2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing

Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model

Ali Bou Nassif and Luiz Fernando Capretz
Department of ECE, Western University
London, Ontario, Canada
{abounas, lcapretz}@uwo.ca

Danny Ho
NFA Estimation Inc.
Richmond Hill, Ontario, Canada
danny@nfa-estimation.com

Abstract— **Software cost estimation is a crucial element in project management. Failing to use a proper cost estimation method might lead to project failures. According to the Standish Chaos Report, 65% of software projects are delivered over budget or after the delivery deadline. Conducting software cost estimation in the early stages of the software life cycle is important and this would be helpful to project managers to bid on projects. In this paper, we propose a novel model to predict software effort from use case diagrams using a cascade correlation neural network approach. The proposed model was evaluated based on the MMER and PRED criteria using 214 industrial and 26 educational projects against a multiple linear regression model and the Use Case Point model. The results show that the proposed cascade correlation neural network can be used with promising results as an alternative approach to predict software effort.**

Keywords- cascade correlation neural network, early software estimation, project management, Use Case Points

I. INTRODUCTION

Several cost estimation techniques have been used in the last few decades. These techniques include algorithmic models, expert judgment, estimation by analogy and machine learning. Algorithmic models are the oldest and most famous methods used in software cost estimation. Examples of algorithmic models include COCOMO [1], SLIM [2], Function Points [3] and Use Case Points [4]. In algorithmic models, linear and non-linear regression models are used to predict software effort. The dependent variable is usually software effort, where the independent variables include software size and some non-functional requirements. Expert judgment involves consulting a group of experts to use their experiences to propose an estimation of a given project [5]. The Delphi technique [6] is used to provide communication and cooperation among the experts. Estimation by analogy is a method in which the proposed project is compared to similar historical projects where all required information about the historical projects is documented. Estimation by analogy is actually a systematic form of expert judgment since experts look for analogies. Shepperd et al.'s work [7] is one of the prominent works in estimating software effort using analogy. Machine learning techniques include fuzzy logic, neural networks, neuro-fuzzy and genetic algorithm. Machine learning models have been very popular in the last two decades and they can be used as standalone models for software effort prediction or can be used in conjunction with other models such as algorithmic, estimation by analogy and expert judgment models.

In this research investigation, we propose a novel Cascade Correlation Neural Network (CCNN) model to predict software effort from use case diagrams. The use case diagrams are developed in the early stages of the software life cycle and thus, the proposed model can be used for early software effort estimation. Our CCNN model has three inputs. These inputs include software size, team productivity and project complexity. Software size is computed based on the use case point (UCP) model. Team productivity factor is also calculated based on the UCP model with some modifications. Regarding project complexity, a new method to calculate the complexity value is proposed. To validate the proposed model, a multiple linear regression model was developed using the same training points that were used to train the CCNN model. Then, the proposed CCNN model is compared with the multiple linear regression model as well as the UCP model using the MMER and PRED criteria. There are two main hypotheses raised in this research. These include:

1. The estimation accuracy based on the MMER and PRED criteria of the proposed CCNN model is better than that of the multiple linear regression model.
2. The estimation accuracy based on the MMER and PRED criteria of the proposed CCNN model is better than that of the UCP model

The remainder of the paper is organized as follows: Section II presents the background of the work. Section III lists some related work. Section IV demonstrates the CCNN and the multiple linear regression models. The evaluation of the proposed model is conducted in Section V. Finally, Section VI concludes the paper.

II. BACKGROUND

This section defines the main terms used in this paper which includes the UCP model, evaluation criteria, regression analysis and neural network.

A. Use Case Point Model

The use case point (UCP) model was first described by Gustav Karner in 1993 [4]. This model is used for software cost estimation based on the use case diagrams. First, the software size is calculated according to the number of actors and use cases in a use case diagram multiplied by their complexity weights, as shown in Tables I and II.

978-0-7695-4761-9/12 $26.00 © 2012 IEEE
DOI 10.1109/SNPD.2012.40

IEEE computer society

The software size is calculated through two stages. These include the Unadjusted Use Case Points (UUCP) and the Adjusted Use Case Points (UCP). UUCP is achieved through the summation of the Unadjusted Use Case Weight (UUCW) and Unadjusted Actor Weight (UAW). UUCW is represented in Equation (1).

$$UUCW = \sum_{i=1}^{3} n_i \times w_i. \quad (1)$$

Where n_i is the number of items of variety i of the use cases and w_i is the complexity weight of the corresponding use case. Similarly, UAW is represented as follows:

$$UAW = \sum_{j=1}^{3} m_j \times c_j. \quad (2)$$

Where m_j is the number of items of variety j of the actors and C_j is the complexity weight of the corresponding actor. Consequently, UUCP can be defined as follows:

$$UUCP = UUCW + UAW. \quad (3)$$

After calculating the UUCP, the Adjusted Use Case Points (UCP) is calculated. UCP is achieved by multiplying UUCP by the Technical Factors (TF) and the Environmental Factors (EF). TF and EF represent the non-functional requirements of the software. The technical and environmental factors are depicted in tables III and IV, respectively. The technical factor is detailed as follows:

$$TF = 0.6 + 0.01 \sum_{i=1}^{13} T_i \times W_i. \quad (4)$$

Where T_i is a factor that takes values between 0 and 5. The value "0" indicates that the factor is unrelated while the value "5" indicates that the factor is indispensable. The value "3" specifies that the technical factor is not very important, nor irrelevant (average). W_i represents the weight of technical factors (Table III). On the other hand, the environmental factor (EF) can be described as follows:

$$EF = 1.4 - 0.03 \sum_{i=1}^{8} E_i \times W_i. \quad (5)$$

Where E_i is the Environmental Factor (which is similar to T_i in Equation 4), taking values between 0 and 5. Finally, the Adjusted Use Case Points (UCP) can be defined as follows:

$$UCP = UUCP \times TF \times EF. \quad (6)$$

TABLE I. COMPLEXITY WEIGHTS OF USE CASES [4]

Use Case Complexity	Number of Transactions	Weight
Simple	Less than 4	5
Average	Between 4 and 7	10
Complex	More than 7	15

TABLE II. COMPLEXITY WEIGHTS OF ACTORS [4]

Actor Complexity	Description	Weight
Simple	Through an API	1
Average	Through a text-based user interface	2
Complex	Through a Graphical User Interface	3

TABLE III. TECHNICAL FACTORS [4]

T_i	Complexity Factors	W_i
T_1	Easy installation	0.5
T_2	Portability	2
T_3	End user efficiency	1
T_4	Reusability	1
T_5	Complex internal processing	1
T_6	Special security features	1
T_7	Usability	0.5
T_8	Application performance objectives	1
T_9	Special user training facilities	1
T_{10}	Concurrency	1
T_{11}	Distributed systems	2
T_{12}	Provide direct access for third parties	1
T_{13}	Changeability	1

TABLE IV. ENVIRONMENTAL FACTORS [4]

E_i	Efficiency and Productivity Factors	W_i
E_1	Familiar with Objectory	1.5
E_2	Object oriented experience	1
E_3	Analyst capability	0.5
E_4	Stable requirements	2
E_5	Application experience	0.5
E_6	Motivation	1
E_7	Part-time workers	-1
E_8	Difficult programming language	-1

For effort estimation, The UCP model proposes 20 person-hours to develop one UCP as shown in Equation (7):

$$Effort = Size \times 20. \quad (7)$$

Where Effort is measured in person-hours and Size is measured in UCP.

B. Evaluation Criteria

Several methods exist to compare cost estimation models. Each method has its advantages and disadvantages. In our work, two different evaluation methods have been used. These methods include the Mean of Magnitude of Error Relative to the Estimate (MMER) and the Prediction Level (PRED)

MMER: The Magnitude of Error Relative to the estimate (MER) [8] for each observation i can be obtained as:

$$MER_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i|}{Predicted\ Effort_i}. \quad (8)$$

MMER can be achieved through the summation of MER over *N* observations:

$$MMER = \frac{1}{N}\sum_{1}^{N} MER_i . \quad (9)$$

PRED (x) can be described as defines the average fraction of the MER's off by no more than *x* as defined by [9]:

$$PRED(x) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 1 if\ MRE_i \le x \\ 0\ otherwise \end{cases} . \quad (10)$$

The estimation accuracy is proportional to PRED (x) and inversely proportional to MMER.

C. Cascade Correlation Neural Network

Cascade Correlation Neural Networks (CCNN) are also known as self-organizing networks, and were proposed by Fahlman et al. [10]. A CCNN network consists of an input, hidden and output layers. At the beginning of the training stage, a CCNN network is only composed of an input and output layers as shown in Figure 1. Each input is connected to each output. The "+1" sign represents the bias; however, the "x" sign represents an adjustable weight between the input and the output neurons. After that, neurons are added to the hidden layer one by one. Figure 2 shows the architecture of a CCNN network with one neuron. The inputs of a new hidden neuron are the existing input neurons as well as the output of the previous added neuron as shown in Figure 2. The square sign represents fixed weights. When a new hidden neuron is added, the training algorithm tries to maximize the magnitude of the correlation between the new hidden neuron's output and the residual error so that the residual error is reduced. If adding a new hidden neuron does not significantly reduce the residual error, the training process will halt.

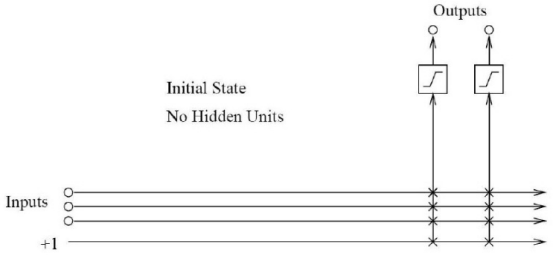

Figure 1. CCNN with no hidden neurons [10]

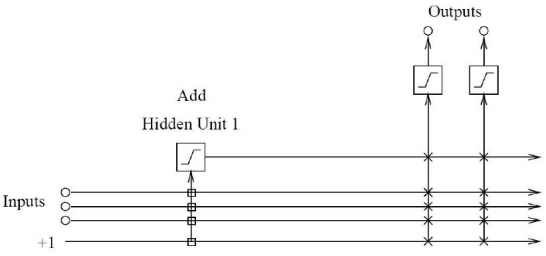

Figure 2. CCNN with one hidden neuron [10]

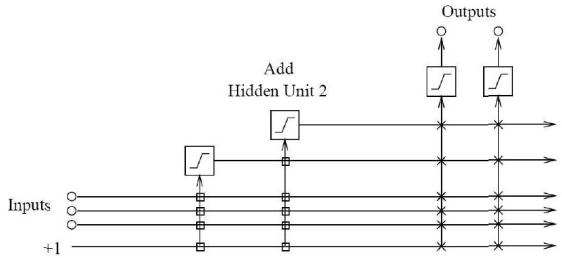

Figure 3. CCNN with two hidden neurons [10]

III. Related Work

Azzeh et al. [11] and [12] proposed two models for software effort estimation. The first one is an estimation- by-analogy model based on the integration of fuzzy set theory with grey relational analysis and fuzzy numbers. However, the second model is based on analogy estimation with fuzzy numbers and can be used in the early stages of the software life cycle. Both models were evaluated using five different datasets such as ISBSG, Desharnais, Kemerer, Albrecht & Gaffney and COCOMO 81. MMRE, MdMRE, MMER and PRED(25) were used as evaluation criteria. Results proved that the proposed models are competitive when compared with other models such as case-based reasoning, multiple linear regression, stepwise regression and ANN.

Pendharkar et al. [13] developed a Bayesian network to predict software development effort. The proposed model can incorporate decision making risks. The model was evaluated using 33 industrial projects and was compared with other neural network and regression tree forecasting models. The authors proved that their model can be a competitive model for software effort prediction based on the absolute error criterion.

Idri et al. [14] proposed two Radial Basis Function Neural Network model for software effort estimation. Each of the RBFNN models uses different formula to calculate the width of the RBF functions. The model was trained using COCOMO 81 and Tukutuku datasets and evaluated based on MMRE and PRED criteria.

Reddy et al. [15] proposed a RBFNN model for software effort estimation. The model was trained based on the k-mean clustering algorithm and was evaluated using COCOMO 81 dataset.

Heiat [16] compared a neural network model with regression models. The evaluation was conducted on 67 projects from three different sources. The author concluded that the neural network model was competitive to regression models when third generation language was used. However, regression models gave better results when combinations of third and fourth generation language projects were used. The evaluation criterion used was the mean absolute percentage error (MAPE).

Lopez-Martin [17], [18], [19] and [20] created regression models from short scale programs and from ISBSG repository. The author also developed fuzzy logic and neural network models such as Feed-forward and General

Regression Neural Networks. The authors proved that these models can be used as alternatives to regression models to predict software effort. The evaluation criteria used were the Mean of the Magnitude of Relative Error (MMRE) and the Mean of Magnitude of error Relative to the Estimate (MMER).

Other regression and machine learning models exist and are used to improve the accuracy of software estimation. Examples of these models include [21], [22], [23], [24], [25], [26] and [27].

None of the above work proposed a cascade correlation neural network model for software effort prediction from use case diagrams. Moreover, the proposed model takes into consideration some non-functional requirements (NFR), such as productivity and complexity.

IV. Regression and CCNN models

This section introduces the multiple linear regression and CCNN models. Our dataset contains 240 projects. Among these projects, 70% (168 projects) were randomly chosen to train the models and 30% (72 projects) were used to test the model. Each of the proposed models takes 3 inputs which include software size, productivity and project complexity. Software size was estimated based on the UCP model as described in Section II A. The productivity factor was calculated based on Table IV according to this equation:

$$Productivity = \sum_{i=1}^{8} E_i \times W_i. \quad (11)$$

Where E_i and W_i are the Environmental factors and their corresponding weights as depicted in Table IV. One adjustment has been made to the weight of E4 (Stable Requirements) by proposing a new weight of value "4" instead of "2". This adjustment was made because we have found that the accuracy of software estimation is highly influenced by the stability of the requirements. The value of E4 was adjusted by using the trial and error method. A statistical test was run after proposing a new value until the value of "4" was reached.

Regarding project complexity, we introduce five levels of complexity based on these rules:

- Level1: The complexity of a project is classified as Level1 if the project team is familiar with this type of project and the team has developed similar projects in the past. The number and type of interfaces are simple. The project will be installed in normal conditions where high security or safety factors are not required. Moreover, Level1 projects are those of which around 20% of their design or implementation parts are reusable (came from old similar projects). The weight of the Level1 complexity is 1.
- Level2: This is similar to the level1 category with a difference of that only about 10% of these projects are reusable. The weight of the Level2 complexity is 2.
- Level3: This is the normal complexity level where projects are not said to be simple, nor complex. In this level, the technology, interface, installation conditions are normal. Furthermore, no parts of the projects had been previously designed or implemented. The weight of the Level3 complexity is 3.
- Level4: In this level, the project is required to be installed on a complicated topology/architecture such as distributed systems. Moreover, in this level, the number of variables and interface is large. The weight of the Level4 complexity is 4.
- Level5: This is similar to Level4 but with additional constraints such as a special type of security or high safety factors. The weight of the Level5 complexity is 5.

A. Multiple Linear Regression Model

The multiple linear regression model was constructed using 168 data points. Minitab version 16 was used for this purpose. The equation of the regression model is:

$$Effort = 3661 + (32.7 \times Size) - (183 \times Productivity) + (1080 \times Complexity). \quad (12)$$

Where Effort is measured in person-hours and Size in UCP. Productivity is measured based on Equation (11) and Complexity is measured as proposed in Section IV. To measure the accuracy of the regression model, we measured the value of the coefficient of determination R^2 which is 0.882. This indicates that approximately 88 % of the variation in Effort can be explained by the independent variables Size, Complexity and Productivity. Moreover, we measured the Analysis of Variance (ANOVA) and the model parameters. The "p" value of the model is 0.000 which indicates that there is a significant relationship among the variables at the 95% confidence level. The "p" values of the independent variables are 0.000 and 0.0083 for the constant. Since the highest "p" value of the model's parameters is less than 0.05, this indicates that all independent variables are significant at the 95% confidence level, and consequently the model is verified.

B. Cascade Correlation Neural Network Model

The CCNN model was trained using DTREG software [28]. Like the multiple linear regression model, the CCNN model was trained using 168 data points. During the training stage of the CCNN model, a k-fold cross validation technique was used to prevent overfitting. When a new hidden neuron is added, both training and validation errors are measured. If the training error starts to decrease where the validation error starts to increase, the training process will start. In our experiments, the number of hidden neurons was set to 4 as shown in Figure 4. In this figure, the blue graph (lower one) represents the training stage where the red graph (higher one) represents the validation stage. This figure shows that after the number of hidden neurons becomes four, the training error is still decreasing, but the validation error

starts to increase. Figure 5 shows the relationship between the size and effort of the training points with the training curve. The R^2 and the Root Mean Squared (RMS) values reported in the training stage are 0.94 and 2,331, respectively. The R^2 and the Root Mean Squared (RMS) values reported in the validation stage are 0.81 and 4,373, respectively. The type of activation functions used in the hidden and output layers are the Gaussian and the linear functions, respectively.

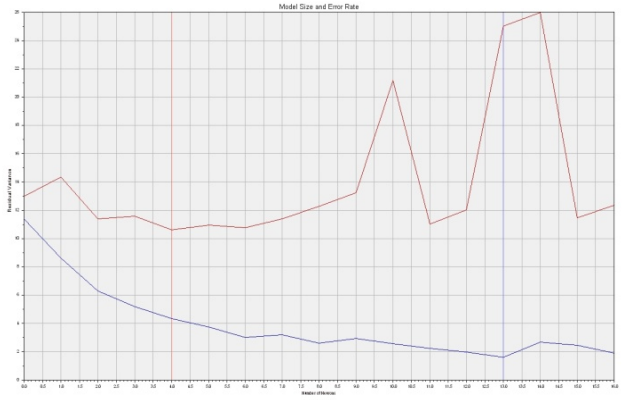

Figure 4. Number of hidden neurons

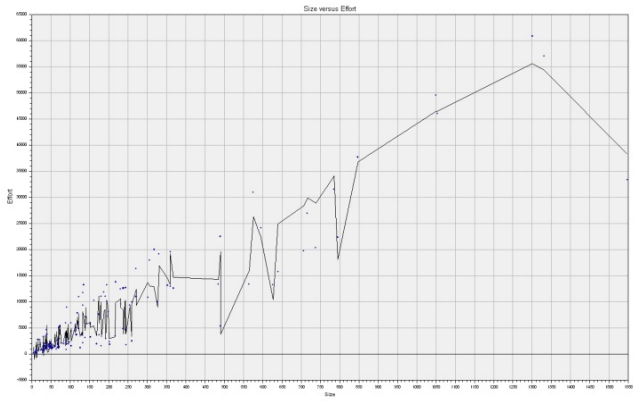

Figure 5. Training data points

V. Model Evaluation and Discussion

This section presents the evaluation of the proposed CCNN model against the multiple linear regression as well as the UCP model based on the MMER and PRED criteria.

A. Project Dataset

This research is based on software effort prediction from use case diagrams. We have encountered many difficulties in acquiring industrial projects because revealing UML diagrams of projects is considered confidential to many companies. For this reason, we have prepared a questionnaire that could help us obtain industrial data without actually having UML diagrams. In this questionnaire, we asked for example, the quantity of use cases in each project, the number of transactions in the use case description, actual software size and effort as well as some non-functional requirements such as the project complexity, and factors contributing to productivity. Two hundred and forty projects were collected from four main sources such that 214 are industrial projects from three different sources and 26 are educational ones.

B. Model Evaluation

The CCNN model was evaluated using 72 data points that were not included in the training stage against the multiple linear regression model as well as the UCP model. The criteria used for evaluation are the MMER, PRED(25), PRED(50), PRED(75) and PRED(100). Table V shows the values of the CCNN, multiple linear regression and UCP models. Figure 6 shows the interval plot at 95% confidence level of the MMER for the three models.

TABLE V. Model Evaluation

Criteria	CCNN	Multiple Linear Regression	UCP
MMER	0.54	0.57	0.99
PRED(25)	32	26.38	33.33
PRED(50)	57	55.55	48.61
PRED(75)	82	77.77	51.38
PRED(100)	89	86.11	61.11

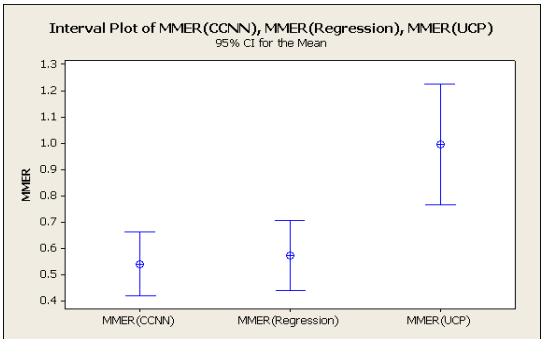

Figure 6. Intervel Plot of MMER

C. Discussion

Table V shows that the proposed CCNN model outperforms the multiple linear regression and UCP models by 3% and 45%, respectively based on the MMER criterion. Moreover, based on Figure 6, the width of the interval plot of the CCNN model is the shortest based on MMER. This means there is no major difference between the highest and lowest MMER values which is good as opposed to the plots of other models.

Based on the MMER values in Table V, the MMER value of the proposed CCNN model is lower than the MMER values of the multiple linear regression and UCP models. Although PRED(25) of the UCP model is slightly better than the PRED(25) of the CCNN model, PRED(50), PRED(75) and PRED(100) of the CCNN model are better than the corresponding PRED values of the UCP model. We conclude that the CCNN model outperforms the UCP model based on the MMER and PRED criteria. Hence, the two hypotheses proposed in Section I are proven.

VI. CONCLUSIONS

This paper proposed a novel correlation cascade neural network model to predict software effort from use case diagrams. The inputs of the model are software size, productivity and project complexity. To evaluate the CCNN model, a multiple linear regression model was developed that has the same inputs as the CCNN model. The multiple linear regression and the CCNN models were trained using 168 projects and evaluated using 72 projects. The CCNN model was then evaluated against the multiple linear regression model as well as the Use Case Point model. The results show that the proposed CCNN model outperforms the multiple linear regression and UCP models based on the MMER and PRED criteria and can be used as an alternative method to predict software effort from use case diagrams.

REFERENCES

[1] B. W. Boehm, *Software Engineering Economics.* Prentice-Hall, 1981.

[2] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering,* vol. 4, pp. 345-361, 1978.

[3] A. Albrecht, "Measuring application development productivity," in *IBM Application Development Symp.* 1979, pp. 83-92.

[4] G. Karner, "Resource Estimation for Objectory Projects," *Objective Systems,* 1993.

[5] R. T. Hughes, "Expert judgement as an estimating method," *Information and Software Technology,* vol. 38, pp. 67-75, 1996.

[6] N. Dalkey and O. Helmer, "An Experimental Application of the Delphi Method to the Use of Experts," *Management Science,* vol. 9, pp. pp. 458-467, 1963.

[7] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *Software Engineering, IEEE Transactions on,* vol. 23, pp. 736-743, 1997.

[8] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell and M. J. Shepperd, "What Accuracy Statistics Really Measure," *IEE Proc. - Softw,* vol. 148, no. 3, pp. 81-85, June, 2001.

[9] M. Jorgensen, "Experience with the accuracy of software maintenance task effort prediction models," *IEEE Transactions on Software Engineering,* vol. 21, pp. 674-681, 1995.

[10] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in Neural Information Processing Systems,* vol. 2, pp. 524-532, 1990.

[11] M. Azzeh, D. Neagu and P. Cowling, "Fuzzy grey relational analysis for software effort estimation," *Empirical Software Engineering,* vol. 15, pp. 60-90, 2010.

[12] M. Azzeh, D. Neagu and P. I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers," *Journal of Systems and Software,* vol. 84, pp. 270-284, 2011.

[13] P. C. Pendharkar, G. H. Subramanian and J. A. Rodger, "A probabilistic model for predicting software development effort," *Software Engineering, IEEE Transactions on,* vol. 31, pp. 615-624, 2005.

[14] A. Idri, A. Zakrani and A. Zahi, "Design of Radial Basis Function Neural Networks for Software Effort Estimation," *International Journal of Computer Science Issues,* vol. 7, pp. 11-17, 2010.

[15] C. S. Reddy, P. S. Rao, K. Raju and V. V. Kumari, "A New Approach For Estimating Software Effort Using RBFN Network," *International Journal of Computer Science and Network Security,* vol. 8, pp. 237-241, 2008.

[16] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Information and Software Technology,* vol. 44, pp. 911-922, 2002.

[17] C. Lopez-Martin, "A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables," *Applied Soft Computing,* vol. 11, pp. 724-732, 1, 2011.

[18] C. Lopez-Martin, "Applying a general regression neural network for predicting development effort of short-scale programs," *Neural Computing & Applications,* vol. 20, pp. 389-401, 2011.

[19] C. Lopez-Martín, C. Yanez-Marquez and A. Gutierrez-Tornes, "Predictive accuracy comparison of fuzzy models for software development effort of small programs," *Journal of Systems and Software,* vol. 81, pp. 949-960, 2008.

[20] C. Lopez-Martin, C. Isaza and A. Chavoya, "Software development effort prediction of industrial projects applying a general regression neural network," *Empirical Software Engineering,* vol. 17, pp. 1-19, 2011.

[21] W. L. Du, D. Ho and L. F. Capretz, "Improving Software Effort Estimation Using Neuro-Fuzzy Model with SEER-SEM," *Global Journal of Computer Science and Technology,* vol. 10, pp. 52-64, 2010.

[22] Y. Li, M. Xie and T. Goh, "Adaptive ridge regression system for software cost estimating on multi-collinear datasets," *Journal of Systems and Software,* vol. 83, pp. 2332-2343, 2010.

[23] G. Kousiouris, T. Cucinotta and T. Varvarigou, "The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks," *Journal of Systems and Software,* vol. 84, pp. 1270-1291, 2011.

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

[25] A. Mittal, K. Parkash and H. Mittal, "Software cost estimation using fuzzy logic," *SIGSOFT Softw. Eng. Notes,* vol. 35, no. 1, pp. 1-7, 2010.

[26] A. B. Nassif, D. Ho and L. F. Capretz, "Regression model for software effort estimation based on the use case point method," in *2011 International Conference on Computer and Software Modeling,* Singapore, 2011, pp. 117-121.

[27] A. B. Nassif, L. F. Capretz and D. Ho, "Estimating software effort based on use case point model using sugeno fuzzy inference system," in *23rd IEEE International Conference on Tools with Artificial Intelligence,* Florida, USA, 2011, pp. 393-398.

[28] P. Sherrod, "DTREG," *Software for Predictive Modeling and Forecasting,* 2011.