Estimating Software Effort Using an ANN Model Based on Use Case Points

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-- In this paper, we propose a novel Artificial Neural Network (ANN) to predict software effort from use case diagrams based on the Use Case Point (UCP) model. The inputs of this model are software size, productivity and complexity, while the output is the predicted software effort. A multiple linear regression model with three independent variables (same inputs of the ANN) and one dependent variable (effort) is also introduced. Our data repository contains 240 data points in which, 214 are industrial and 26 are educational projects. Both the regression and ANN models were trained using 168 data points and tested using 72 data points. The ANN model was evaluated using the MMER and PRED criteria against the regression model, as well as the UCP model that estimates effort from use cases. Results show that the ANN model is a competitive model with respect to other regression models and can be used as an alternative to predict software effort based on the UCP method.

Keywords-- Software Effort Estimation, Use Case Points, Artificial Neural Network.

I. INTRODUCTION

Software estimation is a crucial element in software engineering and project management. Incorrect software estimation leads to late delivery, surpassing the budget and project failures. According to the International Society of Parametric Analysis (ISPA) [1] and the Standish Group International [2], the main reasons behind project failures include optimism in conducting software estimation as well as misunderstanding and uncertainty in software requirements. At the inception of each software project, project managers use several techniques to predict software size and effort that will help them learn the cost, required time and the number of staff required to develop a project [3]. Examples of these techniques include Algorithmic Models such as COCOMO [4], SLIM [5] and SEER-SEM [6], Expert Judgment [7], Estimation by Analogy [8] and Machine Learning techniques.

In this paper, we present a novel Artificial Neural Network (ANN) model to estimate software effort based on the UCP method. The importance of our model is that it can be used in the early stages of the software life cycle where software estimation is required and difficult to conduct at this phase [9]. The proposed ANN model takes three inputs which include software size, productivity and project complexity. Software size and productivity are estimated using the UCP model [10]. A new approach to calculate the project complexity of a project is also introduced. To better evaluate the proposed ANN model, we introduce a multiple linear regression model to predict software effort based on three independent variables. We then tested the ANN model against the regression model as well as the UCP model based on the Mean of Magnitude of error Relative to the Estimate (MMER) and prediction level PRED. Results show that the ANN model outperforms the multiple linear regression model and UCP models based on the MMER criterion by 8% and 50% respectively, and thus, can be a competitive model for software effort prediction.

The remainder of this paper is organized as follows: Section II presents a background of terms used in this paper. Section III introduces related work whereas Section IV introduces the model's inputs. Section V illustrates the proposed ANN and multiple linear regression models. In Section VI, the proposed ANN will be evaluated and in Section VII, threats to validity are listed. Finally, Section VIII concludes the paper and suggests future work.

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 [10]. 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. The complexity weights of use cases and actors are presented in tables I and II, respectively. 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). 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 factors are depicted in tables II and III, respectively.

TABLE I. COMPLEXITY WEIGHTS OF USE CASES [10]

Use Case Complexity	**Number of Transactions**	**Weight**
Simple	Less than 4 (should be realized by less than 5 classes)	5
Average	Between 4 and 7 should be realized between 5 and 10 classes)	10
Complex	More than 7 (should be realized by more than 10 classes)	15

TABLE II. TECHNICAL FACTORS [10]

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 III. ENVIRONMENTAL FACTORS [10]

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, Karner proposed 20 person-hours to develop each UCP.

B. Evaluation Criteria

In our work, two different evaluation methods have been used which are the Mean of Magnitude of Error Relative to the Estimate (MMER) and the Prediction Level (PRED).

MER: The Magnitude of Error Relative to the estimate for each observation i can be obtained as:

$$MER_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i\ |}{Predicted\ Effort_i}. \quad (1)$$

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

$$MMER = \frac{1}{N}\sum_{1}^{N} MER_i. \quad (2)$$

PRED (x) can be described as:

$$PRED(x) = \frac{k}{n}. \quad (3)$$

where k is the number of projects in which $MER \leq x$ and n is the total number of projects. The estimation accuracy is directly proportional to PRED (x) and inversely proportional to MMER.

C. Neural Network

Artificial Neural Network (ANN) is a network composed of artificial neurons or nodes which emulate the biological neurons [11]. ANN can be trained to be used to approximate a non-linear function, to map an input to an output or to classify outputs. The most prominent topology of ANN is the feed-forward networks. Feed-forward ANN layers are usually represented as *input*, *hidden* and *output* layers. If the hidden layer does not exist, then this type of the ANN is called *perceptron*. The perceptron is a linear classifier that maps an input to an output provided that the output falls under two categories. The perceptron can map an input to an output if the relationship between the input and output is linear. If the relationship between the input and output is not linear, one or more hidden layers will exist between the input and output layers to accommodate the non-linear properties. Several types of feed-forward neural networks with hidden layers exist. These include Multilayer Perceptron (MLP), Radial Basis Function Neural Network (RBFNN) and General Regression Neural Network (GRNN). A MLP contains at least one hidden layer and each input vector is represented by a neuron. The number of hidden neurons varies and can be determined by trial and error so that the error is minimal. In this paper, MLP type is used to predict software effort based on software size calculated based on the UCP method, team productivity and project complexity. Figure 1 shows the ANN architecture used in this paper with three inputs and four hidden neurons. The selection process of the number of the hidden neurons is illustrated in Section V, B.

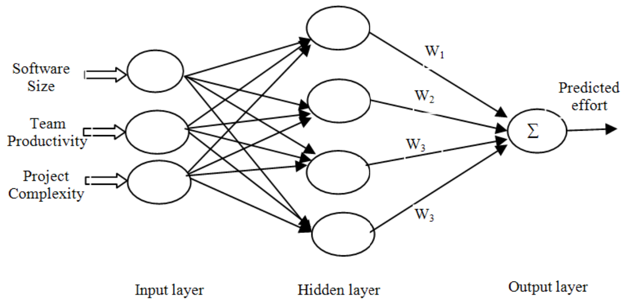

Figure 1. Architecture of ANN

III. RELATED WORK

Some issues related to the UCP model have been addressed in previous work. Authors in [12] and [13] worked on adjustment factors, while others in [13] and [14] highlighted the discrepancies in designing use case models. Researchers in [15], [16] and [17] proposed different size metrics such as Transactions, TTPoints and Paths, while others [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] and [29] went further to extend the UCP model by providing new complexity weights or by modifying the method used to predict effort.

Neural network models such as [30], [31], [32], [33], [34] and [35] were used to predict software effort.

Estimation using analogy such as [36], [8], [37], [36] and [38] was also used for software effort prediction.

None of the above work deals with creating neural network models to predict software effort based on the use case points model. Moreover, our model was evaluated on industrial projects that are considered large. Another contribution of this work is to simplify the project complexity factor proposed by the UCP model by introducing five levels of complexity levels as shown in Section IV.

IV. Model's Inputs

The inputs of the model are software size, productivity and complexity. Software size was estimated based on the UCP model as described in Section II, A.

The productivity factor was calculated based on Table III according to this equation:

$$Productivity = \sum_{i=1}^{8} E_i \times W_i . \tag{4}$$

Where E_i and W_i are the Environmental factors and their corresponding weights as depicted in Table III.

The complexity of the project is an important factor in software effort prediction. Complexity can be interpreted as an item having two or more elements [39] [40]. There are two dimensions of complexity. These include business scope such as schedule, cost, risk and technical aspect which is the degree of difficulty in building the product [40]. Technical complexity deals with the number of components of the product, number of technologies involved, number of interfaces and types of interfaces [40]. The project complexity can be classified as low complexity, medium complexity or high complexity [40]. Project complexity should be distinguished from other project characteristics such as size and uncertainty [39]. Complex projects require more effort to develop than simple projects that have the same size. In our research, we identify the project complexity based on five levels (from Level1 to Level5) [41] [42]. The reason behind defining five levels is to be compatible with other cost estimation models such as COCOMO where cost drivers are classified into five or six levels (such as Very Low, Low, Nominal, High, Extra High). Additionally, this classification is compatible to the project complexity classification in [40]. Each level has its corresponding weight. The five complexity levels are defined as follows:

- 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. Also, Level1 projects are those of which around 20% of their design or implementation parts are reused (came from old similar projects). The weight of the Level1 complexity is 1.
- Level2: This is similar to level1 category with a difference that only about 10% of these projects are reused. 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.

V. Regression and ANN models

This section introduces the multiple regression and ANN 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.

A. Multiple Linear Regression Model

The main goal of creating a multiple linear regression model from the training dataset is to compare the ANN model with the regression model. The ANN model is deemed to be valid if it outperforms the regression model. The multiple linear regression model was constructed using 168 data points. Before we apply regression on data, we applied a normality test on the variables "Effort" and "Size" and we found that "Effort" and "Size" were not normally distributed. For this purpose, ln(Effort) and ln(Size) were used instead of Effort and Size. The equation of the regression model is:

$$\begin{aligned} \ln(Effort) &= 3.53 + (0.88 \times \ln(Size)) \\ &-(0.009 \times Productivity) + (0.31 \times Complexity). \end{aligned} \tag{5}$$

To measure the accuracy of the regression model, we measured the value of the coefficient of determination R^2 which is 0.88. 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 is 0.000, this indicates that all independent variables are significant at the 95% confidence level, and consequently the model is verified. We also measured the Variance Inflation Factor (VIF) of each independent variable to see if the multicollinearity issue (when one independent variable has a relationship with other independent variables) exists. We found that the highest VIF factor is for the variable "Productivity" which is 1.65. This indicates that the multicollinearity issue does not exit (VIF is less than 4). Equation (5) also shows that "Size" and "Complexity" positively correlate to "Effort" where "Productivity" negatively correlates to "Effort". This means when software size or the complexity of the project increases, software effort would increase. On the other hand, when the team productivity of the project increases, the effort would decrease. This interpretation is compatible with the literature.

B. *Artificial Neural Network*

Figure 1 shows the architecture of the ANN used in this paper. Like the regression model, the ANN model was trained using 168 data points. One of the most important parameters of a ANN model is to determine the number of the hidden neurons. If the number is very small, the model will not fit the data points properly. However, if the number of the hidden neurons is too high, overfitting might occur. Overfitting occurs when the training error is very small but the validation/ testing error is large.
In our model, the conjugate gradient algorithm [43] is used for training. The initial number of the hidden neurons is set to one, and then it is incremented by one until optimal results are achieved. The parameters of the model are: Maximum Iterations = 10,000, Convergence Tolerance = 1.0e^{-5}, Minimum Gradient = 1.0e^{-6} and Minimum Improvement Delta = 1.0e^{-6}. To avoid overfitting, 20% of the training data were held out and used for validation. If the training error is decreasing and the validation error starts to increase, the training should be stopped to avoid overfitting. The 10-fold cross validation technique was used. At each number of hidden neurons, the residual variance is calculated. The residual variance determines how well the model fits the dataset. The smaller the variance, the more accurate the model is. Figure 2 shows that the smallest residual variance (12.13%) is achieved when the number of the hidden neurons is four. DTREG software was used in developing the ANN model [44]

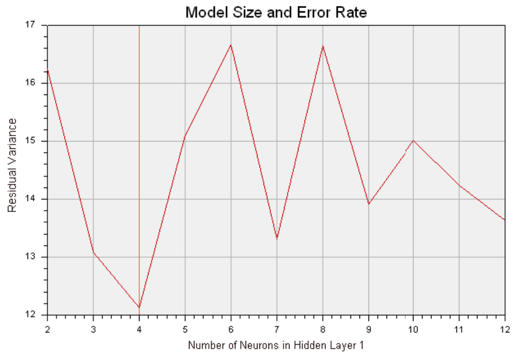

Figure 2. Number of Hidden Neurons

The type of activation function used in the hidden layer is the Sigmoid (Logistic); however, the linear function was used in the output layer. Figure 3 shows the actual versus the predicted effort values.

VI. Model Evaluation and Discussion

This section presents the evaluation of the proposed ANN model against the regression as well as the UCP model.

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 of each use case, actual software size and effort as well 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 and 26 are educational ones. The statistical profile of the project effort of the four datasets is depicted in Table IV. S1, S2, S3 and S4 correspond to Source1, Source2, Source3 and Source4, respectively, whereas Ind and Edu correspond to Industrial and Educational, respectively.

TABLE IV. Statistical Profile of Datasets

Source	#prj	Mean	StDev	Min	Max	Skew
S1(Ind)	13	36849.0	39350	4648	129353	1.37
S2(Ind)	156	6225.0	9258	120	60826	3.52
S3(Ind)	45	20573.0	47327	570	224890	3.26
S4(Edu)	26	1689.2	496.6	850	2380	-0.24

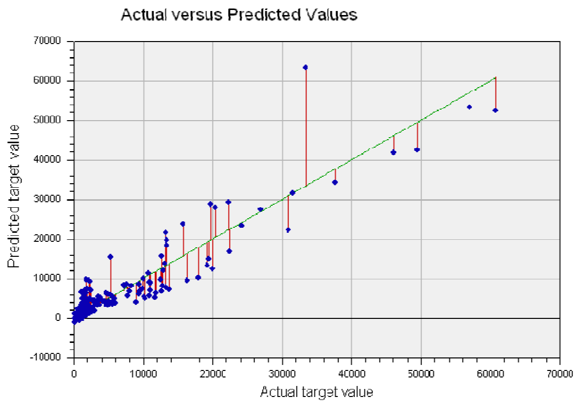

Figure 3. Actual Versus Predicted Effort

B. *Model Evaluation*

The ANN model was evaluated using 72 data points that were not included in the training stage. The criteria used are MMER, PRED(0.25), PRED(0.50), PRED(0.75) and PRED(1). Table V shows the values of the ANN, regression and UCP models. Figure 4 shows the interval plot at 95% confidence level of the MMER for the three models.

TABLE V. MODEL EVALUATION

Criteria	ANN	Regression	UCP
MMER	0.49	0.57	0.99
PRED(0.25)	29.16	29.16	33.33
PRED(0.50)	54.16	63.8	48.61
PRED(0.75)	86.11	87.5	51.38
PRED(1)	90.27	91.6	61.11

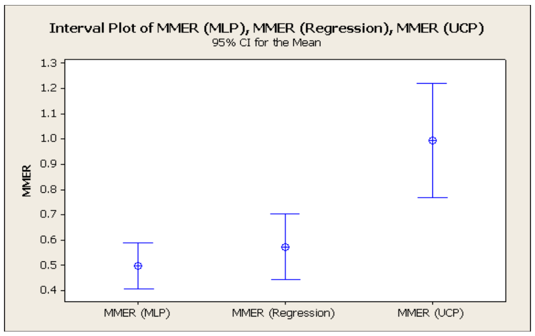

Figure 4. Intervel Plot for MMER

C. Discussion

Table V shows that the proposed ANN model outperforms the Regression and UCP models by 8% and 50% respectively based on MMER. The UCP model slightly surpasses the ANN model based on PRED(0.25). However, the ANN model gave better results in PRED(0.50), PRED(0.75) and PRED(1). Moreover, based on Figure 4, the width of the interval of the ANN model is the shortest based on MMER. This means there is no huge difference between the highest and lowest MMER values which is good as opposed to the interval plots of other models.

To thoroughly evaluate the ANN model against the UCP model, a statistical test has been conducted. We applied Anderson-Darling normality test and we found that the MER of all models are not normally distributed. For this reason, we used the non-parametric Mann-Whitney test to compare the ANN with the UCP model. We found that the p-value is 0.0246. This indicates that the results are statistically significant at 95% confidence level.

VII. THREATS TO VALIDITY

Threats to validity can be summarized as:

- We have encountered difficulties in collecting data especially industrial projects because companies do not reveal the UML models of their projects. For this reason, questionnaires were filled by people who work in the companies where data were collected. So we had to trust the information given to us about the datasets. For instance, an error in counting the number of the use cases or transactions will lead to an imperfection in the model's design and validation.
- It was difficult to elicit all the environmental factors (Table III) from the project team. For instance, employees might incorrectly answer questions that are related to their motivation of experience.
- Because of the lack of industrial projects, some educational projects (projects developed by students) were used. Students usually focus on the programming part when developing projects and ignore other stages in the software development life cycle, and this will underestimate the actual effort.

VIII. CONCLUSIONS

This paper proposed a new feed-forward Artificial Neural Network (ANN) model to predict software effort based on the use case points model. The inputs of the proposed model are software size, productivity and project complexity. To evaluate the ANN model, a multiple linear regression model was developed that has the same inputs as the ANN model. The regression and the ANN models were trained using 168 projects and evaluated using 72 projects. The ANN model was then evaluated against the regression model as well as the Use Case Point model. Results show that the proposed ANN model outperforms the regression and UCP models based on the MMER and PRED criteria and can be used an as alternative method to predict software effort from use case diagrams.

Future work will focus on trying other models such as Radial Basis Function Neural Network and General Regression Neural Network.

REFERENCES

[1] D. Eck, B. Brundick, T. Fettig, J. Dechoretz and J. Ugljesa, "Parametric estimating handbook," The International Society of Parametric Analysis (ISPA), Fourth Edition. 2009.

[2] J. Lynch. Chaos manifesto. The Standish Group. Boston. 2009[Online]. Available: http://www.standishgroup.com/newsroom/chaos_2009.php.

[3] A. B. Nassif, L. F. Capretz and D. Ho, "Software estimation in the early stages of the software life cycle," in *International Conference on Emerging Trends in Computer Science, Communication and Information Technology,* 2010, pp. 5-13.

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

[5] 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.

[6] D. D. Galorath and M. W. Evans, *Software Sizing, Estimation, and Risk Management.* Boston, MA, USA: Auerbach Publications, 2006.

[7] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *International Journal of Forecasting,* vol. 23, pp. 449-462, 2007.

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

[9] R. Ferguson, D. Goldenson, J. McCurley, R. Stoddard, D. Zubrow and D. Anderson, "Quantifying uncertainty in early lifecycle cost estimation (QUELCE)," Software Engineering

Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Tech. Rep. CMU/SEI-2011-TR-026, 2011.
[10] G. Karner, "Resource Estimation for Objectory Projects," *Objective Systems,* 1993.
[11] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine,* vol. 3, no.2, pp. 4-22, 1987.
[12] S. Diev, "Use cases modeling and software estimation: applying use case points," *SIGSOFT Softw. Eng. Notes,* vol. 31, pp. 1-4, 2006.
[13] B. Anda, H. Dreiem, D. I. K. Sjoberg and M. Jorgensen, "Estimating software development effort based on use cases-experiences from industry," in *Proceedings of the 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools,* 2001, pp. 487-502.
[14] M. Arnold and P. Pedross, "Software size measurement and productivity rating in a large-scale software development department," in *Proceedings of the 20th International Conference on Software Engineering,* 1998, pp. 490-493.
[15] G. Robiolo and R. Orosco, "Employing use cases to early estimate effort with simpler metrics," *Innovations in Systems and Software Engineering,* vol. 4, pp. 31-43, 2008.
[16] G. Robiolo, C. Badano and R. Orosco, "Transactions and paths: Two use case based metrics which improve the early effort estimation," in *International Symposium on Empirical Software Engineering and Measurement,* 2009, pp. 422-425.
[17] M. Ochodek and J. Nawrocki, "Automatic transactions identification in use cases," in *Balancing Agility and Formalism in Software Engineering*, B. Meyer, J. R. Nawrocki and B. Walter, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 55-68.
[18] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-UCP) model," in *International Conference on Computational Intelligence and Software Engineering,* 2009.
[19] F. Wang, X. Yang, X. Zhu and L. Chen, "Extended use case points method for software cost estimation," in *International Conference on Computational Intelligence and Software Engineering,* 2009.
[20] G. Schneider and J. P. Winters, *Applied use Cases, Second Edition, A Practical Guide.* Addison-Wesley, 2001.
[21] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *COMPSAC '06,* 2006, pp. 221-228.
[22] 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.
[23] 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.
[24] P. Mohagheghi, B. Anda and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," in *Proceedings of the 27th International Conference on Software Engineering,* St. Louis, MO, USA, 2005, pp. 303-311.
[25] S. Frohnhoff and G. Engels, "Revised use case point method - effort estimation in development projects for business applications," in *11th International Conference on Quality Engineering in Software Technology (CONQUEST 2008),* 2008, pp. 15-32.
[26] A. B. Nassif, L. F. Capretz and D. Ho, "Enhancing Use Case Points Estimation Method using Soft Computing Techniques," *Journal of Global Research in Computer Science,* vol. 1, pp. 12-21, 2010.
[27] A. B. Nassif, L. F. Capretz and D. Ho, "A regression model with mamdani fuzzy inference system for early software effort estimation based on use case diagrams," in *Third International Conference on Intelligent Computing and Intelligent Systems,* Guangzhou, Guangdong, 2011, pp. 615-620.
[28] A. B. Nassif, D. Ho and L. F. Capretz. "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software,* 2012, DOI: 10.1016/j.jss.2012.07.050.
[29] M. Ochodek, J. Nawrocki and K. Kwarciak, "Simplifying effort estimation based on Use Case Points," *Information and Software Technology,* vol. 53, pp. 200-213, 2011.
[30] 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.
[31] 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.
[32] 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.
[33] 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.
[34] Y. Kultur, B. Turhan and A. Bener, "Ensemble of neural networks with associative memory (ENNA) for estimating software development costs," *Knowledge-Based Systems,* vol. 22, pp. 395-402, 8, 2009.
[35] 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.
[36] 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.
[37] E. Kocaguneli, T. Menzies, A. B. Bener and J. W. Keung, "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation," *IEEE Transactions on Software Engineering,* vol. 38, pp. 425-438, 2012.
[38] A. Tosun, B. Turhan and A. B. Bener, "Feature weighting heuristics for analogy-based effort estimation models," *Expert Systems with Applications,* vol. 36, pp. 10325-10333, 2009.
[39] D. Baccarini, "The concept of project complexity—a review," *Int. J. Project Manage.,* vol. 14, pp. 201-204, 8, 1996.
[40] L. Ireland, "Project complexity: A brief exposure to difficult situations," Asapm, Tech. Rep. PrezSez 10-2007, 2007.
[41] A. B. Nassif, L. F. Capretz and D. Ho, "Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model," in *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD),* Kyoto, Japan, 2012, pp. 589-594.
[42] A. B. Nassif, "Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models," *Western University,* 2012.
[43] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks,* vol. 6, pp. 525-533, 1993.
[44] P. Sherrod, "DTREG," *Software for Predictive Modeling and Forecasting,* 2011.