's Guide,Version-4,Page-5.28.

[13] N.K.Bose and P.Liang, "Neural Network Fundamentals with Graphs, Algorithms and Applications", Tata McGraw Hill Edition,1998.

[14] B. Yegnanarayana, "Artificial Neural Networks", Prentice Hall of India, 2003.

[15] Gennaro Costagliola and Genoveffa Tortora, "Class Point: An Approach for the Size Estimation of Object-Oriented Systems", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 1, JANUARY 2005,page 52-74

[16] Wei Zhou and Qiang Liu," Extended Class Point Approach of Size Estimation for OO Product", IEEE sponsred 2nd International Conference on Computer Engineering and Technology,2010,Vol-4 ,Page:117-122.

[17] S. Kanmani, J. Kathiravan, S. Senthil Kumar and M. Shanmugam," Neural Network Based Effort Estimation using Class Points for OO Systems",IEEE-International Conference on Computing: Theory and Application(ICCTA'07),2007.

[18] Magne Jørgensen and Torleif Halkjelsvik, "The Effects of Request Formats on Judgment-based Effort Estimation", Journal of Systems and Software, vol. 83, 2010, pp. 29-36.

[19] Mehwish Nasir, " A Survey of Software Estimation Techniques and Project Planning Practices", Seventh ACIS International Conference on Software Engineering,Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), IEEE,2006.

Jagannath Singh is a M.Tech research scholar at National Institute of Technology Rourkela India. He had received B.E from Biji Pattnaik University of Technology, Orissa. His research interest includes Software Engineering, Artificial Neural Networks and Microprocessor.

Bibhudatta Sahoo received the M.Sc. Engineering in Computer Science from National Institute of Technology Rourkela, INDIA, in 1999. He is currently an assistant professor in the Department of Computer Sc. & Engineering, NIT Rourkela, India. His interest include Parallel & Distributed Systems, Networking, Computational Machines, System Software, High performance Computing, VLSI algorithms He is a member of the IEEE Computer Society & ACM.