Optimization of model for estimating software effort Performance analysis of the software cost estimation methods

G.S.Aditya Rao[1], Dr.Ravindra Gupta[2],

[1] *M.Tech Scholar, Computer Science Department, SRK University , Bhopal, India*
[2] *Professor, Computer Science Department, SRK University , Bhopal, India*

Abstract

In software engineering, estimations are frequently used to determine expected but yet unknown properties of software development processes or the developed systems, such as costs, time, number of developers, efforts, sizes, and complexities. Plenty of estimation models exist, but it is hard to compare and improve them as software technologies evolve quickly. We suggest an approach to estimation model design and automated optimization allowing for model comparison and improvement based on commonly collected data points. This way, the approach simplifies model optimization and selection. It contributes to a convergence of existing estimation models to meet contemporary software technology practices and provide a possibility for selecting the most appropriate ones.
.

Keyword *Estimation Models, Optimization, Software Engineering.*

1.1 INTRODUCTION

Software cost estimation is the summation of predictions of both building effort and calendar time used to develop a software project. The building effort includes the summation of working hours and the total of workers included in the process of soft project development. Just from the inception of software project development, organizations of this nature came across to the problem of poor estimations of development effort and development time of software projects. A good reason for this was and which is persistent even this time is the availability of vague information about the software project to be developed at the time of its estimation process. A better estimate of software product is the only thing that can let any software development project manager to evaluate the project progress, gives him / her good track of potential cost control and accuracy in delivery time. This in widespread, however, gives the organization a better insight of resource utilization and consequently will land the organization in a better schedule of its futuristic projects. For this purpose, a good number of software cost estimation techniques from last so many decades have been proposed which differ from algorithmic

1.2 OBJECTIVES

The aims of this study, alongside the objectives to achieve these aims are stated as follows:

To provide a better conceptual understanding of reason of "cost overruns" which is ascertained through an extensive review of the literature, the factors that contribute to the difference between the initially estimated cost and the resulting final costs at project completion.

To develop an efficient software cost estimation model to forecast likely total cost of projects including its time schedule based on past data of completed projects. The various things to attain it include:

- Identification and collection of reliable datasets for the cost modelling process;
- Establishment of an artificial neural network modelling protocol and its hybridization with a meta heuristic algorithm for developing accurate software cost estimation model.
- Validation of the proposed model using precise evaluation criterions like magnitude of relative, mean magnitude of relative error and median of magnitude of relative error and its comparative evaluation with other existing techniques.

1.3 CONTRIBUTIONS

The contributions of this research to the field of software cost estimation are aligned with the research aims described previously. But the reader of this thesis deserves to understand the intentions, the argument and the original contribution of the work

1. A Software Cost Estimation model based on Input Selection Procedure & Artificial Neural Network.

2. A Software Cost Estimation model based on Functional Link Artificial Neural Network & Improved Particle Optimization algorithm.
3. A Software Cost Estimation model based on Artificial Neural Network Model

2.1 INTRODUCTION

Software cost estimation is a practice of developing an estimate of financial as well astime scheduling resources of any software project to be developed. The various such software development activities are given by [PMBOK, 2000]. Moreover, the various techniques and tools used, given inputs, and expected outputs for any general cost estimate are given below in figure 2.1.

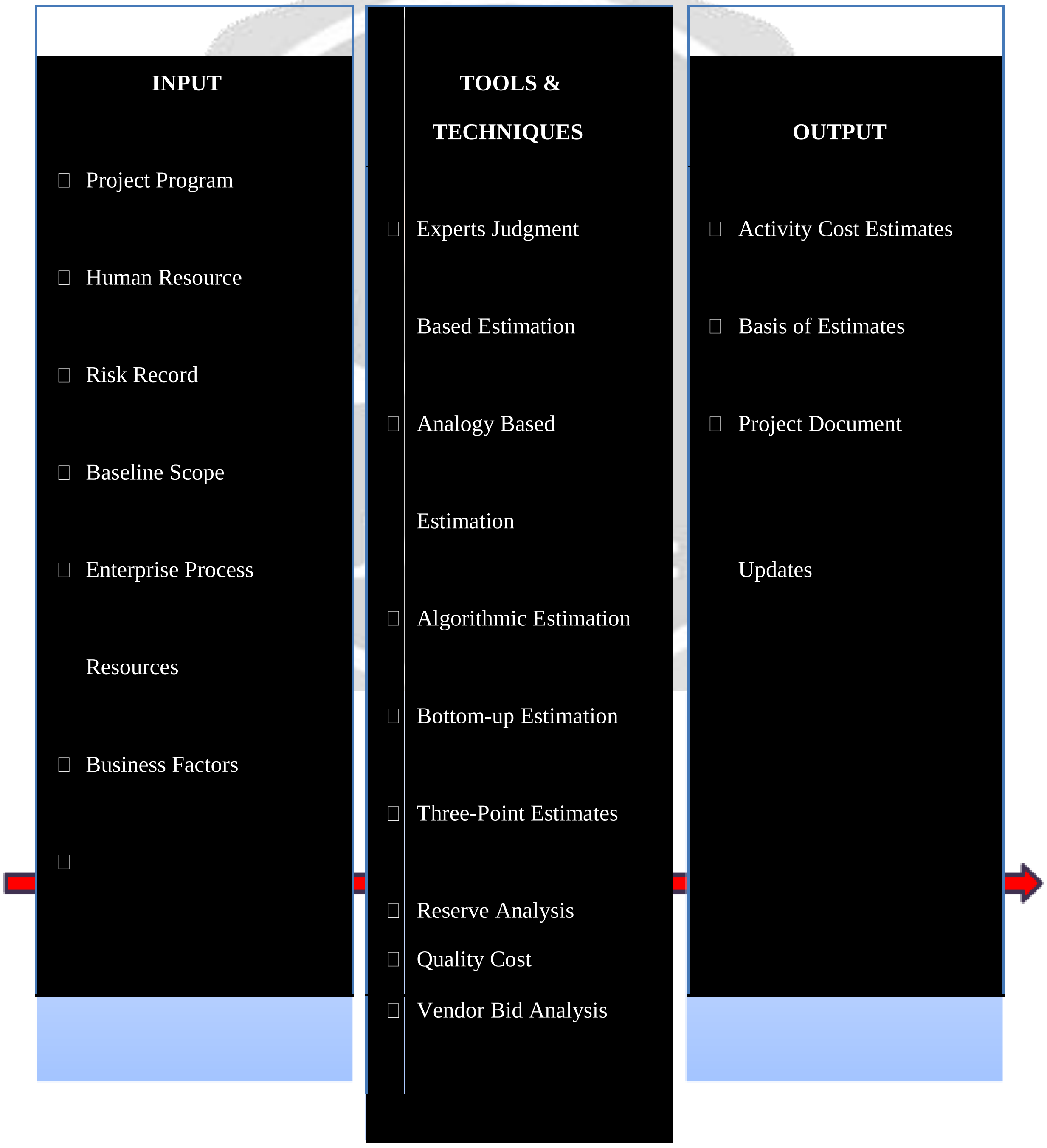

Figure 2.1: Project Management with respect to Cost Estimation

Origin of an estimate, update of a project document and estimates of activity cost are various important outputs of a software project. A Project document is an output related to project management and includes updates of all documents of the project. These updates include risk register as already depicted in figure In long survey of review of literature of software estimation, hardly any difference is found to be maintained between software development effort and software development cost. A big reason for this is the direct dependency of all kinds of costs, because of their personal nature on software development effort. In worldwide software development process, there exists a good probability of varying rates of software costs between different places, thus concluding that effort given at one place might be having expressive higher costs than same effort to be given at some other place [Conchúir etal., 2009]. So, in this research study costs and efforts are used as substitutes if notexplicitly specified.

2.1.1 BASIS OF ESTIMATION

The Basis of estimation necessarily must report a complete and unambiguous understanding of how the derivation of cost estimate is done [PMBOK, 2000] as shown in figure 2.1, the information of extra details to be required, including their quantity and kind and more precisely their varying nature as per applications. The various extra details for activity cost estimates include the following:

- A complete estimate basis documentation showing how the estimate was prepared.
- A full blueprint of estimate coverage representing what to estimate.

4. A thorough record of assumptions to be made.
5. Record of constraints.
6. Definition of self-assurance scale of final estimate.
7. Declaration of possible range of estimates that can be possible like an IT item is supposed to cost from a range of 1000 (INR) plus minus 10(INR)

So, it is important to mention here that it becomes easier to go through detailed reviews, to get easy revision of futuristic estimates only once the above information regarding the estimation is clearly defined and documented. However, in contrary to that an ambiguous basis of estimate lets the activity estimates to be useless and hopeless.

2.4 SOFTWARE COST ESTIMATION& SOFTWARE QUALITY

In this short section, we will discuss how software cost estimation affects software quality of any software development product. The term software quality is usually observed in a thin logic and in an isolated approach from its process of development. Software quality therefore becomes restricted to concepts like "usability", "error freeness", "extensibility", etc. Product quality thus discounts the two basic parameters of cost and time. Essential software quality, which may in real, be called as software process quality, on the other side, must cover these aspects too. So, the best software in terms of software quality process is often only good enough software in terms of quality of product. This section is aimed to be a support to integral concept of software quality and indirectly to quality of product.

2.4.1 REAL ESTIMATION, MEASUREMENT& ITS RELATIONSHIP

Same applies to software development projects too. Measuring just three economical parameters of effort, time and product can never lead to better software estimation process. Moreover, knowing the average ratio of product and effort, and the ratio between product and time, cannot even give the treasonable values for the parameters before estimation of at least one of them has not been completed. The first significant criteria towards estimation, is to form a model of the final product. A model of one such type is built by measurement of parameters: effort and time of modeling as well as the model‟s product metric is to be done. The second key thing to be done is to draw a correlation between the parameters of the modelling process to those of a complete process of production, with a hope that the earlier predict the latter.

2.4.2 PSYCHOLOGICAL AND ORGANIZATIONAL BARRIERS TO RATIONAL ESTIMATION

Why, in reality, is nobody concerned in a rational cost estimate?
Let us examine this question by looking at the people typically involved in the estimation process:
1.A manager of the software producing unit (SPU).
2.The software products client and
3.The project manager.
4.The SPU‟s manager always keeps his interest in marketing the project just to promote his bonus. Thereby, he is about to sell the software product to any price the customer is ready to pay. The customer, in turn, at every time is willing to pay the least possible. Another significant issue is the knowledge of "truth" regarding the cost prediction at early stage may considerably constrain the deployment of futuristic projects. In other words, we can say that if for any software project one had got the precise knowledge about the cost as early and rationally possible, will surely have a great impact on the decisions to be made regarding the inception or stopping of other forthcoming software projects.

2.5 SOFTWARE COST ESTIMATIONAND RISK MANAGEMENT

2.5.1 INTRODUCTION

Software cost and size estimation is a prediction problem of very high magnitude. Software cost estimation is restricted not only to software development companies but is expanded to software"s to be developed at other places like those of in Research and Development organizations, Aircraft organizations and many alike. Since size of software project is normally considered to be the most dominant factor in defining software"s cost, good size estimates of any software project size are precarious to good cost estimation.. The impression of risk is dominant to any such kind of analysis, and the two techniques that have got the ability to increase responsibility of holding risks related to software cost estimates include Identification of the zones of uncertainty that later convert and proceed towards risks and Comprehensive Analysis of whole process of software cost estimation process inorder to govern the areas where there is possibility of reduction in vagueness interms of its risk mitigation.

2.5.2 SIZING METHODS

Software cost estimate is a direct influence of software size estimation, thus selection of a right method for size estimation is equally important as prediction of software cost. In practise, it has been observed that in most of the software estimation cases, the estimation risk which is an absolute difference between the estimate cost and the actual software cost is predominantly dependant on accurate estimates of size than other drivers related to cost. Thereby, it becomes significant to mention that software sizing estimation needs to be done as consistent and accurate as possible.

3.2 MODEL BASED TECHNIQUES

The In model-based techniques, inorder to determine estimation of cost interms of effort in person months and calendar time of any software project to be developed, a mathematical model is used containing some cost factors to decide this estimation. The functionality of the mathematical model is usually dictated by the concept and testing of the mathematical models made for estimation of the currently going software project. However, in most of the cases, these models also want their calibration with past data from historical projects

3.2.1 PUTNAM'S MODEL

This model was given in 1970s by Larry Putnam [Putnam, 1978]. In this model, the Putnam uses a Rayleign curve function to define estimates of both effort and time needed to complete a precise sized software development project.

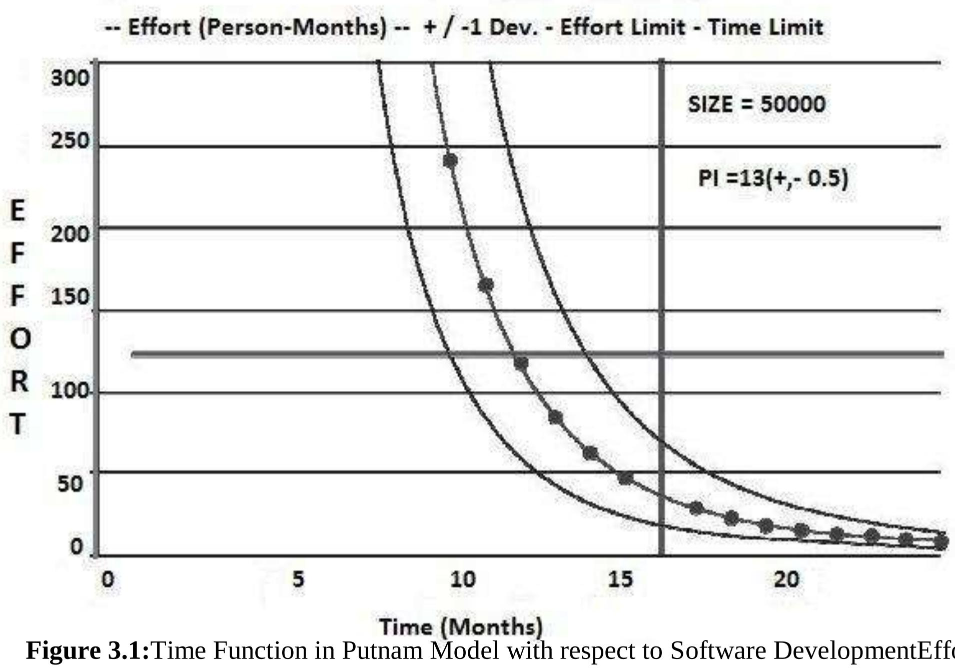

Figure 3.1: Time Function in Putnam Model with respect to Software DevelopmentEffort

The Putnam model can be adjusted with some adjustment questions if there exists non availability of data from completed projects of past. So in conclusion, simplicity interms of calibration of this model with history data becomes its significant property.

3.2.2 FUNCTION POINT ANALYSIS

This model was developed by Albrecht at IBM as a way to measure the amount of functionality in a system. They are derived from the requirements. Unlike lines of code, which capture the size of an actual product, function points do not relate to something physical but, rather, to something logical that can be assessed quantitatively. As shown in Figure 3.2, the function-point metric is calculated in two steps. First, a table like Table 3.1, which captures both data and transaction information, is used to calculate an initial function point count.

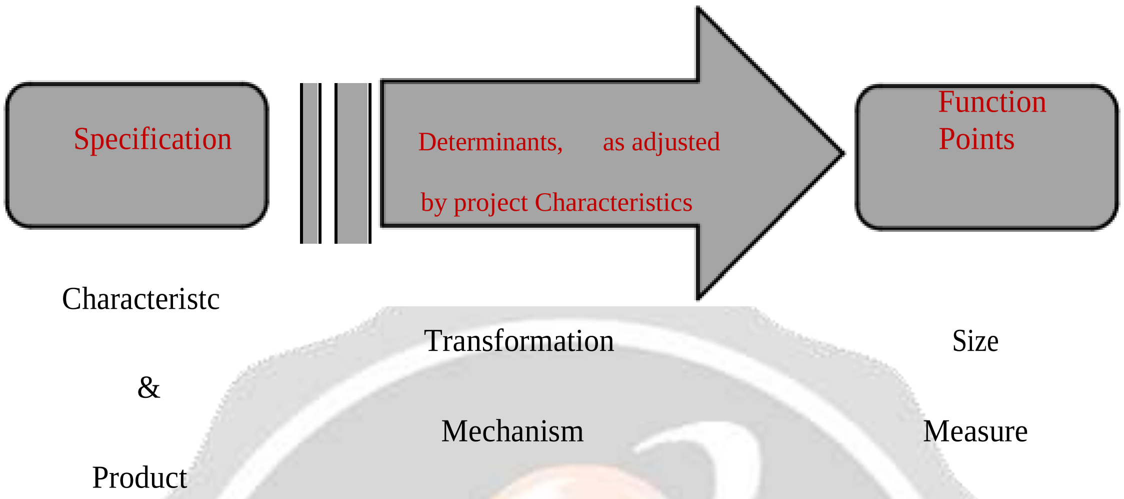

Figure 3.2: Transformation of Requirements into Function or Feature Points

Table 3.1: Initial Function-Point Count

Input Type	COUNT		
	Simple	**Average**	**Complex**
Number of External Inputs	**3**	**4**	**6**
Number of External Outputs	**4**	**5**	**7**
Number of External Queries	**3**	**4**	**6**
Number of Internal Logical Files	**7**	**10**	**15**
Number of External Interface Files	**5**	**7**	**10**

For each of the row"s count in column second is simply multiplied by a suitable weight as given in column 3, column 4, and column 5 to produce a numerical value that represents the mandatory quantity of functionality backed by that row. These five numbers (weighted) are later summed to produce an "unadjusted function point" count.

In second step adjustment of initial count is done by using features that let the project to be either extra or less challenging than a typical one. The project is next categorised by making use of a six-stage scale with no influence represented with a weight of 0, incidental represented with a weight of 1, moderate represented with a weight of 2, average represented with a weight of 3, significant represented with a weight of 4 and essential represented with weight of 5 inorder to answer every question as given below in a set of 14 questions:

1. Whether the system needs reliable recovery and backup?
2. Does the system require data communications?
3. Are there processing functions distributed?
4. Is performance needed to be critical?
5. Is there any possibility of the system run in under heavily loaded operational environment?
6. Does the system need on-line data entry?
7. Does the data entry, if on-line, need the input transaction to be developed over many operations or screens?
8. Is there online Updation of master files?
9. Are the files, inquiries, inputs and outputs to be complex?
10. Whether the internal processing complex?
11. Whether code is reusable?
12. Does installation and conversion to be part of design?
13. Whether the system made for different organizations with several installations?
14. Whether the application is made to simplify variation and user-friendliness by the user?

3.4.2 NEURAL NETWORKS

Techniques or models based on the principles of learning are called as neural networks. Neural networks are categorized in terms of three basic entities which include the core processing element called as neuron, the interconnection structure and the learning algorithm. Performance of an Artificial Neural Network is defined by its basic architecture which includes various parameters like number of hidden layers in an ANN, the neuron (node) count in each of these layers, transfer function used at each node, weights and parameters of the training algorithm used including their settings too. A general model for software cost estimation based on Artificial Neural Networks is shown below:

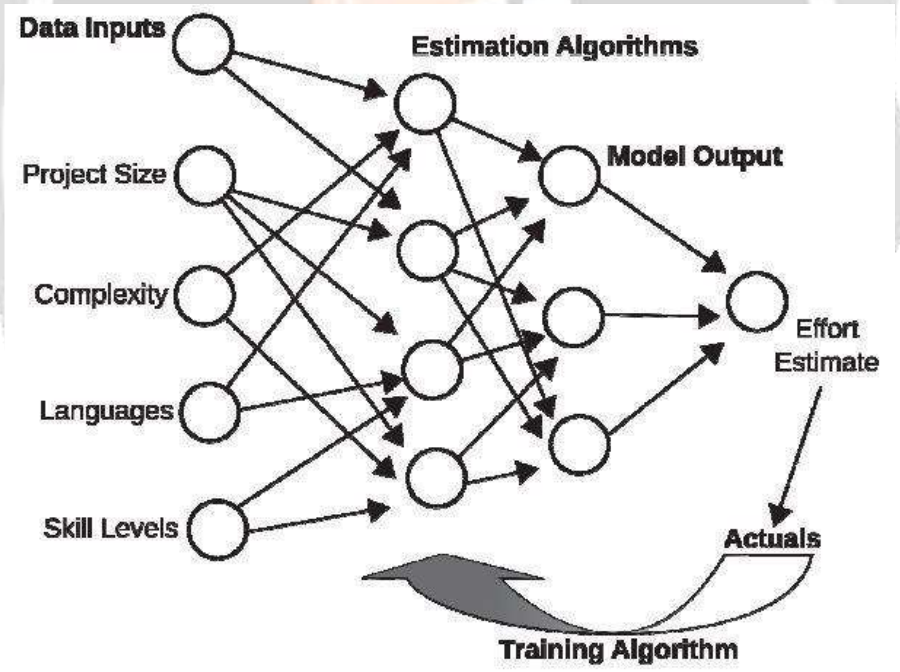

Figure 3.9: Neural Network for Software Cost Estimation

3.5 DYNAMIC BASED TECHNIQUES

Dynamics-All techniques of software cost estimation studied so far are static with respect to time or duration in which software development process is carried out. However, a single technique that acknowledges the change of cost factors like software development effort corresponding to duration of software development process is Dynamic based technique of software cost estimation.

4.1 SOFTWARE COST ESTIMATION BASED ON FUNCTIONAL LINK ARTIFICIAL NEURAL NETWORK MODEL& IMPROVED PARTICLE SWARM OPTIMIZATION

In this study, software development effort and development time is estimated by choosing a functional link artificial neural network. A functional link artificial neural network is a high order, single-layer feed forward artificial neural network. It consists of a layer for taking inputs, called as input layer and a layer for dispatching output, called as output layer. Moreover, it contains no hidden layers as the whole processing to be done on inputs in hidden layers is primarily done using functional expansion before being passed to the input layer. Output interms of software cost estimation in our study is generated by the functional link artificial neural network by simply intensifying the input (cost drivers). Every single input neuron keeps a correspondence to a component of input vector. The output layer on the other hand is composed of a single output neuron that calculates the software development effort as a linear weighted sum of the outputs originating from the input layer.

5.2 DATA SETS USED IN THE EXPERIMENT

For the purpose of assessing the software cost estimation using the proposed artificial neural network based models, four data sets from different companies are chosen. One of the data sets has been got from the study of Mair et al. In this study, 32 data sets[Mair, C.,*et*.al, 2005] were available publicly among which only one data set COCOMO 81 has been selected as this is the lone dataset containing data of more than 50 software development projects.

5.4 EXPERIMENTATION RESULTS& THEIR ANALYSIS

This subsection presents the experimentation results of the proposed technique discussed in section 4. Here in this subsection, we‟ll first present the results obtained while implementing the software cost estimation model based on hybrid of input selection procedure and artificial neural network model. Next we‟ll present the obtained experimentation results of software cost estimation model based on functional link artificial neural network and improved particle swarm optimization.

Table 5.2

MRE (%) of Proposed Model and other Two Existing Techniques on 11 randomly Selected Projects of COCOMO81 Dataset.

S No	Project ID	MRE (%) on COCOMO dataset		
		COCOMO II Model	B.T Rao, *et al.* Model	Proposed Model
01	05	7.44	5.23	3.44
02	12	19.83	9.18	7.12
03	30	6.49	3.21	2.22
04	38	50.98	14.40	11.10
05	40	12.40	4.89	3.89

Table 5.3

MRE of Proposed Model and other Two Existing Models on 10 Randomly Selected Projects of Cocnasa Dataset.

S No	Project ID	MRE (%) on Cocnasa dataset		
		COCOMOII Model	B.T Rao, *et al.*	Proposed Model
01	1	9.33	5.12	3.68
02	5	8.84	3.78	2.89
03	15	16.75	8.80	6.44
04	25	14.09	5.68	3.88
05	30	8.81	4.40	3.12

06	45	5.35	3.69	2.72
07	47	16.40	7.60	4.07
08	59	8.66	3.20	2.97
09	61	13.10	8.61	4.12
10	62	6.22	3.30	2.88

06	42	13.9	6.60	4.17
07	54	13.67	7.80	4.90
08	60	11.78	6.83	5.22
09	62	13.2	5.43	2.12
10	75	17.09	8.66	5.10

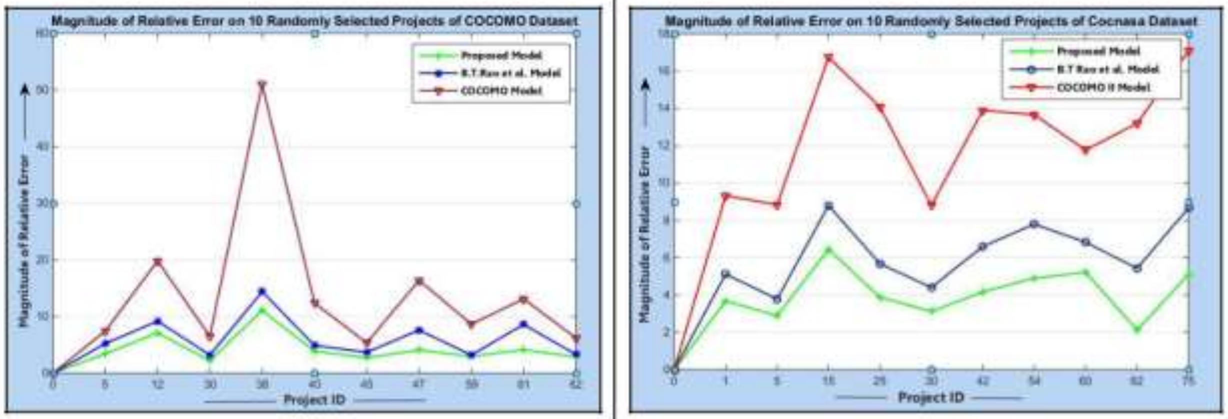

Figure5.1

MRE (%) Based Graphical Description of TwoExisting
Models and Proposed Model on Cocnasa Data Set

As Mean MRE is exposed to outliers, Median of MRE (MdMRE) is used as second evaluation criteria. The MdMRE of all of the three techniques was calculated for both of the datasets and is given in below

6.1 CONCLUSION

Designing a software system requires software effort estimation significantly. Numerous research works have been carried out to increase the precision of effort estimate of the software system. This paper has proposed a novel approach to estimate the software effort precisely. The approach has been contributed by neural network classification process and an optimization process. The neural network has classified various software parameters. For betterment of classification performance, ABC has been used to optimize the weights of neural network. Error parameters such as MRE, MMRE and MARE have been determined and performance comparison has been made with the existing method. The experimental outcomes have demonstrated the proposed system outperform the existing method in estimating the software effort more precisely.

REFERENCES

- Albrecht, A. (1979). Measuring Application Development Productivity. In Press, I., editor, IBM Application Development Symposium, pages 83–92.
- Beck, K. and Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd Edition). AddisonWesley Professional.
- Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., and Steece, B. (2009). Software Cost Estimation with COCOMO II. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition edition.
- Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD ’00, pages 93–104, New York, NY, USA. ACM.
- Brooks, Jr., F. P. (1995). The Mythical Man-month (Anniversary Edition). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Clark, B. (2015). COCOMO R III project purpose. http://csse.usc.edu/new/wp-content/uploads/2015/04/ COCOMOIII Handout.pdf. Accessed April 12, 2019.
- Fletcher, R. (1987). Practical Methods of Optimization; (2nd Edition). Wiley-Interscience, New York, NY, USA.
- Galorath, D. D. and Evans, M. W. (2006). Software Sizing, Estimation, and Risk Management. Auerbach Publications, Boston, MA, USA.
- Huang, L. (2017). A quasi-newton algorithm for large-scale nonlinear equations. Journal of Inequalities and Applications, 2017(1):35.
- Humphrey, W. S. (1996). Using a defined and measured personal software process. IEEE Software, 13(3):77– 88.
- Jenson, R. L. and Bartley, J. W. (1991). Parametric estimation of programming effort: An object-oriented model. Journal of Systems and Software, 15(2):107 –