

A Survey Of Effort Estimation Techniques For The Software Development

Anuj Khuttan, Ashwini Kumar, Archana Singh

Abstract: Effort estimation at the early stage of the software development is one of the most challenging parts of any organization. Many organization use different techniques to evaluate effort required for producing software, at the different levels of software life cycle model. There are various models like COCOMO, COCOMO II, Putnam model that have already used to estimate the software effort for projects. Researchers have proposed many new models to evaluate Effort. The objective of this paper is to compare Putnam model, COCOMO, and ANN-COCOMO and find out which technique give more accurate results.

Index Terms: ANN-COCOMO, COCOMO II, Putnam Model, Size.

1 INTRODUCTION

The ability to accurately and consistently estimate software development efforts, especially in the early stages of the development life cycle, is required by the project managers in planning and conducting software development. Several estimates are involved to effectively manage the software Effort. Estimation has become necessary for any community to develop useful models that estimate effort accurately. Putnam model is an empirical software effort estimation model developed by Lawrence H. Putnam. In this model effort estimates are made by providing size and calculating the associated effort^[1]. One of the most widely used techniques is COCOMO (constructive cost model) introduced by Barry Boehm in 1981, and is still in use by software engineering community^{[2][3]}. Software development efforts estimation is the process of predicting the most realistic use of effort required to develop or maintain software based on incomplete, uncertain and/or noisy input. ANN-COCOMO based software estimation neural networks proposed by Iman Attarzadeh and Siew Hock Ow, in there proposed neural network model, the accuracy of Effort estimation can be improved and the estimated cost can be very close to the actual Effort^[5]. The paper is organized as follows. Section 2 describes the various types of Effort Estimation technique like Putnam model, COCOMO model etc. Section 3 describes evaluation method and results of the different models and also calculation based on the data given by NASA. and finally, section 4 conclude this paper.

2 Estimation Techniques

Estimating accurate effort, cost, time required to develop a software etc. is a most challenging part of any organization. Many researchers have developed models to estimate Effort required building a project.

Some of them are as followed:

- (1) Putnam effort estimation (also known as SLIM)
- (2) COCOMO estimation model.
- (3) ANN-COCOMO based software estimation

2.1 Putnam effort estimation (also known as SLIM)

One of the popular software Effort estimation model is the Putnam model. The form of this model is:

$$\text{Technical constant } C = \text{size} * B^{1/3} * T^{4/3}$$

$$\text{Total Person Months } E = 1/T^{4*}(\text{size}/C)^3$$

T= Required Development Time in years

Size is estimated in LOC

C is a parameter dependent on the development environment and it is determined on the basis of historical data of the past projects.

Rating: C=2,000 (poor), C=8000 (good) C=12,000 (excellent).

The Putnam model is very sensitive to the development time: decreasing the development time can greatly increase the person-months needed for development One significant problem with the PUTNAM model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) of the software to be developed. There is often great uncertainty in the software size. It may result in the inaccuracy of effort estimation^[1].

2.2 COCOMO estimation model

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Bohem. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs and Effort but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases^[3].

COCOMO applies to three classes of software projects:

1. Organic projects – for "small" teams with "good" experience working with "less than rigid" requirements
2. Semi-detached projects – for "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
3. Embedded projects - developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects.

The basic COCOMO equations take the form

$$\text{Effort Applied (E)} = a(\text{KLOC})^b \text{ [person-months]}$$

$$\text{Development Time (D)} = c (\text{Effort Applied})^d \text{ [months]}$$

2.3 ANN-COCOMO based software estimation

In ANN-COCOMO based software estimation, neural networks are customized to accommodate the COCOMO II Post-architecture model. There are five scale input layer in the proposed neural network that corresponds to all Effort Multipliers (EM) and Scaling Factor (SF) as well as two bias values. In ANN-COCOMO based network is not a fully connected network but specified hidden layer nodes that take into account the contribution of EM and SF separately as shown in Fig. 1.

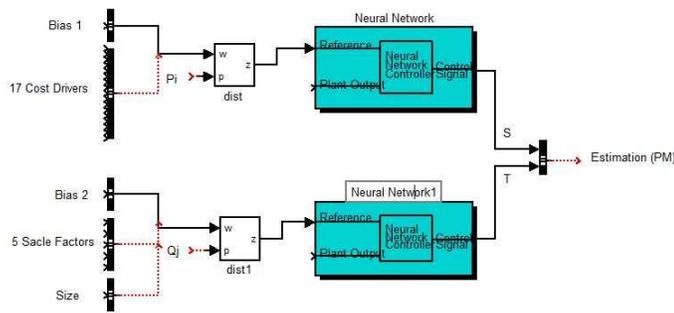


Fig1: COCOMO II model based on neural networks.

Note that in the above network, all EM_i values used in COCOMO II are pre-processed to $\log(EM_i)$ and the size of the product in KSLOC is not considered as one of the inputs to the network but as a cofactor for the initial weights for scale factors (SF). The activation function in the hidden layer is the sigmoid function defined by $f(x) = \frac{1}{1 + e^{-x}}$. The weights associated to the input nodes connected to the hidden layer are denoted by P_i for Bias1 and each input $\log(EM_i)$ for $1 \leq i \leq 17$. On the other hand, the weights associated with each SF_j input nodes to the hidden layer are $q_j + \log(\text{size})$ for $1 \leq j \leq 5$ and the bias denoted by Bias2. S and T as shown in Fig. 2 denote the weights of the arcs from the hidden layer nodes to the output node. The weights S and T are relevant to the values of the nodes in the hidden layer. The output node has the identity function associated with it. One new concept to traditional neural networks is the addition of $\log(\text{Size})$ to the weight q_j of SF's input, which adjusts the weights q_j . Another major difference in this network compared to traditional neural network is the training aspect of neural network. In order to accommodate the COCOMO II formula, we adjust the initial values of weights S and T to the offset of the values of the nodes in the hidden layer. However, once we collect sufficient data set, we can use the back-propagation method to train the system. Initially, if the data set is not available or not reliable, the weights in the network results the COCOMO II estimation using random generated input/output. The output of the network (PM) would be derived from COCOMO II, "1" by considering the initial values of Bias1 as $\log(A)$ and Bias2 as 1.01. The weights in the network are initialized as $p_i=1$ for $1 \leq i \leq 17$ and $q_j=1$ for $1 \leq j \leq 5$. Propagating the inputs forward, we get the values of nodes in the hidden layer as:

$$f(p_0 + \text{Bias1} + \sum_{i=1}^{17} p_i \cdot \log(EM_i)) = \text{sigmoid}(\text{Bias1} + \sum_{i=1}^{17} p_i \cdot \log(EM_i)) = \frac{1}{1 + e^{-x}}$$

$$f((q_0 + \log(\text{size}) * \text{Bias2} + \sum_{j=1}^5 (q_j + \log(\text{size}))(SF_j)) =$$

$$\text{Sigmoid}(\log(\text{size}) * (\text{Bias2} + \sum_{j=1}^5 SF_j)) = \frac{1}{1 + e^{-x}}$$

Then initialization of weights S and T as follow:

$$S = \frac{1}{1 + e^{-x}} \text{ and } T = \frac{1}{1 + e^{-x}}$$

The output of the network is calculated as:

$$PM = S * x + T * y = \frac{1}{1 + e^{-x}} = A * \text{size}^{1.01} * \prod_{i=1}^5 SF_i$$

Note that the initial values of S and T are adjusted so that the nodes in the hidden layer have same contribution to the output node PM. The following section describes the procedure for training the network [4].

3 Results And Discussion

3.1 Data

To compare all the model discussed above we have to take data form COCOMO NASA2/ Software cost estimation which have the data of different centers, 93 NASA projects between years 1971-1987 was collected by Jairus Hihn, JPL, NASA, Manager SQIP Measurement & Benchmarking Element [11].

3.2 Evaluation Method

For evaluating the different software effort estimation models, the most widely accepted evaluation criteria are the mean magnitude of relative error (MMRE) and probability of a project having a relative error of less than or equal to 0.25 (Pred(I)). The Magnitude of Relative Error (MRE) is defined as follows

$$MRE_i = \frac{|(ACTUAL_EFFECTIVE_PM_i - PREDICTED_EFFECTIVE_PM_i)|}{ACTUAL_EFFECTIVE_PM_i}$$

The MRE value is calculated for each observation i whose effort is predicted. The aggregation of MRE over multiple observations (N) can be achieved through the Mean

$$MMRE = \frac{1}{N} \sum_{i=1}^N MRE_i \text{ [4]}$$

3.3 CALCULATIONS

Effort calculated with the given NASA data

S.No	Size	Putnam (in PM)	COCOMO (in PM)	ANN COCOMO (in PM)
1	25.9KLOC	110.136	114.8	117.6
2	38KLOC	72	176.39	210
3	33.2KLOC	58	151.6	85
4	20KLOC	19.99	85.9	40
5	31.5KLOC	54.42	142.96	60

3.4 MRE

S.No.	Model	MMRE
1	COCOMO	0.0406
	ANN COCOMO	0.0238
2	COCOMO	0.592
	ANN COCOCMO	0.160

3.5 MMRE

S.No.	Model	MMRE
1	COCOMO	0.3163
	ANN COCOMO	0.0919

4 CONCLUSION

It is clear with our survey and calculations are that Putnam model is very sensitive with development time; as long as development time is decreasing Effort required to develop software is increasing. On the other hand COCOMO model must know the cost driver to predict more accurate results. At last, ANN-COCOMO the accuracy of Effort estimation can be improved and the estimated effort can be very close to the actual cost.

REFERENCES

- [1] Putnam, L. H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Vol. 4, No. 4, pp. 345 – 361, 1978.
- [2] Boehm B. W. "Software Engineering Economics", Englewood Cliffs, NJ, Prentice-Hall, 1981.
- [3] Boehm B., Abts, C., and Chulani, S., Software Development Cost Estimation Approaches – A Survey, University of Southern California Center for Software Engineering, Technical Reports, USC-CSE- 2000-505, 2000.
- [4] Imran Attarzadeh, Siew Hock Ow, " Purposing a New Software Cost Estimation Model Based on Artificial Neural Networks", IEEE,2010.
- [5] Roger S. Pressman, " Software Engineering: A Partitioner's Approach", McGraw Hill, edition 5th. 2001.
- [6] Farhad SOLEIMANIAN GHAREHCHOPOGH, "Neural Network Application in Software Cost Estimation: A Case Study", IEEE, 2011.
- [7] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. k. Clark, B. Steece, A. W. Brown, S. Chulani and C. Abts, "Software Cost Estimation with COCOMO II", Prentice Hall, 2000.
- [8] G. Witting and G. Finnie, "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort," Journal of Information Systems, vol. 1, no.2, 1994, pp. 87-94.
- [9] N. Karunanithi, D. Whitely, Y. K. Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software Engineering, vol. 9, no.4, 2011, pp. 53-59.
- [10] B. Boehm, C. Abts, and S. Chulani, "Software Development Cost Estimation Approaches – A Survey," University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000- 505, 2000.
- [11] Tim Menzies, " COCOMO NASA 2/Software Cost Estimation", Promise Software Engineering Repository (<http://promise.site.uottawa.ca/SERepository/datasets-page.html>), 3 April 2006.

Author Profile



ANUJ KHUTTAN, He is pursuing M.tech, in Galgotias University Greater Noida, Delhi NCR, He is completed B.tech in Electronics And Communication from Guru Gobind Singh Indraprastha University, New Delhi, Area of interest is Software Engineering



Archana Singh, She is pursuing M.tech, in Galgotias University Greater Noida, Delhi NCR, She is completed M.Sc in Information Technology from Punjab Technical University, Jalundhar, Area of interest is Software engineering and Data Mining.



Ashwini kumar, He is Assistant professor Dept. of Computer Science at Galgotias University Greater Noida, Delhi NCR, He is completed Area of interest is computational intelligence and simulations, computational neuroscience, soft computing