UML Based Object Oriented Software Development Effort Estimation Using ANN

Jagannath Singh
Department of Computer Sc. and Engineering
National Institute of Technology Rourkela
Rourkela, India
E-mail: jagannath.singh@gmail.com

Bibhudatta Sahoo
Department of Computer Sc. and Engineering
National Institute of Technology Rourkela
Rourkela, India
E-mail: bdsahu@nitrkl.ac.in

Abstract—**Today the software industry may be the fastest growing industries in the world. As the industry grows the size and cost of the software are also increasing. Hence there is need of effective techniques for cost estimation in order to control the costs and make the software more competitive. The software development techniques are also changing, and now- a- days most of the companies using object – oriented techniques for software development. UML diagrams are used for analysis and design of a software and different UML constructs can be used at different stages of software development for estimation of resources like efforts & cost etc. Use case point and class point analysis were accepted for the estimation of object-oriented software. In class point analysis some unknown constants are multiplied with known values of product attributes, like no. of methods, classes etc. These add impurities to the estimation model. So in this paper we have applied Artificial Neural model for estimation directly from known attributes and we find that it gives good results than class point analysis.**

Keywords-Effort estimation; Artificial Neural Networks; Use case Points;Class Points; NEM; NSR; NOA.

I. INTRODUCTION

Now-a-days Object oriented technology is becoming very popular in software development industry. It is because of the features offered by OO programming like Encapsulation, Inheritance, Polymorphism, Abstraction, Cohesion and Coupling. Modern OO software development technologies such as .NET and Java are rich of features those are capable of developing highly maintainable, reusable, testable and reliable software [16].

Apart from the advantages of OO technique mentioned above, these features are producing obstacles for cost and effort estimation of the software. Traditional software estimation techniques like COCOMO and Function Point Analysis have proven unsatisfactory for measuring cost and effort of all types of software [4]. The Lines of Code (LOC) and the Function Points (FPs) were both used for procedure oriented software projects. But this concept conflicts with the object-oriented paradigm. Procedure oriented design split data and procedures while object-oriented design combines them. There are multiple dimensions that an OO metric must have if it is to provide accurate effort or cost prediction. It is important to measure the amount of raw functionality the software delivers, but it is equally important to include information about communication between objects and reuse through inheritance in the 'size' as well.

Need of accurate estimation is always important for bidding for a contract or determining whether a project is feasible in the terms of a cost-benefit analysis. In present scenario most of the project planning depends upon estimates. Here we are trying to develop an ANN based estimation model similar to Class Point Analysis, but with better prediction accuracy.

II. REVIEW OF UML BASED EFFORT ESTIMATION

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).

The OO paradigm presents some critical challenges to software effort and cost estimation. Since object oriented development methods are not perfectly match with the way software is developed outside of the OO paradigm. OO development requires not only different approach to design and implementation, it requires a different set of software metrics.

Function Point metrics are used for traditional procedural paradigm software, but those are not efficient enough to describe OO software. To find the solution we have to move upstream in the software development process to requirement analysis and design. Currently UML diagrams are widely used in the software development industry for requirement analysis and detail design before going for coding.

A. UML Diagrams

UML is a modeling tool which is used by software professionals all over the world for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. It is a graphical notation used for object-oriented analysis and design [2]. One of the purposes of UML was to provide the development community with a stable and common design language that could be used to develop and build computer applications.

B. *Use of UML Diagrams in Estimation*[17]

Macro lifecycle:

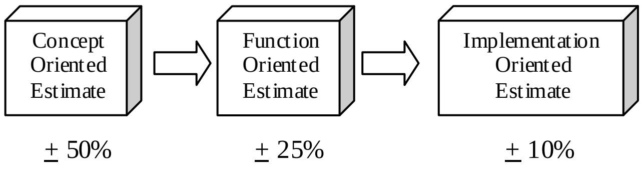
Concept Oriented Estimate
Function Oriented Estimate
Implementation Oriented Estimate
± 50%
± 25%
± 10%

Figure 1. Typical cost estimating accurecies

Initially, at the beginning stage, we may be presented with a vague definition of the project. Though the requirements may not yet be fully understood, the general purpose of the new software can be recognized. At this point, estimates with an accuracy of plus or minus fifty percent are typical for an experienced estimator using informal techniques (i.e., historical comparisons, Delphi Method, and so on). After the requirements are reasonably well understood, a function-oriented estimate can be prepared. At that point, estimates with an accuracy of plus or minus twenty-five percent are typical for an experienced estimator, using above techniques. Finally, after the detailed design is complete, an implementation-oriented estimate may be prepared. This estimate is typically accurate within plus or minus ten percent. So we can conclude that estimation after detailed design is usually more accurate.

UML is a widely recognized software design tool. There are several diagrams in UML, but we focus mainly on use case diagram and class diagram. While functional requirements are captured through use case diagram, detail design information is provided in class diagrams [2].

C. *UML based estimation techniques*

USE CASE POINTS

The early work starts from use case points, which was first given by Karner in 1993 as a software project effort estimation model[2]. Use case diagrams contain the functional behavior of the target system, determined during the requirement analysis phase. UCP effort estimation is an extension of existing estimation methods, such as function point analysis. In this process, first of all actors are characterized as simple, average or complex and the total unadjusted actor weight (UAW) is calculated by counting the number of actors in each category, multiplying each total by its specified weighting factor, and then adding the points. Next, categorization of the use cases is done as simple, average or complex, depending on the number of transactions, including the transactions in alternative flows. Then the unadjusted use case weights (UUCW) are calculated by Next, the use case points are adjusted based on the values assigned to a number of technical factors and environmental factors. Each factor is assigned a value between 0 and 5 depending on its assumed influence on the overall project. This step has 3 different formulae:

1. The Technical Complexity Factor:
   $TCF = 0.6 + (.01 * Tfactor)$
2. The Environmental Factor:
   $EF = 1.4 + (-0.03 * Efactor)$
3. Adjusted use case points:
   $UCP = UUCP * TCF * EF$

CLASS POINTS

The Class Point approach provides a system-level estimation of the size of OO products [2]. It has been conceived by recasting the ideas underlying the FP analysis within the OO paradigm and by suitably combining well-known OO measures based on design documentation. In particular, two measures are proposed, named CP1 and CP2. CP1 is meant to be used at the beginning of the development process to carry out a preliminary size estimation, which can be later refined by applying CP2 when more information is available. The Class Point size estimation process is structured into three main phases, corresponding to analogous phases in the FP approach.

During the first step the design specifications are analyzed in order to identify and classify the classes into four types of system components, namely the problem domain type, the human interaction type, the data management type, and the task management type.

During the second step, each identified class is assigned a complexity level, which is determined on the basis of the local methods in the class and of the interaction of the class with the rest of the system. This is achieved by exploiting suitable measures for OO classes. The measures and the way they are used to carry out this step represent the substantial difference between CP1 and CP2. Indeed, in CP1 the complexity level of each class is determined on the basis of the Number of External Methods (NEM)[2] and the Number of Services Requested (NSR). In CP2, besides the above measures, the Number Of Attributes (NOA) measure is taken into account in order to evaluate the complexity level of each class.

Once a complexity level of each class has been assigned, such information and its type are used to assign a weight to the class. Then, the Total Unadjusted Class Point value (TUCP) is computed as a weighted sum. Finally, the Class Point value is determined by adjusting the TUCP with a value obtained by considering global system characteristics as in FPA.

1. The Technical Complexity Factor:
   $TCF = 0.55 + (0.01 * TDI)$
2. Class Points:
   $CP = TUCP * TCF$

D. *Comparision of Use Case Point and Class Point*

TABLE I. COMPASION[2]

METHOD	PROS	CONS
USE CASE POINTS	-Simple, easy to apply method. - Provides 13 technical, 8 environmental factors for guidance.	- Lacks information, only counting the number of actors and use cases. - Not addressing a change in requirements or use cases. - TCF & EF is outdated.
CLASS POINTS	- Exploiting many UML diagrams. - Method can be totally automated and its output is in terms of SLOC, so can be compared with well known standard like FP.	- Uses expert decision on component type, complexity level, degree of influence etc, which are highly dependent on the expert's decision. - Goes deep into UML design, i.e. sequence and state diagrams, hence makes it a late design activity instead of early analysis one.

III. MOTIVATION

Drawback of Class Point Analysis

In Class Point Analysis both the forms CP1 and CP2 are based on three object oriented metrics- NEM, NSR and NOA. The class complexities are measured through these three parameters. Then Total Unadjusted Class Points (TUCP) is computed by multiplying the no. of classes in each category with weighting values. Finally this is multiplied with Technical Complexity Factor (TCF) to produce the CPs. So the basis of Class Point Analysis is those three metrics. If we can predict the effort from these values only then the accuracy should increase. We have planned to use Artificial neural network for effort estimation of OO projects using OO metrics, i.e. NEM, NSR and NOA [2].

NEM

The Number of External Methods (NEM) measures the size of the interface of a class and is determined by the number of locally defined public methods.

NSR

The Number of Services Requested (NSR) provides a measure of the interconnection of system components. It is again applicable to a single class and is determined by the number of different services requested from other classes.

NOA

When computing CP2, the number of attributes is also taken into account in order to evaluate the complexity level of a class.

Can we use ANN?

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 [10].

IV. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Network (ANN) is a massively parallel adaptive network of simple nonlinear computing elements called Neurons, which are intended to abstract of the human nervous [9,14,15]. An artificial neural network comprises of eight basic components *(i)neurons, (ii)activation function, (iii)signal function, (iv)pattern of connectivity, (v)activity aggregation rule, (vi) activation rule, (vii) learning rule* and *(viii)environment* [12].

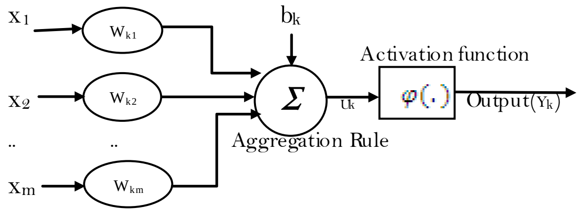

Figure. 2. Architecture of an artificial neuron

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

$u_k = \sum_{j=1}^{m} w_{kj} x_j$ and $y_k = \varphi(u_k + b_k)$

where x_1, x_2, …, 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, φ(.) 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 [14,15]. 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 [9].

A. *ANN Architecture*

Depending upon the architecture the ANN is of two types. A *feed-forward* ANN, is the architecture in which the network has no loops. But *feed-back* (recurrent) ANN is an architecture in which loops occurs in the network [14,15]. An ANN can be a single-layer perceptron or a multi-layer perceptron. In 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. But 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. In this paper we have used multiple layer feed forward ANN for simulation.

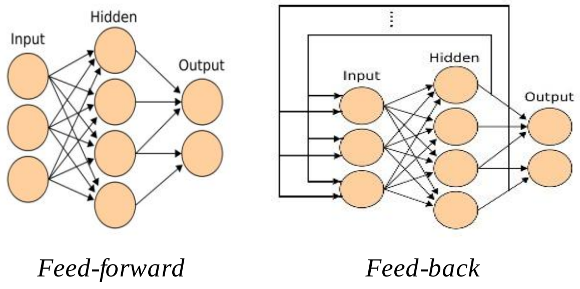

Figure. 3. Architecture of an artificial neuron network(ANN)

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 [11] . The 7 steps for effort estimation using ANN can be summarized as follows:

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 with the validation set data.
6. *Testing:* Finally test the created ANN by feeding test dataset.
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, simulation with the ANN can be conducted with the parameter of any new project, as show in fig.4 and it will output the estimated effort for that project.

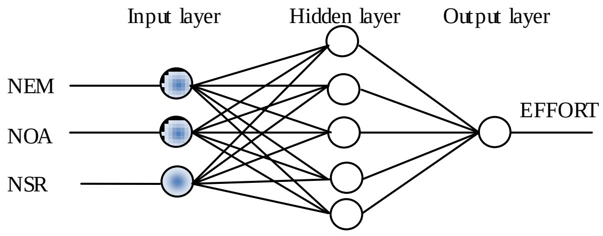

Figure. 4. Diagram of feed-forward multilayer ANN used in this paper

V. RELATED WORKS

Gennaro Costagliola, et al.[1] 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 NEM, NSR and 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.

Wei Zhou and Qiang Liu [6] in the year 2010 has 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. [7] 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[1],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.

VI. EXPERIMENT

A. Dataset Preparation

We have used data from 40 Java systems developed during two successive semesters of graduate courses on Software Engineering [1]. Table II reports the data of the 40 projects, having attributes Effort (EFH), CP1, CP2, NEM, NSR and NOA. Thus, the first ten projects form the first test set, the subsequent ten projects form the second one, and so on.

B. ANNs Preparation

In this experiment we have created a feed-forward neural network, as shown in fig.5 and compare it's performance with Class Point Approach. MATLAB10 *NN tool* is used for this experiment. 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. Training algorithm used is 'trainlm'. For training the dataset is divided into three divisions-for training 32(80%), for validation 4(10%) and for testing 4(10%). Stopping criteria was set by number of epochs as 1000 and goal as 0.00.

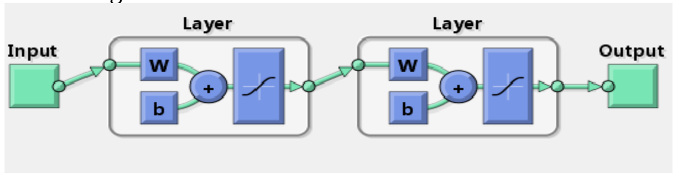

Figure 5. The Feed-Forward ANN using MATLAB

TABLE II. DATASET

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	1803	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

VII. SIMULATION & RESULTS

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 tables given below compares the prediction of our ANN with those of CP1 and CP2.

TABLE III. ESTIMATION FOR SET 1

	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

TABLE IV. ESTIMATION FOR SET 2

	Actual Effort	CP1	MRE	CP2	MRE	ANN	MRE
1	482	459.51	0.05	458.95	0.05	478	0.01
2	1083	801.62	0.26	941.26	0.13	1080.5	0.00
3	205	289.86	0.41	283.79	0.38	233.8	0.14
4	851	809.86	0.05	804.28	0.05	847.9	0.00
5	840	945.64	0.13	919.9	0.10	820.5	0.02
6	1414	1371.1	0.03	1483.89	0.05	1404.2	0.01
7	279	349.28	0.25	292.83	0.05	300.6	0.08
8	621	689.95	0.11	615.53	0.01	550.2	0.11
9	601	680.83	0.13	668.27	0.11	619.4	0.03
10	680	639.84	0.06	634.5	0.07	686.5	0.01
MMRE			0.15		0.10		0.04

TABLE V. ESTIMATION FOR SET 3

	Actual Effort	CP1	MRE	CP2	MRE	ANN	MRE
1	482	459.51	0.05	458.95	0.05	478	0.01
2	1083	801.62	0.26	941.26	0.13	1080.5	0.00
3	205	289.86	0.41	283.79	0.38	233.8	0.14
4	851	809.86	0.05	804.28	0.05	847.9	0.00
5	840	945.64	0.13	919.9	0.10	820.5	0.02
6	1414	1371.1	0.03	1483.89	0.05	1404.2	0.01
7	279	349.28	0.25	292.83	0.05	300.6	0.08

8	621	689.95	0.11	615.53	0.01	550.2	0.11
9	601	680.83	0.13	668.27	0.11	619.4	0.03
10	680	639.84	0.06	634.5	0.07	686.5	0.01
MMRE			0.15		0.10		0.04

TABLE VI. ESTIMATION FOR SET 4

	Actual Effort	CP1	MRE	CP2	MRE	ANN	MRE
1	484	473.47	0.02	450.26	0.07	434.6	0.10
2	481	636.98	0.32	609.09	0.27	409.89	0.15
3	861	893.98	0.04	910.27	0.06	853.38	0.01
4	417	451.19	0.08	443.49	0.06	463.9	0.11
5	268	411.42	0.54	410.45	0.53	379.17	0.41
6	470	478.26	0.02	470.31	0.00	527.52	0.12
7	436	325.72	0.25	343.87	0.21	282.9	0.35
8	428	351.02	0.18	364.19	0.15	447.08	0.04
9	436	370.56	0.15	412.06	0.05	384.86	0.12
10	356	321.47	0.10	329.34	0.07	372.07	0.05
MMRE			0.17		0.15		0.15

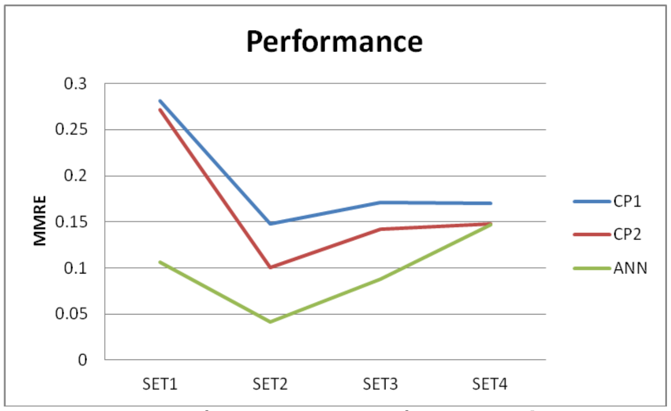

Figure 6. Performance comparison of CP1, CP2 and ANN

TABLE VII. PERFORMANCE COMPARESION

	Aggregate MMRE
CP1	0.192
CP2	0.165
ANN	0.097

VIII. CONCLUSION

Estimation is one of the crucial tasks in object oriented software project management. In this paper we have tried to use UML class diagram documents for effort estimation and compare with those of Class Points Approach. Here we have used MATLAB10 NN toolbox for creating and training a feed-forward ANN. The results from our simulation shows that the aggregated MMRE of CP1 is 0.192 , CP2 is 0.165 and that of ANN is found to be 0.097. This shows that a Feed-forward neural network give the better performance than that of CP1 or CP2. We have experimented with one of the dataset and further investigation can be done with other datasets.

REFERENCES

[1] 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

[2] Vineet Khera,"UMLBased Effort Estimation In Component Based Systems", M.E. Thesis,Thapar University,June2009.

[3] Anda B., Benestad H. C., Hove S. E., "A multiple-case study of software effort estimation based on use case points", 2005 International Symposium, p 407-416, 2005 .

[4] Arlene F. Minkiewicz, "Measuring Object Oriented Software with Predictive Object Points", Report prepared by PRICE Systems,LLC, 1997.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] Jagannath Singh and Bibhu Datta Sahoo, "Software Effort Estimation with Different Artificial Neural Network ", IJCA Special Issue on-2nd National Conference- Computing, Communication and Sensor Network, Sl. No.27, 2011.

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

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

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

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

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

[16] Nasib S. Gill and Sunil Sikka, "New Complexity Model for Classes in Object Oriented System", ACM SIGSOFT Software Engineering Notes, Sept 2010, Vol-35, No-5.

[17] WilliamRoetzheim, "Estimating Effort Using Use-Case and UML Class-Method Points", UML World International Conference,2006.