World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:10, 2008

Estimating Development Time of Software Projects Using a Neuro Fuzzy Approach

Venus Marza, Amin Seyyedi, and Luiz Fernando Capretz

Abstract**—**Software estimation accuracy is among the greatest challenges for software developers. This study aimed at building and evaluating a neuro-fuzzy model to estimate software projects development time. The forty-one modules developed from ten programs were used as dataset. Our proposed approach is compared with fuzzy logic and neural network model and Results show that the value of MMRE (Mean of Magnitude of Relative Error) applying neuro-fuzzy was substantially lower than MMRE applying fuzzy logic and neural network.

Keywords**—**Artificial Neural Network, Fuzzy Logic, Neuro-Fuzzy, Software Estimation

I. Introduction

MANY existing research papers have proposed various estimation techniques, but no single software development estimation technique is the best for all situations [1]. A careful comparison of the results of the several approaches is most likely to choose the best one and produce realistic estimates [2]. The neural network research started in the 1940s, and the fuzzy logic research started in the 1960s, but the neuro-fuzzy research area is relatively new [3]. The objective of this paper is to present a feasible way of combining fuzzy logic and neural networks for achieving higher accuracy.

Neural network techniques are based on the principle of learning from historical data, whereas fuzzy logic is a method used to make rational decisions in an environment of uncertainty and vagueness. However, fuzzy logic alone does not enable learning from the historical database of software projects. Once the concept of fuzzy logic is incorporated into neural network, the result is a neuro-fuzzy system that combines the advantages of both techniques [4]. A software tool (MATLAB 7.4) was used to process the fuzzy logic, neural network and neuro-fuzzy systems.

The paper is organized as follows: Section 2 reviews some related work in fuzzy logic and neural network domain, section 3 discusses fuzzy logic approach for time estimation in software development, section 4 describes neural network techniques for time estimation, section 5 begins with a brief discussion of neuro-fuzzy model in general and this is followed by comparison between three described approaches, finally section 6 offers conclusions and a recommendation for future research.

Venus Marza is an MSc student of computer engineering, Islamic Azad University of South Tehran Branch, Tehran, Iran (e-mail: venus.marza@gmail.com).

Amin Seyyedi is a member of Department of Computer Engineering, Islamic Azad University of Maku Branch, (e-mail: amseyyedi@gmail.com).

Luiz Fernando Capretz is a professor at the Department of Electrical and Computer Engineering, University of London, Ontario, Canada N6A, Tel.: +1 519 661 2111Fax: +1 519 850 2436 (e-mail: lcapretz@eng.uwo.ca).

II. Related Work

Estimation accuracy is largely affected by modeling accuracy [5]. Finding good models for software estimation is very critical for software engineering in bidding and planning. In the recent years many software estimation models have been developed [6], [7], [8], [9].

López Martín *et al.* [6] proposed a fuzzy logic model for development time estimation. Ting su *et al.* [7] described an enhanced fuzzy logic model for the estimation of software development effort which had the similar capabilities as the previous fuzzy logic model in addition to enhancements in empirical accuracy in terms of MMRE. Abbas Heiat [8] used artificial neural network techniques like RBF (Radial Basis Function) and MLP (Multi-Layer Perceptron) for estimating software development effort. Furthermore, Xishi Huang *et al.* [9] developed a novel neuro-fuzzy Constructive Cost Model (COCOMO) for software cost estimation which uses the desirable features of a neuro-fuzzy approach, such as learning ability and good interpretability, in COCOMO model.

III. Fuzzy Logic Approach

Since fuzzy logic foundation by Zadeh in 1965, it has been the subject of important investigations [10]. It is a mathematical tool for dealing with uncertainty and also it provides a technique to deal with imprecision and information granularity [11].

The purpose in this section is not to discuss fuzzy logic in depth, but rather to present these parts of the subject that are necessary for understanding of this paper and for comparing it with Neuro-Fuzzy model.

Fuzzy logic offers a particularly convenient way to generate a keen mapping between input and output spaces thanks to fuzzy rules' natural expression [12]. There are some major modules: first stage transformed the classification tables into a continuous classification, this process is called Fuzzification [13]. These are then processed in fuzzy domain by inference engine based on knowledge base (rule base and data base) supplied by domain experts [14]. Finally the process of translating back fuzzy numbers into single "real world" values is named Defuzzification [13].

Here, the development time of forty-one modules and for each module, coupling (Dhama), complexity (McCabe), and lines of code metrics were registered, all programs were written in Pascal, hence, module categories belong to procedures or functions. The development time of each of the forty-one modules were registered including five phases: requirements understanding, algorithm design, coding, compiling and testing. The statistics and a brief description related to each module are depicted in Table I which is prepared by Lopez-Martin *et al.* [6].

TABLE I
MODULES DESCRIPTION AND METRICS

	Module Description	MC	DC	LOC	DT (min)
1	Calculates t value	1	0.25	4	13
2	Inserts a new element in a linked list	1	0.25	10	13
3	Calculates a value according to normal distribution equation	1	0.333	4	9
4	Calculates the variance	2	0.083	10	15
5	Generates range square root	2	0.111	23	15
6	Determines both minimum and maximum values from a stored linked list	2	0.125	9	15
7	Turns each linked list value into its z value	2	0.125	9	16
8	Copies a list of values from a file to an array	2	0.125	14	16
9	Determines parity of a number	2	0.167	7	16
10	Defines segment limits	2	0.167	8	18
11	From two lists (X and Y), returns the product of all xi and yi values	2	0.167	10	15
12	Calculates a sum from a vector and its average	2	0.167	10	15
13	Calculates q values	2	0.167	10	18
14	Generates the sum of a vector components	2	0.2	10	13
15	Calculates the sum of a vector values square	2	0.2	10	14
16	Calculates the average of the linked list values	2	0.2	10	15
17	Counts the number of lines of code including blanks and comments	2	0.2	15	13
18	Prints values non zero of a linked list	2	0.25	10	12
19	Stores values into a matrix	2	0.25	10	12
20	Generates range square root	3	0.083	17	22
21	Returns the number of elements in a linked list	3	0.125	11	19
22	Calculates the sum of odd segments (Simpson's formula)	3	0.125	15	18
23	Calculates the sum of pair segments (Simpson's formula)	3	0.125	15	19
24	Generates the standard deviation of the linked list values	3	0.143	13	21
25	Returns the sum of square roots of a list values	3	0.143	14	20
26	Prints a matrix	3	0.143	14	21
27	Calculates the sum of odd segments (Simpson's formula)	3	0.143	15	19
28	Calculates the sum of pair segments (Simpson's formula)	3	0.143	15	20
29	Calculates the average of linked list values	3	0.167	13	15
30	Returns the sum of a list of values	3	0.167	14	13
31	Generates the standard deviation of linked list values	3	0.2	18	19
32	Prints a linked list	3	0.25	9	13
33	Calculates gamma value (G)	3	0.25	12	12
34	Calculates the average of vector components	3	0.25	17	12
35	Calculates the range standard deviation	4	0.077	16	21
36	Calculates beta 1 value	4	0.077	31	21
37	Returns the product between values of two vectors and the number of these pairs	4	0.111	16	19
38	Counts commented lines	4	0.2	24	18
39	Reduces final matrix (according to Gauss method)	5	0.143	22	24
40	Reduces a matrix (according to Gauss method)	5	0.143	22	25
41	Counts blank lines	5	0.2	22	18

MC: McCabe Complexity, DC: Dhama Coupling, LOC: Lines of Code, DT: Development Time(minutes)

Implementing a fuzzy system requires that the different categories of the different inputs be presented by fuzzy sets, which in turn is presented by membership functions. A natural membership function type that readily comes to mind is the triangular membership functions [15].

A triangular MF is a three-point (parameters) function, defined by minimum (a), maximum (c) and modal (b) values, that is MF(a, b, c) where $a \leq b \leq c$. Their scalar parameters (a, b, c) are defined as follows [2]:

$MF(x) = 0$ if $x < a$

$MF(x) = 1$ if $x = b$

$MF(x) = 0$ if $x > c$

Based on the correlation of the variables, fuzzy rules can be formulated. Correlation is the degree of relation between two pairs of variables which varies from -1.0 to +1.0. The equation of the Correlation Coefficient is the following [16]:

$$r = \frac{n[\sum(X_i.Y_i)] - (\sum X_i)(\sum Y_i)}{\sqrt{[n(\sum X_i^2) - (\sum X_i)^2][n(\sum Y_i^2) - (\sum Y_i)^2]}} \quad (1)$$

The result of computing Correlation as shows in Table II is indicated that there is an acceptable correlation between development time (DT) and the next three metrics: McCabe complexity (MC), Dhama coupling (DC), and lines of code (LOC), because their absolute values are higher than 0.5 [6].

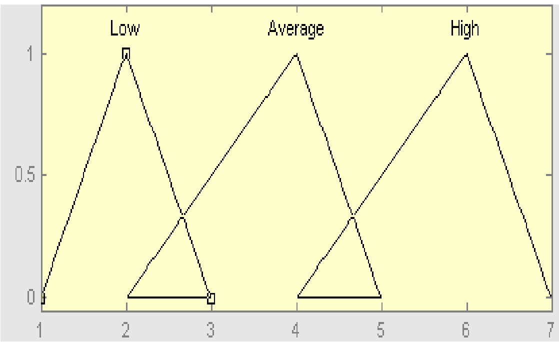
TABLE II
CORRELATION BETWEEN VARIABLES

Pair	r	Pair	r
MC_DC	-0.3860	DT_MC	0.7078
MC_LOC	0.7653	DT_DC	-0.7051
DC_LOC	-0.4346	DT_LOC	0.5827

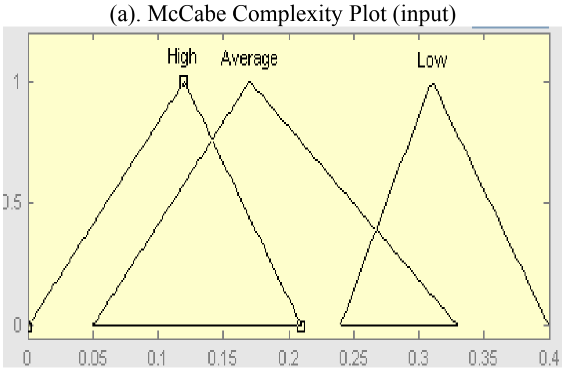
For example the absolute value of correlation between DT and DC is higher than 0.5, therefore if one of them be low another one should be low too. So by using Table II, six rules are derived [6]:

1. If *Complexity* is low and *Size(LOC)* is small then ***DT*** is low
2. If *Complexity* is average and *Size(LOC)* is medium then ***DT*** is average
3. If *Complexity* is high and *Size(LOC)* is big then ***DT*** is high
4. If *Coupling* is low then ***DT*** is low
5. If *Coupling* is average then ***DT*** is average

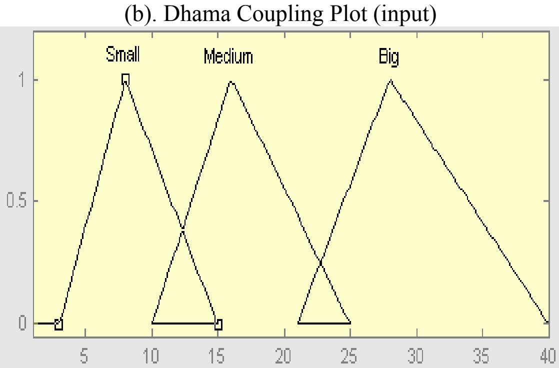
6. If *Coupling* is high then ***DT*** is high



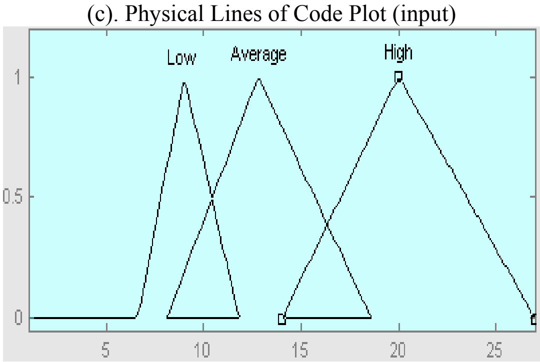
(a). McCabe Complexity Plot (input)



(b). Dhama Coupling Plot (input)



(c). Physical Lines of Code Plot (input)



(d). Development Time Plot (output)

Fig. 1 Inputs and Output Fuzzy Plots

By using triangular membership functions, input and output fuzzy membership functions are shown in Fig. 1.

IV. Neural Network Model

In recent years, a number of studies have used neural networks in various stages of software development [13]. Artificial neural network are used in estimation due to its ability to learn from previous data. In addition, it has the ability to generalize from the training data set thus enabling it to produce acceptable result for previously unseen data [7]. Artificial neural networks can model complex non-linear relationships and approximate any measurable function so it is very useful in problems where there is a complex relationship between inputs and outputs [14], [9].

When looking at a neural network, it immediately comes to mind that activation functions are look like fuzzy membership function [3].

Generally the radial basis function networks enjoy faster convergence than back-propagation networks [3]. So Radial Basis Function (RBF) model was used here. There are many techniques for training a neural network. The main techniques employed by neural networks are supervised and unsupervised learning [8]. RBF is in supervised category and finds a surface that best fits to given training data. Supervised training works in much the same way as a human learns new skills, by showing the network a series of examples [8]. Dataset is randomly divided into two parts: 25 of them are for training and all of them are used for validation. By MATLAB 7.4, RBF network was created, data were normalized between 0 and 1, and test data were applied into network. The results of this implementation are gathered in Table III.

V. Neuro-Fuzzy System

The hybridization of neural networks and fuzzy logic is the basic idea behind the neuro-fuzzy system. Neuro-fuzzy hybridization is done in two ways [17]: fuzzy neural networks (FNN) and neuro-fuzzy systems (NFS). FNN is a neural network equipped with the capability of handling fuzzy information. NFS is a fuzzy system augmented by neural networks to enhance some characteristics like flexibility and adaptability [18], [19], [20]. This paper is based on the second approach.

Here Takagi-Sugeno neuro-fuzzy system was used which makes use of a mixture of back propagation to learn the membership functions and least mean square estimation to determine the coefficients of the linear combination in the rule's conclusions. The Takagi-Sugeno neuro-fuzzy system schema is depicted in Fig. 2 [21]:

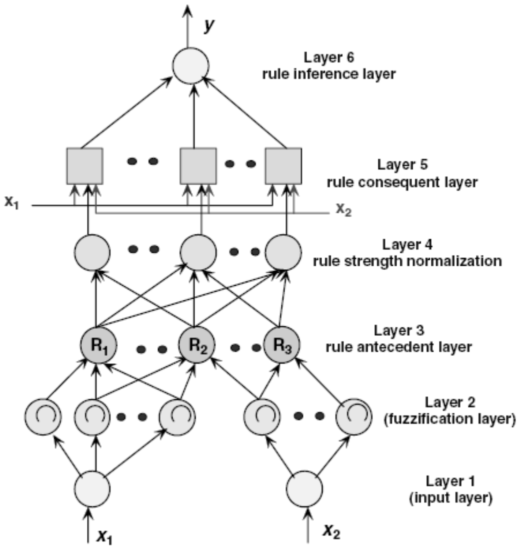


Fig. 2 Takagi-Sugeno Neuro_Fuzzy system

Perhaps the first integrated hybrid neuro-fuzzy model is ANFIS, and also due to Takagi-Sugeno rules implementation in ANFIS, it has lowest Root Mean Square Error (RMSE) among the other Neuro-Fuzzy models. So ANFIS was used here for implement neuro-fuzzy model and Its' architecture is very similar to Fig. 2.

In ANFIS, the adaptation (learning) process is only concerned with parameter level adaptation within fixed structures [21]. The objective of the parameter-learning phase is to adjust parameters of the fuzzy inference system (FIS) such that the error function during training dataset, reaches minimum or is less than a given threshold [22].

When Gaussian membership functions were used, operationally ANFIS can be compared with a radial basis function network. Our model was just trained at 20 epochs, also the previous training and testing data were used. The detailed functioning of each layer is as follows [21]: layer1, 2, 3 functions the same way as Mamdani FIS. Every node in layer 4 (rule strength normalization) calculates the ratio of i-th rule's firing strength to the sum of all rules firing strength:

$$\overline{w_i} = \frac{w_i}{w_1 + w_2} \quad , \quad i = 1, 2, \ldots \tag{2}$$

Every node in layer 5 (rule consequent layer) is with a node function:

$$\overline{w_i}\, f_i = \overline{w_i}\,(p_i x_1 + q_i x_2 + r_i) \tag{3}$$

Where $\overline{w_i}$ is the output of layer 4, and $\{p_i, q_i, r_i\}$ is the parameter set. A well-established way to determine the consequent parameters is using the least means squares algorithm. The single node in layer 6 (rule inference layer) computes the overall output as the summation of all incoming signals:

$$Overall\ output = \sum_i \overline{w_i}\, f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{4}$$

By MATLAB, the ANFIS structure with type: 'sugeno', and method: 'prod', or method: 'max', impMethod: 'prod' and aggMethod: 'max' was implemented and the results are given at Table III.

The Validation results of our experiments are assessed by Mean Magnitude Relative Error (MMRE) as estimation accuracy. MMRE is defined as [23]:

$$MMRE = \frac{1}{n} \sum_{i=1}^{i=n} \left(\frac{| T_i - \overline{T}_i |}{T_i} \right) = \frac{1}{n} \sum_{i=1}^{i=n} MRE_i \tag{5}$$

Where there are n projects; T_i is the Actual Time, and $\overline{T}_i$ is the Predicted Time.

TABLE III
THE MRE AND MMRE COMPARISON BETWEEN ESTIMATION MODELS

Module	Actual DT	Fuzzy Logic MRE	Neural Network MRE	Neuro-Fuzzy MRE
1	13	0.0000	0.1010	0.0000
2	13	0.0000	0.0000	0.0000
3	9	0.0167	0.0764	0.0000
4	15	0.1200	0.0616	0.0000
5	15	0.1867	0.0112	0.0714
6	15	0.0867	0.1118	0.0000
7	16	0.0188	0.1098	0.0000
8	16	0.0813	0.1363	0.1667
9	16	0.0250	0.2425	0.2222
10	18	0.1389	0.4339	0.2500
11	15	0.0400	0.0440	0.1667
12	15	0.0400	0.2247	0.0000
13	18	0.1333	0.0475	0.0000
14	13	0.0769	0.0096	0.0000
15	14	0.0000	0.1896	0.0000
16	15	0.0667	0.1037	0.0500
17	13	0.1615	0.1008	0.0455
18	12	0.0000	0.0454	0.0500
19	12	0.0000	0.3910	0.0000
20	22	0.2000	0.2221	0.0000
21	19	0.0737	0.1310	0.0000
22	18	0.0222	0.0374	0.0000
23	19	0.0737	0.1268	0.0000
24	21	0.1762	0.0442	0.0333
25	20	0.1350	0.0760	0.0000
26	21	0.1762	0.6917	0.0000
27	19	0.0947	0.1536	0.0833
28	20	0.1350	0.2796	0.0001
29	15	0.1133	0.1363	0.1667
30	13	0.2846	0.1471	0.0000
31	19	0.2000	0.0475	0.0000
32	13	0.0000	0.2263	0.0556
33	12	0.0833	0.2279	0.0000
34	12	0.0833	0.1758	0.0417
35	21	0.1905	0.0497	0.0455
36	21	0.1048	1.2159	0.0000
37	19	0.0947	0.5766	0.0000
38	18	0.1556	0.0879	0.0000
39	24	0.2792	0.8469	0.0000
40	25	0.3080	0.2495	0.0000
41	18	0.1556	0.1039	0.0313
		Fuzzy Logic	Neural Network	Neuro -Fuzzy
MMRE		0.1057	0.202305	0.036098

VI. CONCLUSIONS AND FUTURE RESEARCH

The paper suggests a new approach for estimating of software projects development time. The major difference between our work and previous works is that neuro-fuzzy technique is used for software development time estimation and then it's validated with gathered data. Here, the advantages of neural network and fuzzy logic are combined and learning ability and good generalization are obtained. The main benefit of this model is its good interpretability by using the fuzzy rules and another great advantage of this research is that it can put together expert knowledge (fuzzy rules) project data and the learning ability of neural network model into one general framework that may have a wide range of applicability in software estimation. The results showed that neuro-fuzzy system is much better than two other mentioned methods (fuzzy logic and neural network).

In order to achieve more accurate estimation, voting the estimated values of several techniques and combine their results maybe be useful.

REFERENCES

[1] H. Park, S. Baek, "An empirical validation of a neural network model for software effort estimation", Expert Systems with Applications, 2007.
[2] C. Lopez-Martin, C.Yanez-Marquez, A.Gutierrez-Tornes, "Predictive accuracy comparison of fuzzy models for software development effort of small programs, The journal of systems and software", Vol. 81, Issue 6, 2008, pp. 949-960.
[3] J. Jantzen, "Neuro-fuzzy modeling", Report no 98-H-874, 1998.
[4] W. Xia, L.F. Capretz, D. Ho, F.Ahmed, "A new calibration for function point complexity weights", Information and Software Technology, Vol.50, Issue 7-8, 2007, pp.670-683.
[5] M. Jorgensen, B. Faugli, T. Gruschke, "Characteristics of software engineers with optimistic prediction", Journal of Systems and Software, Vol. 80, Issue. 9, 2007, pp. 1472-1482.
[6] C.L. Martin, J.L. Pasquier, M.C. Yanez, T.A. Gutierrez, "Software Development Effort Estimation Using Fuzzy Logic: A Case Study", IEEE Proceedings of the Sixth Mexican International Conference on Computer Science (ENC'05), 2005, pp. 113-120.
[7] M.T. Su, T.C.Ling, K.K.Phang, C.S.Liew, P.Y.Man, "Enhanced Software Development Effort and Cost Estimation Using Fuzzy Logic Model", Malaysian Journal of Computer Science, Vol. 20, No. 2, 2007, pp. 199-207.
[8] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort", Information and Software Technology, Vol. 44, Issue 15, 2002, pp. 911-922.
[9] X. Huang, Danny Ho, J. Ren, L.F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach", Applied Soft Computing , Vol.7, Issue 1, 2007, pp. 29-40.
[10] A. Idri, A.Abran, "A Fuzzy Logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements", Proceedings of the seventh international software metrics symposium (METRICS '01), 2001, pp.85-96.
[11] S.N. Sivanandam, S. Sumathi, S.N. Deepa, "Introduction to fuzzy logic using MATLAB", Springer, 2007.
[12] A. Lotfi Zadeh, "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions", IEEE Transactions on Circuits and Systems, Fundamental Theory and Applications, Vol. 45, No 1, 1999, pp 105-119.
[13] M.R.Braz & S.R.Vergilio, "Using Fuzzy Theory for Effort Estimation of Object-Oriented Software", Proceedings of the 16th IEEE international Conference on Tools with Artificial Intelligence (ICTAI 2004), 2004, pp. 196-201.
[14] K.K.Aggarwal, Y.Singh, P.Chandra, M.Puri, "Sensitivity Analysis of Fuzzy and Neural Network Models", ACM SIGSOFT Software Engineering Notes, Vol. 30, Issue 4, 2005, pp. 1-4.
[15] A.A. Moataz, O.S.Moshood, A.Jarallah, "Adaptive fuzzy-logic-based framework for software development effort prediction", Information and Software Technology, Vol. 47, Issue 1, 2005, pp. 31-48.
[16] W.S. Humphrey, "A Discipline for Software Engineering", Addison Wesley, 2002.
[17] S. Mitra, Y.Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework", IEEE Transactions on Neural Networks, Vol.11, No.3, 2000, pp. 748-768.
[18] D. Nauck, F. Klawonn, R. Kruse, "Foundations of Neuro-Fuzzy Systems", Wiley, Chichester, 1997.
[19] D. Nauck, "A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches", In Proceedings of Fuzzy-Systeme'94, 2nd GI-Workshop, Munich, Semen Corporation, 1994.
[20] M.O. Saliu, "Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction", A Thesis Presented to the DEANSHIP OF GRADUATE STUDIES, King Fahd University of Petroleum & Minerals Dhahran, April 2003.
[21] A. Abraham, "Adaptation of Fuzzy Inference System Using Neural Learning", Springer-Verlag Berlin Heidelberg, 2005, pp. 59-83.
[22] Y. Shi, M. Mizumoto, N.Yubazaki, M. Otani, "A Learning Algorithm for Tuning Fuzzy Rules Based on the Gradient Descent Method", Proceedings of Fifth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96), New Orleans, USA, Vol.1, 1996, pp.55-61.
[23] V. Xia, L. F. Capretz, D. Ho, "A Neuro-Fuzzy Model for Function Point Calibration", WSEAS Transactions on Information Science & Applications, Vol. 5, Issue 1, 2008, pp. 22-30.

Venus Marza was born 1984, in Tehran, Iran. She got her BSc in software engineering from Islamic Azad University of North Tehran Branch, Tehran, Iran, in 2006. She is now an MSc student at the Azad University of South Tehran Branch, Tehran, Iran.

Luiz F. Capretz has over 22 years of experience in the software engineering field as a practitioner, manager and educator. Before joining the University of Western Ontario, in Canada, he has worked since 1981 at both the technical and managerial levels, taught and carried out research on the engineering of software in Brazil, Argentina, England and Japan. He was the Director of Informatics and Coordinator of the computer science program at two universities (UMC and COC) in the State of Sao Paulo/Brazil. He has authored and co-authored over 50 peer-reviewed research papers on software engineering in leading international journals and conference proceedings, and has co-authored the book, Object-Oriented Software: Design and Maintenance, published by World Scientific. His current research interests are software engineering (SE), human factors in SE, software estimation, software product lines, and software engineering education. Dr. Capretz received his PhD from the University of Newcastle upon Tyne (U.K.), MSc from the National Institute for Space Research (INPE-Brazil), and BSc from UNICAMP (Brazil). He is a senior member of IEEE.