Software Development Cost and Time Forecasting Using a High Performance Artificial Neural Network Model

Iman Attarzadeh and Siew Hock Ow

Department of Software Engineering
Faculty of Computer Science & Information Technology
University of Malaya, 50603 Kuala Lumpur, Malaysia
attarzadeh@siswa.um.edu.my, show@um.edu.my

Abstract. Nowadays, mature software companies are more interested to have a precise estimation of software metrics such as project time, cost, quality, and risk at the early stages of software development process. The ability to precisely estimate project time and costs by project managers is one of the essential tasks in software development activities, and it named software effort estimation. The estimated effort at the early stage of project development process is uncertain, vague, and often the least accurate. It is because that very little information is available at the beginning stage of project. Therefore, a reliable and precise effort estimation model is an ongoing challenge for project managers and software engineers. This research work proposes a novel soft computing model incorporating Constructive Cost Model (COCOMO) to improve the precision of software time and cost estimation. The proposed artificial neural network model has good generalisation, adaption capability, and it can be interpreted and validated by software engineers. The experimental results show that applying the desirable features of artificial neural networks on the algorithmic estimation model improves the accuracy of time and cost estimation and estimated effort can be very close to the actual effort.

Keywords: Software Engineering, Software Project Management, Software Cost Estimation Models, COCOMO Model, Soft Computing Techniques, Artificial Neural Networks.

1 Introduction

Accurate and consistent software development effort prediction in the early stage of development process is one of the critical tasks in software project management. Project managers use effort estimation to make on-time and better managerial decisions during project development life cycle and especially for determination of project details, allocation of project resources, project tasks, schedule controlling, and process monitoring. In software development process, the effort directly related to software schedule, cost and manpower factors that are critical and important for any project. The software development effort estimation is counted as a very complex process because of essential project factors such as development environments, platform factors, human factors, product factors, customers' needs, and finally the

R. Chen (Ed.): ICICIS 2011, Part I, CCIS 134, pp. 18–26, 2011.

difficulty of managing such large projects. During last decades, the governments and many mature organisations invest on software development to achieve their purpose.

Therefore, the accurate estimation of project time, cost, and staffing are needed to effectively plan, monitor, control and assess software development companies and project managers. The effort estimation in software engineering is based on two large methods: algorithmic methods and non-algorithmic methods. Algorithmic methods carry a mathematical formula that is inferred from regression model of historical data and project attributes. Constructive Cost Model (COCOMO) [1, 2] and Function Points [3] (FP) are two well-known methods of this category. Non-algorithmic methods [4, 5, 6], usually, are based on heretical projects information and comparing new project activities to past projects, then make estimation on the new project tasks. Expert judgment and analogy-based estimation [5] are two samples of non-algorithmic methods. This research work intends to use the soft computing approach, artificial neural networks, to propose a novel software effort estimation model incorporating constructive cost model to improve the precision of software effort estimation.

2 Related Work

Wittig and his colleagues proposed a simple neural network for software cost estimation. The purpose of that research was to examine the performance of back-propagation training algorithm. First, they used a metric model, ($SPQR/20^2$), to generate adequate data for the experiment. Second, they used a set of actual experiments from developed past software. In both methods, Function Points (FPs) method used as the measurement method of the input parameter, Size, and the development hours used as the unit of system output, Effort [7]. The experiments results in their research work show the ability of neural networks to make better estimation. Samson et al. used another mathematical model, Cerebellar Model Arithmetic Computer (CMAC). Then applied proposed neural networks architecture on the CMAC to make effort estimation based on software code size. The CMAC model proposed by Albus [8] and it is an approximation perceptron function. The established model based on neural networks was trained on COCOMO dataset in order to estimate software development effort from size of software. Also, they used linear regression techniques in same manner to compare the acquired data. The results of the proposed prediction model performed better than linear regression on the same data set. In other research, Boetticher used more than 33,000 different experiments data, collected from separate software companies, to examine the proposed neural network [9].

He used different software metrics such as size, complexity, objects, and vocabulary to estimate software effort using neural networks. Boetticher in another research used bottom-up approach to apply the past experiments on the two separate neural networks. The bottom-up approach uses data collected from software products rather than software projects. In that model, the Source Lines of Code (SLOC) metric used as the input of neural network model to estimate the project effort. Karunanitthi et al., also, proposed a cost estimation model based on artificial neural networks [10]. The established neural network was able to generalise from trained datasets. The

model results show better effort estimation results than other compared models. Srinivasan et al. applying machine leaning approaches based on neural networks and regression trees to algorithmic cost estimation models [11].

They reported that the main advantages of learning systems such as adaptable and nonparametric. However, they did not discuss on the used dataset and how divided it for training and validation process. In another research, Khoshgoftaar et al. considering a case study on the real time estimation software. They used a primary neural network model to establish a real time cost estimation model [12, 13]. But the presented validation process of their model is not adequate. Shepperd and Schofield presented a software estimation model based on analogy. This research was one the basic work on that time [14]. They used the information of past projects to make estimation for new projects. The proposed method was a heuristic method, so they could not make a good evaluation of the performance of their proposed model. Jorgenson examined many parameters in cost estimation based on expert judgment. His research established some essential parameters that affects on software development process [15].

3 The Proposed Artificial Neural Network Based on COCOMO Model

The constructive cost model, COCOMO, is a mathematical and regression-based effort estimation model that was proposed and establishes by Barry Boehm in 1981 [1, 2]. The calculation of effort based on the COCOMO post architecture model is given as:

$$Effort = A \times [Size]^{B} \times \prod_{i=1}^{17} Effort\ Multiplier_{i} \tag{1}$$

$$where\ B = 1.01 + 0.01 \times \sum_{j=1}^{5} Scale\ Factor_{j}$$

In the equation "1":

A: Multiplicative Constant
Size: Size of the software project measured in terms of KSLOC or FP.

The proposed artificial neural network architecture is customised to adopt the widely used algorithmic model, COCOMO model. The input and output parameters of COCOMO model are categorised as follow [1, 2]:

- Size parameter – Input parameter: software size based on thousands source lines of code.
- Five Scale Factors (SF) – Input parameter: scale factors are based software productivity variation and project's activities.
- Seventeen Effort Multipliers (EM) - Input parameter: effort multipliers are based on project attributes, product attributes, personnel attributes, and hardware attributes.
- Effort – Output parameter: the calculated effort is based on person-month (PM) unit.

In the new proposed model, twenty five input parameters are defined that corresponds to number of SFs and EMs as well as two bias values. Therefore, the input layer in the new artificial neural network includes 25 nodes. However, in order to customise the COCOMO model in the new ANN model, a specific ANN layer as a hidden layer and a widely used activation function, Sigmoid, with some data pre-processing for input and hidden layer are considered. The ANN architecture and configuration shows in Figure 1.

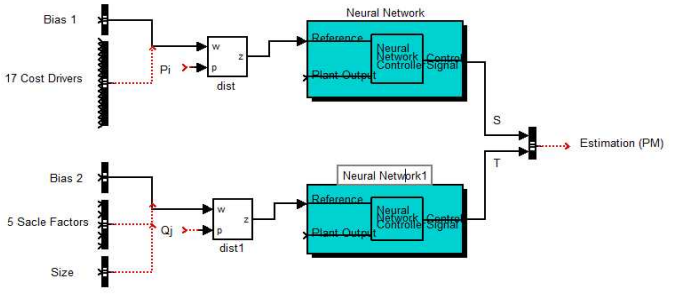

Fig. 1. The proposed COCOMO II model based on artificial neural networks

In the proposed ANN, the scale factors and effort multipliers values are pre-processed to $\log(SF_i)$ and $\log(EM_i)$, also the input size of the product in KSLOC is considered as one of the initial weights for scale factors. The Sigmoid function is considered as the activation function of hidden layer. It defined by $f(x) = \frac{1}{1+e^{-x}}$. The assigned weights of nodes in the input layer to hidden layer are defined by P_i for Bias1 and each input $\log(EM_i)$ for $1 \leq i \leq 17$. On the other hand, the assigned weights of each SF_j in the input layer to hidden layer are defined as $q_j + \log(size)$ for $1 \leq j \leq 5$ by Bias2. 'S' and 'T' are the final weights of nodes from the hidden layer to the output layer as shown in Figure 1. 'S' and 'T' are the values of the relevant nodes in the hidden layer and in the output layer the identity function uses them to generate the final effort value.

One of the additional contributions in the new ANN architecture is using pre-processed Log(Size) to the weight q_j of SF's input for adjusting the weights q_j. Another major contribution of the new ANN architecture compared to previous models is the training approach of the new artificial neural network. In order to customise the COCOMO model in the new ANN architecture, the initial values of weights 'S' and 'T' are adjusted to the offset of the nodes values in the hidden layer. The network inputs initialise to a random data from the data set at the beginning of training. The network output, effort estimation, would be inferred from the equation of COCOMO model, in "1", by using the initial values of Bias1 as Log (A) and Bias2 as 1.01. The network weights are initialised as $p_i = 1$ for $1 \leq i \leq 17$ and $q_j = 1$ for $1 \leq j \leq 5$. The nodes values in the hidden layer generated by propagating the values of input nodes into the network as follow:

$$f(p_0\ Bias1 + \sum_1^{17} p_i * log(EM_i)) = sigmoid\ (Bias1 + \sum_1^{17} p_i * log(EM_i)) = \frac{A * \prod_{i=1}^{17} EM_i}{1 + A * \prod_{i=1}^{17} EM_i} = \alpha \tag{2}$$

$$f((q_0 + \log(size)) * Bias2 + \sum_{j=1}^{5}(q_j + \log(size))(SF_j) = sigmoid\ (\log(size) * (Bias2 + \sum_{j=1}^{5} SF_j) = \frac{Size^{1.01 + \sum_{j=1}^{5} SF_j}}{1 + Size^{1.01 + \sum_{j=1}^{5} SF_j}} = \beta \tag{3}$$

Then initialisation of weights 'S' and 'T' as follow:

$$S = \frac{\beta}{2\,(1-\alpha)(1-\beta)} \quad \text{and} \quad T = \frac{\alpha}{2\,(1-\alpha)(1-\beta)} \tag{4}$$

The network output calculated as:

$$PM = S * \alpha + T * \beta = \frac{\alpha\beta}{(1-\alpha)(1-\beta)} = A.\,Size^{1.01 + \sum_{j=1}^{5} SF_j} * \prod_{i=1}^{17} EM_i \tag{5}$$

4 Results and Discussion

Experiments were done by taking three different datasets as follow:

- First dataset, Dataset #1, the projects information from COCOMO dataset - COCOMO dataset involves 63 different projects.
- Second dataset, Dataset #2, the projects information from NASA projects - NASA dataset involves 93 different projects.
- Third dataset, Dataset #3, the artificial dataset that created based on the ANN – The artificial dataset involves 100 different projects information from the COCOMO and NASA datasets.

Therefore, in this research work has used 256 projects information from three different datasets, to evaluate the proposed artificial neural network estimation model. Finally, by aggregation of the obtained results from the ANN model, the accuracy of the proposed model compares to other estimation models.

4.1 Evaluation Method

The two most widely accepted evaluation methods are used for model evaluation, which are as follows:

- Mean Magnitude of Relative Error (MMRE)
- Pred(L): probability of a project having a relative error of less than or equal to L that called Pred(L) or prediction at level L. The common value of L is 25%.

The first evaluation method, Magnitude of Relative Error (MRE) is defined as follows:

$$MRE_i = \frac{|\text{Actual Effort}_i - \text{Predicted Effort}_i|}{\text{Actual Effort}_i} \tag{6}$$

The MRE is calculated for all predicted effort in the datasets, for all observations i. Another related evaluation method is the Mean MRE (MMRE) that can be achieved through the aggregation of MRE over multiple observations (N) as follows:

$$\mathrm{MMRE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}}^{\mathrm{N}} \mathrm{MRE_i} \tag{7}$$

Therefore, usually, the aggregation of the measures used to overcome the problem. In fact, the aggregate measure less sensitive to large values. In the second evaluation method, a complementary criterion is the prediction or estimation at level L, Pred(L) = K/N, where k carries the number of observations of MRE (or MER) that less than or equal to L, and N equals total number of observations. Thus, Pred(25%), for example, means the percentage of projects which were estimated with a MRE (or MER) less or equal than 0.25. The proposed artificial neural network estimation model was trained and evaluated based on the three described datasets in the previous sections. At the first attempt, the COCOMO dataset, 63 projects, applied to the new ANN model. The system results, effort estimation, recorded step by step in a table for future comparisons.

The MRE is calculated based on the estimated effort and corresponding actual effort form the dataset. Finally, the aggregation of the results, MMRE, computes to avoid any sensitivity to large values. Then the Pred(25%) also calculated as the second evaluation method. Same approach applied for NASA dataset, Dataset #2, and Artificial dataset, Dataset #3. All the 256 projects used to the evaluation of the proposed ANN estimation model. The comparison of the obtained results from Dataset #1, #2, and #3 that applied on the new artificial neural network cost estimation model and COCOMO model shows more accuracy in case of effort estimation by the new ANN estimation model. Table 1 shows, the used datasets on one side and the other side, the MMRE and Pred(25%) corresponding values to the models. In fact, all datasets first applied to the proposed ANN model then applied to the COOCMO model, widely used cost estimation model. The final results shown in Table 1 as follows.

Table 1. Comparison between performance of the new model and COCOMO II

Dataset	Model	Evaluation	
		MMRE	Pred (25%)
Dataset #1	COCOMO II	0.581863191	30%
	Proposed Model	0.413568265	40%
Dataset #2	COCOMO II	0.481729632	40%
	Proposed Model	0.468142057	50%
Dataset #3	COCOMO II	0.526316925	40%
	Proposed Model	0.453826571	40%
Mean	COCOMO II	0.529969916	36.6%
	Proposed Model	0.445178964	43.3%

In this research work three different datasets has applied to the proposed artificial neural network cost estimation model and the COCOMO model. For each set of data, dataset, the MMRE and Pred(25%) measures were calculated for models evaluation. The MMRE and Pred(25%) values for each dataset to consider the performance of

each dataset separately as in shown in Table 1. The final results of 256 applied projects shows that the MMRE for the proposed ANN estimation model is 0.445178964 and the Pred(25%) values equals 43.3% when compared to COCOMO model with MMRE = 0.529969916 and Pred(25%) = 36.6%.

Obviously, the results show that the MMRE of proposed ANN model is less than the MMRE of COCOMO model. As it state above, if the value of MMRE is closed to 0, it means the amount of occurred error, the difference of actual effort and estimated effort, is very low. In the other words, it means the accuracy of estimated effort in the ANN estimation model is better than the COCOMO model. When the value of Pred(25%) in the ANN estimation model, 43.3%, compared to the corresponding value in the COCOMO model, 36.6%, it shows that the proposed model provides accurate results than the COCOMO model. If the value of Pred(25%) measure is closed to 100%, it means most of estimated results are closed to the actual results. Therefore, the accuracy of the ANN estimation model is obviously better than the COCOMO model. According to the stated analysis based on the two evaluation methods, MMRE and Pred, the artificial neural network cost estimation model shows better results in case of accuracy of the estimated results. That is one of the good achievements of this research work. Table 2 shows the comparison between the proposed ANN model and the COCOMO model in case of percentage of accuracy in the two examined model.

Table 2. Accuracy of the proposed model

Model	Evaluation	MMRE
Proposed Model vs. COCOMO II	COCOMO II	0.529969916
	Proposed Model	0.445178964
	Improvement %	9.28%

When the percentage of improvement in the ANN estimation model and the COCOMO model is calculated, the result shows 9.28% improvement in case of the accuracy of the estimation with the propose model. In summary, the experimental results from the two effort estimation models confirm the capabilities and better performance of the proposed ANN model in case of accuracy of the estimation when compared to the COCOMO model. Most of the selected data with the proposed ANN effort estimation model resulted in a more precise estimations when compared to the COCOMO model.

5 Conclusion

In software engineering and especially in software project management providing the accurate and reliable software attributes estimation in the early stages of the software development process is one the essential, critical, and crucial issues. It always has been the center of attention for software engineers and project managers as well as mature software companies and organisations. Software development attributes usually have properties of vagueness and uncertainty when the human judgment used

to measure them. Using the ability of artificial neural networks to provide accurate estimations can overcome the characteristics of vagueness and uncertainty that exists in software development attributes. However, utilising adaptive neural network architecture plays a key role in coming up with reliable and accurate effort estimation model. Appling soft computing techniques, e.g. artificial neural networks, can be a considerable attempt in the area of software project management and estimation. One of the main objectives of this research work is to use artificial neural network, as an applicable technique, for software effort estimates that performs better and accurate estimation than other techniques, e.g. COCOMO model, on a given datasets. In this research work a novel artificial neural network presented to handle the uncertainty and imprecision of software effort estimation. The proposed model has shown applying artificial neural network as an applicable technique on the algorithmic cost estimation models accurate estimation is achievable. The ANN effort estimation model has shown better and accurate software effort estimates in view of two evaluation methods, the MMRE and Pred (0.25), as compared to the COCOMO model. The percentage of improvement in the proposed model, 9.28%, demonstrate that utilising artificial neural networks in software effort estimation can be an applicable and feasible approach to address the problem of vagueness and uncertainty exists in software development attributes. Furthermore, the proposed ANN effort estimation model presents better estimation accuracy as compared to the algorithmic COCOMO model. The applying other soft computing techniques for other software engineering weakness can also is considered in the future.

References

[1] Boehm, B.: Software Engineering Economics. Prentice-Hall, Englewood Cliffs (1981)
[2] Boehm, B., Abts, C., Chulani, S.: Software Development Cost Estimation Approaches – A Survey, University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000-505 (2000)
[3] Putnam, L.H.: A General Empirical Solution to the Macro Software Sizing and Estimating Problem. IEEE Transactions on Software Engineering 4(4), 345–361 (1978)
[4] Srinivasan, K., Fisher, D.: Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering 21(2), 123–134 (1995)
[5] Molokken, K., Jorgensen, M.: A review of software surveys on software effort estimation. In: IEEE International Symposium on Empirical Software Engineering, ISESE, pp. 223–230 (2003)
[6] Huang, S., Chiu, N.: Applying fuzzy neural network to estimate software development effort. Applied Intelligence Journal 30(2), 73–83 (2009)
[7] Witting, G., Finnie, G.: Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort. Journal of Information Systems 1(2), 87–94 (1994)
[8] Samson, B.: Software cost estimation using an Albus perceptron. Journal of Information and Software 4(2), 55–60 (1997)
[9] Boetticher, G.D.: An assessment of metric contribution in the construction of a neural network-based effort estimator. In: Proceedings of Second International Workshop on Soft Computing Applied to Software Engineering, pp. 234–245 (2001)
[10] Karunanitthi, N., Whitely, D., Malaiya, Y.K.: Using Neural Networks in Reliability Prediction. IEEE Software Engineering 9(4), 53–59 (1992)

[11] Srinivasan, K., Fisher, D.: Machine learning approaches to estimating software development effort. IEEE Transaction on Software Engineering 21(2), 126–137 (1995)
[12] Khoshgoftar, T.M., Allen, E.B., Xu, Z.: Predicting testability of program modules using a neural network. In: Proceeding of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp. 57–62 (2000)
[13] Khoshgoftar, T.M., Seliya, N.: Fault prediction modeling for software quality estimation: comparing commonly used techniques. Journal of Empirical Software Engineering 8(3), 255–283 (2003)
[14] Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering 23(11), 736–743 (1997)
[15] Jorgenson, M.: A review of studies on expert estimation of software development effort. Journal of Systems and Software 70(4), 37–60 (2004)