Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2009, Article ID 829725, 8 pages
doi:10.1155/2009/829725

Research Article

Improving Effort Estimation by Voting Software Estimation Models

Luiz Fernando Capretz[1] and Venus Marza[2]

[1] *Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada N6A 5B9*
[2] *Faculty of Technology and Engineering, Islamic Azad University South Tehran Branch, 1471715363 Tehran, Iran*

Correspondence should be addressed to Venus Marza, venus.marza@gmail.com

Received 9 February 2009; Accepted 24 June 2009

Recommended by Hossein Saiedian

Estimating software development effort is an important task in the management of large software projects. The task is challenging, and it has been receiving the attentions of researchers ever since software was developed for commercial purpose. A number of estimation models exist for effort prediction. However, there is a need for novel models to obtain more accurate estimations. The primary purpose of this study is to propose a precise method of estimation by selecting the most popular models in order to improve accuracy. Consequently, the final results are very precise and reliable when they are applied to a real dataset in a software project. Empirical validation of this approach uses the International Software Benchmarking Standards Group (ISBSG) Data Repository Version 10 to demonstrate the improvement in software estimation accuracy.

1. Introduction

Estimating software development effort remains a complex problem that attracts considerable research attention. Improving the estimation techniques that are currently available to project managers would facilitate increased control of time and money in software development. Furthermore, any improvement in the accuracy of predicting the development effort can significantly reduce the costs from errors, such as estimating inaccurately, misleading tendering bids, and disabling the monitoring progress.

In order to manage the dynamic development environments, many researchers have applied the artificial intelligence- (AI-) based frameworks, such as neural network and case-base reasoning models [1]. Among these AI-based prediction models, neural networks (NNs) are recognized for their ability to produce reasonably accurate predictions in situations where complex relationships between inputs and outputs exist and where the input data is distorted by high noise levels [2]. Existing research describes several parametric models such as COCOMO [3] and SLIM [4] as well as various nonparametric models such as Fuzzy Logic (FL), the Neural Network (NN), and various combinations of these models, such as the Neurofuzzy method; however, our proposed voting framework is a unique model that extends current works.

In the last two decades, Artificial Neural Network (ANN) has been used for predictions in diverse applications; it is the most commonly used learning-oriented approach for estimating software development effort. Huang et al. [5] propose a model combining fuzzy logic and neural network. The main benefit of this model is its good interpretability by using the fuzzy rules. Another great advantage of this research is that they could put together expert knowledge (fuzzy rules), project data, and traditional algorithmic model into one general framework that may have a wide range of applicability in software cost estimation.

Saliu et al. [6] present a Fuzzy Logic Model (FLM) based upon triangular membership functions. Results showed that the FLM was slightly better than COCOMO equations. In addition, they reported promising experimental summary results in spite of the little background knowledge of the rule base and training data. It does signify that there are potentials for improvements when the framework is deployed in practice, since experienced experts could augment the dataset value with their knowledge.

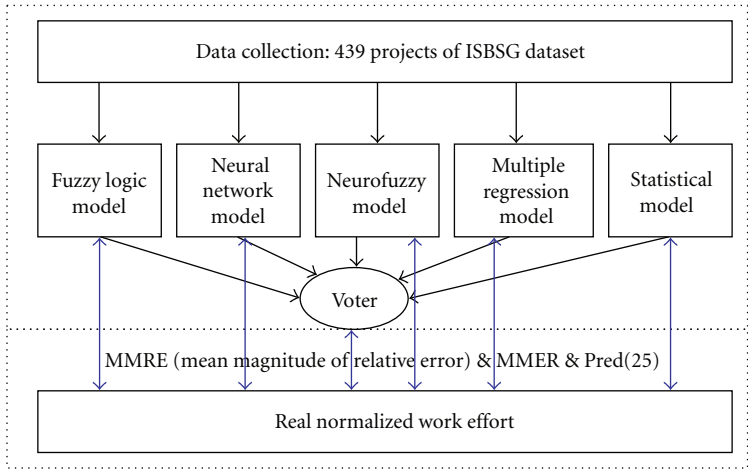

Figure 1: Research methodology.

Martin et al. [7] explained that regression modeling is one of the most widely used statistical modeling technique for fitting a response (dependent) variable as a function of predicator (Independent) variables. Development effort constitutes a dependent variable while all the factors that influence on effort are independent variables. Jiang and Naudé [8] statistically examine the factors that influence development effort. Their paper proposes a parsimonious parametric model for prediction of effort which is both simple and accurate than previous models.

Despite all research done in the field of software estimation, improving accuracy remains a daunting task. This work tries to get the best of current techniques using a voting approach. The remainder of this paper is structured into four sections. In Section 2, we briefly introduce the proposed model and prepare the dataset. Section 3 presents the various parts of the proposed model. The experimental results are examined in Section 4, and finally, Section 5 discusses conclusions and points out to future research.

2. Research Method

The aim of this study is to build and evaluate an estimation model that improves accuracy as a result of voting among traditional popular models.

2.1. Proposed Model Description. All of the estimation approaches that we will describe in detail are valuable as software development effort estimation models. However, by comparing and voting among them, we can achieve greater accuracy estimation. A schematic diagram of our proposed model is depicted in Figure 1.

Since the Neurofuzzy model has a lower MMRE than other models, we can utilize it as the basic model with which all other models are compared. For each model, we first calculate the following factor:

$$X(k) = \frac{|\mathrm{Out}_{\mathrm{Neuro}} - \mathrm{Out}_k|}{\mathrm{Out}_{\mathrm{Neuro}}} * 100, \quad (1)$$

Table 1: Summary of our model variable.

Variables	Scale	Description
Effort (output)	Ratio	Normalized work effort
AFP (Input)	Ratio	Adjusted function points
MTS (Input)	Ratio	Max team size
DT (Input)	Nominal	Development type
DP (Input)	Nominal	Development platform
LT (Input)	Nominal	Language type
DTech (Input)	Nominal	Development techniques

where K represents the various models mentioned in this article. In the case, where there is less than 10% error, X is lower than 10, we can use that model in calculating output and using it in average formula. Each output is computed by averaging the output models which have lower than 10% Error.

2.2. Data Preparation. As previously mentioned, the empirical validation of this study uses International Software Benchmarking Standards Group (ISBSG) Data Repository, Version 10. Specifically, the dataset contains information on 4106 projects, two-thirds of which were developed between the years 2000 and 2007. Projects with any missing values were discarded from further use in the model building, because they may potentially affect the effort estimates. After preprocessing the data, the ISBSG contained 439 software projects with six effort drivers for constructing the estimation models. These six effort drivers are divided and classified as having either ratio or nominal scales. Those with ratio scales include Adjusted Function Points (AFPs) and Max Team Size (MTS), whereas Development Type (DT), Development Platform (DP), Language Type (LT), and Development Technique (DTech) all have nominal scales, as shown in Table 1.

In the case of nominal scale variables, our model relies on the assistance of expert knowledge. A brief description of these variables is as follows.

Table 2: Coefficients of nominal scale variables.

Nominal scale variables		Coefficients
Language type	2GL	0
	3GL	−0.40
	4GL	−0.85
	ApG	−0.71
Development platform	Mid-range	−0.12
	Multiplatform	−0.15
	PC	−0.46
	Mainframe	0
Development type	New-development	0.29
	Enhancement	0
	Redevelopment	0.56
Development technique	RAD	−0.23

(i) *Language Type.* Language type contributes significantly to the software development effort. Its assistance is evident from the regression coefficients of 3GL (−0.40), 4GL (−0.85), and ApG (−0.71). These values are relative to the value (0) for 2GL, which is the default development language. Thus, fourth-generation languages (4GLs) are more useful for reducing development effort than third-generation languages (3GLs).

(ii) *Development Platform.* The computer platform has been considered an important cost driver in estimating software effort. Mainframe computers need to serve numerous users and process large amount of data quickly. Hence, software development for mainframe computers requires considerable effort, while the development for personal computers requires a much smaller effort. Moreover, in comparison to single platform development, multiplatform development requires much more effort, especially as it involves repeated work on the building and testing of all platforms. Mid-range computers, designed to be hosts in a multiuser environment, are of relatively smaller scale and thus require less effort than mainframe computers.

(iii) *Development Type*. While new development involves building software from scratch, software enhancement simply adds, changes, or deletes the functionality of previously existing systems to adapt to evolving business requirements. In some cases, software requires frequent and major changes involving substantial costs; so it is preferable to develop it completely. However, for software that requires relatively few and simple changes, enhancement is more suitable and inexpensive than complete redevelopment.

(iv) *Development Techniques.* Rapid Application Development (RAD) [9] is a software development technique that focuses on building applications in a very short period of time. The key objective of RAD is the fast development of high-quality systems with low costs. Projects using RAD achieved significantly higher productivity than those using traditional development methods [10]. Thus, by using RAD, we can significantly reduce the development effort.

Table 2 illustrates the coefficients of nominal scale variables, and Table 3 shows the coefficients of ratio scale variables.

Table 3: Coefficients of ratio scale variables.

Nominal scale variables	Coefficients
Log (size)	0.56
Log (Team size)	0.68

Table 4: Different kinds of development types.

Development type	Assigned number	Number of cases
Enhancement	0	188
New-development	0.29	238
Redevelopment	0.56	13

Table 5: Different kinds of language types.

Language type	Assigned number	Number of cases
2GL	0	0
3GL	−0.40	209
4GL	−0.85	199
ApG	−0.71	31

Table 6: Different kinds of development platforms.

Development platform	Assigned number	Number of cases
Main-frame	0	207
Mid-range	−0.12	110
Multiplatform	−0.15	5
PC	−0.46	117

Table 7: Different kinds of methodology acquirement.

Development technology	Assigned number	Number of cases
RAD	−0.23	59
Nine other technologies (waterfall, prototyping, data modeling, process, modeling, JAD (joint application development), regression testing, OO (object oriented analysis & design), business area modeling)	0	380

3. Brief Description of Incorporated Models in Voting

The models considered in this work are based on Statistical Models, Fuzzy Logic Models, Neural Network Models, Neurofuzzy Models, and Multiple Regression Models.

3.1. Statistical Model. For this type of model, the effort drivers are divided into two parts; first, there are drivers with a direct effect on effort, such as AFP, MTS, and DT, where applying the drivers will increase the development effort. The second type of driver has an inverse effect on effort and therefore a negative coefficient, including DTech, LT, and DP. For nominal scale factors, we assign a coefficient to each of them, which are obtained from statistical models and are illustrated in Tables 4, 5, 6, and 7.

Table 8: The two input variables for effort estimation.

Direct effect variable	$X1$	$\text{Exp}(4.24+0.56*\log(\text{Size})+0.68*\log(\text{Team Size})+\gamma k\Phi(\text{Development Type } k))\ k=1,2,3$
Inverse effect variable	$X2$	$\text{Exp}(\alpha_i\Phi(\text{Language Type}_i)+\beta j\Phi(\text{Development Platform } j)+\Phi(\text{Development Technology}))$ $i=1,2,3,4;\ j=1,2,3,4$

The purpose of the Statistical Model is to assign effort as the dependent variable and to make the other variables the predicators. Preliminary analysis indicated that multiple colinearity within the data was not a problem. Since regression analysis based on AIC (Akaike's Information Criterion) tends to overestimate the number of parameters when the sample size is large [11], the results produced by AIC produce inaccurate measures and thus should not be the sole source of analysis. Rather, the use of AIC should be combined with other statistical criterion, such as the Analysis of Variance (ANOVA). By considering both procedures, it is evident that the two methods produce similarly significant factors; this model is described as follows:

$$\begin{aligned}\text{Log(Effort)} &= 4.24+0.56*\log(\text{Size})+0.68*\log(\text{Team Size})\\ &\quad+\alpha_i\Phi\left(\text{Language Type}_i\right)\\ &\quad+\beta j\Phi(\text{Development Platform } j)\\ &\quad+\gamma k\Phi(\text{Development Type } k)\\ &\quad-0.23\Phi(\text{RAD}),\\ &\qquad i=1,2,3,4;\ j=1,2,3,4;\ k=1,2,3.\end{aligned} \tag{2}$$

In this case, the function Φ is the indicator function, with binary values of 1 or 0. Specifically, a value of 1 means that the relevant development technique in the parentheses is used; otherwise the value is 0. By applying this equation to the dataset, the results evident in Table 10 are obtained.

3.2. Fuzzy Logic Model. Fuzzy logic enhances the user's ability to interpret the model, allowing the user to view, evaluate, criticize, and possibly adapt the model. Prediction can be explained through a series of rules [12]. After analyzing the fuzzy logic model, experts can check the model to avoid the adverse effects of unusual data, thereby increasing its robustness [13]. Additionally, fuzzy logic models can be easily understood in comparison to regression models and the neural network, thus making it an effective communication tool for management [14]. In comparison to fuzzy logic, case-based reasoning is similarly easy to interpret, but it requires a high volume of data [15].

The number of fuzzy rules for six input variables and three membership functions is calculated by 3^6, which equals 729. As a result, writing these rules is an arduous task; so based on the statistical model we use two input variables, the direct effect and the inverse effect, which are demonstrated in Table 8.

The fuzzy rules for these variables are based on the correlation (r) between pairs of variables. Correlation, the degree to which two sets of data are related, varies from -1.0 to 1.0. The Correlation Coefficient for the input variables is calculated from the following equation:

$$r=\frac{n[\sum(X_i\cdot Y_i)]-(\sum X_i)(\sum Y_i)}{\sqrt{\left[n(\sum X_i^2)-(\sum X_i)^2\right]\left[n(\sum Y_i^2)-(\sum Y_i)^2\right]}}. \tag{3}$$

Table 9: Correlation between variables.

r	$X1$	$X2$	Effort
$X1$	1	0.5	0.8
$X2$	—	1	0.4
Effort	—	—	1

As a result, the correlation of the variables, based on (1), is depicted in Table 9.

An acceptable correlation should have an absolute value higher than 0.5. Since the correlation of variables under the principal diagonal of Table 9 did not meet this criterion and thus was redundant data, we left it blank. The fuzzy inference process uses the Mamdani Approach for evaluating each variable complexity degree when linguistic terms, fuzzy sets, and fuzzy rules are defined. Specifically, we apply the minimum method to evaluate the "and" operation, and consequently, we obtain one number that represents the antecedent result for that rule. The antecedent result, as a single number, creates the consequence using the minimum implication method. Overall, each rule is applied in the implication process and produces one result. The aggregation using the maximum method is processed to combine all consequences from all the rules and produces one fuzzy set as the output. Finally, the output fuzzy set is defuzzified to a crisp single number using the centroid calculation method [16]. This Two-Input-One-Output fuzzy logic system for Effort is depicted in Figure 2. Moreover, the results of this model are shown in Table 10.

3.3. Neural Network Model. Our neural network model uses an RBF network, which is easier to train than an MLP network. The RBF network is structured similarly to the MLP in that it is a multilayer, feed-forward network. However, unlike the MLP, the hidden units in the RBF are different from the units in the input and output layers. Specifically, they contain the RBF, a statistical transformation based on a Gaussian distribution from which the neural network's name is derived [17]. Since the data of our variables differs significantly, first, we normalized the data and then divided it into two categories: 75% or 329 pieces are used for training and 25% or 110 are used for testing. The trajectory of the training phase is depicted in Figure 3. In particular, we used the Generalized Regression Neural Network Model and applied it to the dataset; the results are shown in Table 10.

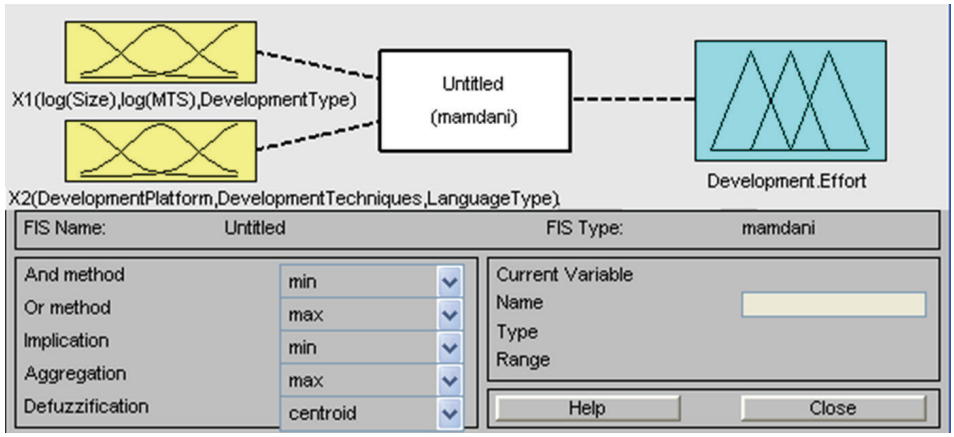

Figure 2: The fuzzy logic system for estimating effort.

Table 10: Comparison of models using MMRE, MMER, and Pred(25).

	Multiple regression model	Neural network model	Fuzzy logic model	Neurofuzzy model	Statistical model	Voting method
MMRE	0.649659945	0.739787708	0.74033832	**0.282264458**	1.115479582	0.30873679
MMER	0.207021353	0.23582904	2.531378977	0.201317968	0.367909284	**0.125078979**
Pred(25)	0.70615034	0.61731207	0.03416856	**0.87471526**	0.4829157	**0.87471526**

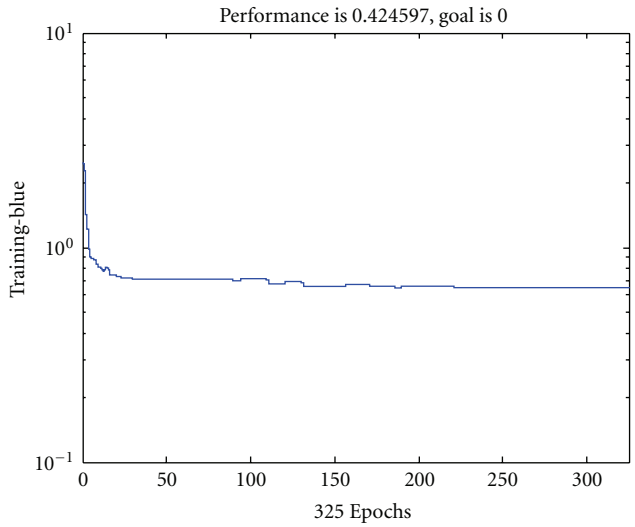

Figure 3: Progress of training phase.

3.4. Neurofuzzy Model. Among several current approaches, the Neurofuzzy technique is a promising strategy that solves several problems involving evaluation accuracy [18, 19]. This approach is a suitable due to its built-in learning capacity, its robustness in the case of uncertain input, and its ability to utilize a variety of information from multiple sources.

In our method, the ANFIS (Adaptive Neurofuzzy Inference System) Architecture is used to implement the neurofuzzy approach, which is functionally equivalent to a Sugeno-type fuzzy rule base. Furthermore, it is also similar to a radial-basis function network. First, each neuron in Layer 1 adapts to a parametric activation function and produces the appropriate grade of membership for the given inputs to satisfy the membership functions. The Gaussian Curve is used for the membership function, which is given by

$$f(x) = \exp\left(\frac{-0.5\ (x-c)^2}{\delta^2}\right), \tag{4}$$

where c is the mean, and δ is the variance. In Layer 2, every node is a fixed node whose output is the product of all incoming signals and represents the firing strength of the ith rule. Subsequently, each node in Layer 3 calculates the ratio of the ith rule's firing strength relative to the sum of the firing strengths for all rules. The nodes of Layer 4 are adaptive nodes, and they produce a node output by the consequent parameters and normalized firing strength from Layer 3. Finally, in Layer 5, there is a fixed node that totals all incoming signals.

In order to find the best point for the epoch number, we calculated the RMS, which shown in (3):

$$\text{RMS} = \frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M} e_i^2, \tag{5}$$

where N is the size of inputs, and M is the number of outputs. The graph of the RMS, according to the number of epochs, is depicted in Figure 4. We deemed the value of 1000 Epochs suitable for producing appropriate results; the outcome of this approach is indicated in Table 10.

3.5. Multiple Regression Model. For this model, we also use two input variables, as shown in Table 8. A linear equation with two independent variables may be expressed as

$$y = b_0 + b_1 x_1 + b_2 x_2, \tag{6}$$

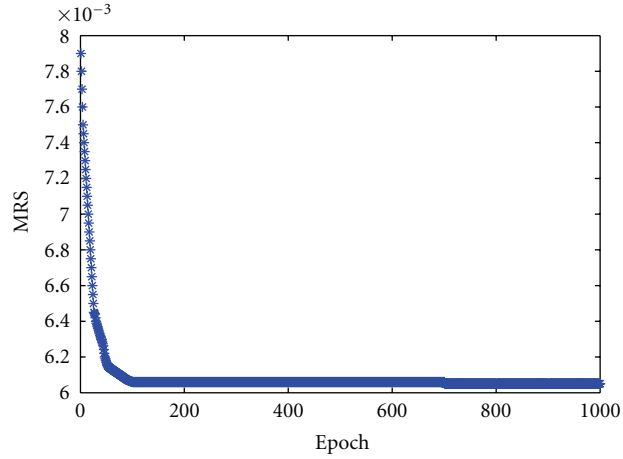

Figure 4: Optimum number of epochs.

where b_0, b_1, and b_2 are constants; x_1 and x_2 are the independent variables, and y is the dependent variable, which corresponds to the normalized work effort. For the multiple regression equation, the values b_0, b_1, and b_2 may be obtained by solving another system of linear equations:

$$\begin{aligned} \sum y &= nb_0 + b_1\left(\sum x_1\right) + b_2\left(\sum x_2\right), \\ \sum x_1 y &= b_0\left(\sum x_1\right) + b_1\left(\sum x_1^2\right) + b_2\left(\sum x_1 x_2\right), \\ \sum x_2 y &= b_0\left(\sum x_2\right) + b_1\left(\sum x_1 x_2\right) + b_2\left(\sum x_2^2\right). \end{aligned} \tag{7}$$

By solving this linear equations system, (4) gives the following linear regression equation. The results of applying this approach to our dataset are given in Table 10:

$$\text{Effort} = 0.44906 + 1.2235 * X_1 + 0.09046 * X_2. \tag{8}$$

4. Experimental Results

The following experiments were used to validate our approach.

4.1. Evaluation Criteria. We employ two common criteria for assessing and comparing the performance of cost estimation models: the Relative Error (RE) given by (9) or the Magnitude of Relative Error (MRE) given by (10), which are defined as follows:

$$\text{RE}_i = \frac{\text{Actual Effort}_i - \text{Predicted Effort}_i}{\text{Actual Effort}_i}, \tag{9}$$

$$\text{MRE}_i = \frac{|\text{Actual Effort}_i - \text{Predicted Effort}_i|}{\text{Actual Effort}_i}. \tag{10}$$

The RE and MRE values are calculated for each project, i, whose effort is subsequently predicted. For multiple projects, the number of which is denoted by N; we can also use the Mean Magnitude of Relative Error (MMRE) :

$$\begin{aligned} \text{MMRE}_i &= \frac{1}{N}\sum_{i=1}^{N}\frac{|\text{Actual Effort}_i - \text{Predicted Effort}_i|}{\text{Actual Effort}_i} \\ &= \frac{1}{N}\sum_{i=1}^{N}\text{MRE}_i. \end{aligned} \tag{11}$$

For our purposes, the MER seems preferable to MRE, especially since the MER measures the error relative to the estimate. As a result, the MER is defined as follows:

$$\text{MER}_i = \frac{|\text{Actual Effort}_i - \text{Predicted Effort}_i|}{\text{Predicted Effort}_i}. \tag{12}$$

As in the case of the MRE and the MMRE, the MER value is calculated for each observation, i, whose effort is then predicted. The aggregation of MER from multiple observations (N) can be achieved through the mean of MER (MMER) as

$$\text{MMER} = \frac{1}{N}\sum_{i=1}^{N}\text{MER}_i. \tag{13}$$

Another criterion that is commonly used is the prediction at level p:

$$\text{Pred}(p) = \frac{k}{N}, \tag{14}$$

where k is the number of projects in which the MRE is less than or equal to p. In this case, we used the Pred(25).

4.2. Evaluation of Proposed Model. We used the MMRE, MMER, and Pred(25) for comparing our method with each previously mentioned model. This comparison, based on 439 input data, is shown in Table 10.

The best result in each row bolded in Table 10. As it can be seen, our proposed method has a lower MMER among the other models, and its MMRE is slightly higher than the Neurofuzzy model which has the best MMRE. Also, both of them have the same Pred(25).

4.3. Discussions of Results. For a more accurate comparison of the models, we plot each MMER model against our proposed model. Figures 5, 6, 7, 8, 9 illustrate comparative diagrams.

These figures showed that most of the outputs in our proposed model had less than 20% error while the other models have not contained these situations.

5. Conclusion and Future Research

The major difference between our work and the previous methods involves the fact that we have selected the popular estimation models for achieving greater accuracy. Moreover, our model was validated with 439 cases of ISBSG Version 10. Overall, this research was focused on three ideas; first, software development effort estimates are the basis for project

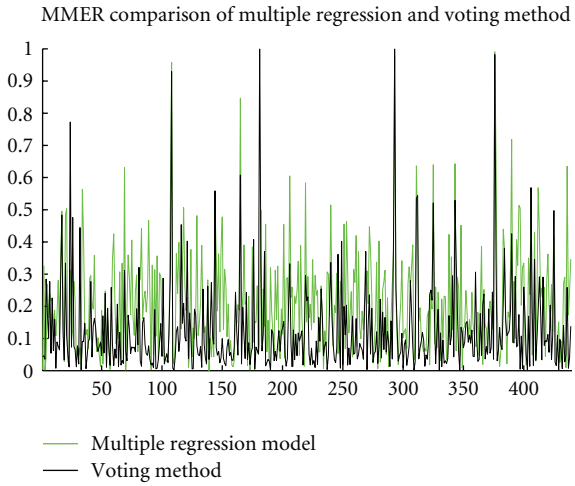

Figure 5: MMER comparison between voting method and multiple regression model.

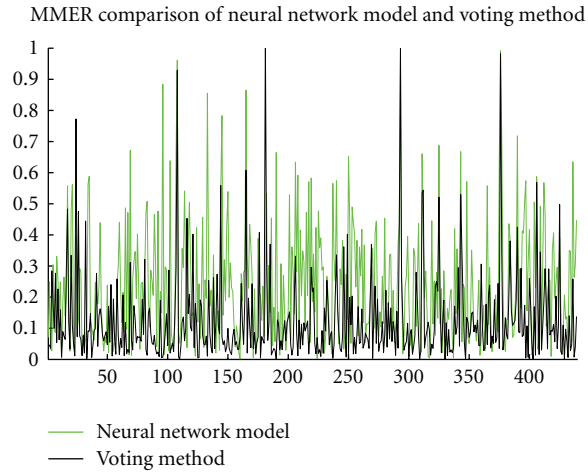

Figure 6: MMER comparison between voting method and neural network model.

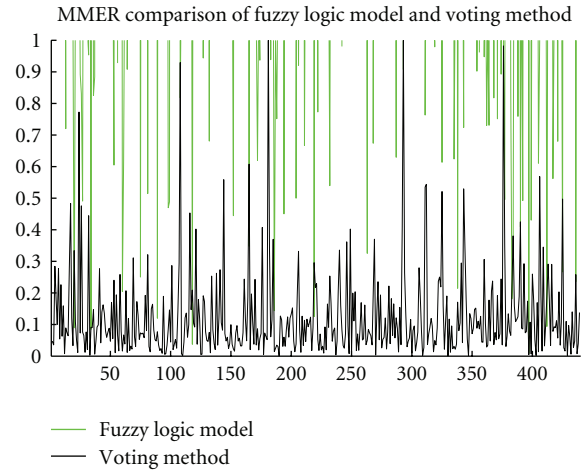

Figure 7: MMER comparison between voting method and fuzzy logic model.

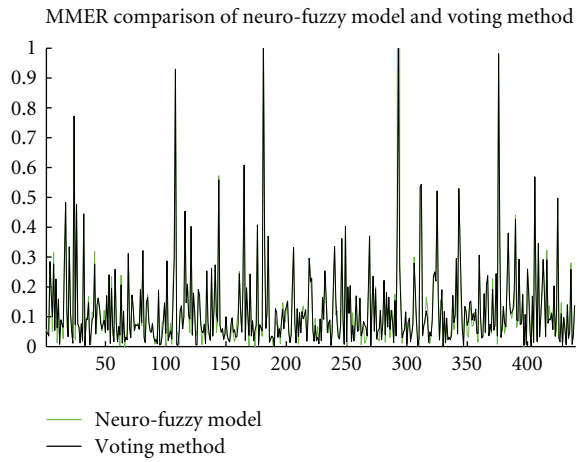

Figure 8: MMER comparison between voting method and neuro-fuzzy model.

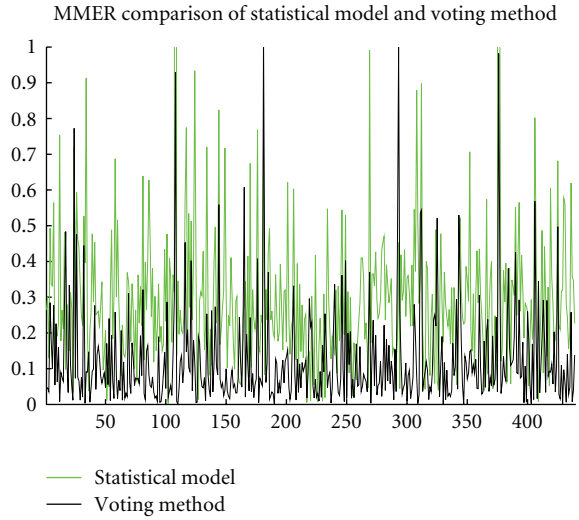

Figure 9: MMER comparison between voting method and statistical model.

bidding and planning, which are critical practices in the software industry. Second, no single software development estimation technique is ideal for all situations. Finally, a careful comparison of several approaches and their respective results can be useful for producing realistic estimates.

The results from Fuzzy Logic, Neural Network, Neurofuzzy, Multiple Regression, and Statistical Models were compared to the voting method based on the MMRE and the MMER. Results showed that the voting approach has a lower MMER in contrast with other models, and that the MMRE of the voting model is slightly higher than that of the Neurofuzzy Model, which has the best MMRE.

The primary limitation of this study involved too many inputs for the models. Perhaps by using Genetic Algorithms we can choose the most important features among the inputs in order to reduce them. Moreover, we did not take into account the factor primary programming languages for reducing redundancy, since they belong to one of

the languages types. Also, there are many development techniques in the dataset, such as waterfall, and prototyping, which can be combined with each other, but in practice, it is very difficult to consider all of them. Nevertheless, these limitations give us motivation to continue this research in future.

References

[1] I. F. de Barcelos Tronto, J. D. da Silva, and N. Sant'Anna, "Comparison of artificial neural network and regression models in software effort estimation," in *Proceedings of IEEE International Conference on Neural Networks (IJCNN '07)*, pp. 771–776, Orlando, Fla, USA, August 2007.

[2] H. Park and S. Baek, "An empirical validation of a neural network model for software effort estimation," *Expert Systems with Applications*, vol. 35, no. 3, pp. 929–937, 2008.

[3] B. Boehm, E. Horowitz, R. Madachy, et al., *Software Cost Estimation with COCOMO II*, Prentice-Hall, Upper Saddle River, NJ, USA, 2000.

[4] L. H. Putnam and W. Myers, *Measures of Excellence*, Prentice-Hall, Upper Saddle River, NJ, USA, 1992.

[5] X. Huang, D. Ho, J. Ren, and L. F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach," *Applied Soft Computing Journal*, vol. 7, no. 1, pp. 29–40, 2007.

[6] M. O. Saliu, M. Ahmed, and J. AlGhamdi, "Towards adaptive soft computing based software effort prediction," in *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS '04)*, vol. 1, pp. 16–21, 2004.

[7] C. L. Martin, J. L. Pasquier, C. M. Yanez, and A. G. Agustin, "Software development effort estimation using fuzzy logic: a case study," in *Proceedings of the 6th Mexican International Conference on Computer Science (ENC '05)*, pp. 113–120, September 2005.

[8] Z. Jiang and P. Naudé, "An examination of the factors influencing software development effort," *International Journal of Computer Information and Systems Science and Engineering*, vol. 1, no. 3, pp. 182–191, 2007.

[9] J. Martin, *Rapid Application Development*, Macmillan, New York, NY, USA, 1991.

[10] G. H. Subramanian and G. E. Zarnich, "An examination of some software development effort and productivity determinants in ICASE tool projects," *Journal of Management Information Systems*, vol. 12, no. 4, pp. 143–160, 1996.

[11] W. N. Venables and B. D. Ripley, *Modern Applied Statistics*, Springer, New York, NY, USA, 2002.

[12] A. R. Gray and S. MacDonell, "Applications of fuzzy logic to software metric models for development effort estimation," in *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS '97)*, pp. 394–399, 1997.

[13] S. G. MacDonell, A. R. Gray, and J. M. Calvert, "FULSOME: fuzzy logic for software metric practitioners and researchers," in *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP '99)*, pp. 308–313, IEEE Computer Society Press, Perth, Western Australia, 1999.

[14] A. R. Gray and S. G. MacDonell, "Fuzzy logic for software metric models throughout the development life-cycle," in *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS '99)*, pp. 258–262, New York, NY, USA, July 1999.

[15] M. T. Su, T. C. Ling, K. K. Phang, C. S. Liew, and P. Y. Man, "Enhanced software development effort and cost estimation using fuzzy logic model," *Malaysian Journal of Computer Science*, vol. 20, no. 2, pp. 199–207, 2007.

[16] W. Xia, L. F. Capretz, D. Ho, and F. Ahmed, "A new calibration for function point complexity weights," *Information and Software Technology*, vol. 50, no. 7-8, pp. 670–683, 2008.

[17] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Information and Software Technology*, pp. 911–922, 2002.

[18] C. López-Martín, C. Yáñez-Márquez, and A. Gutiérrez-Tornés, "Predictive accuracy comparison of fuzzy models for software development effort of small programs," *Journal of Systems and Software*, vol. 81, no. 6, pp. 949–960, 2008.

[19] J. Wong, D. Ho, and L. F. Capretz, "Calibrating function point backfiring conversion ratios using neuro-fuzzy technique," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 16, no. 6, pp. 847–862, 2008.