Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 16 Issue 1 Version 1.0 Year 2016
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Neuro Fuzzy Algorithm to Compute Software Effort Estimation

By N. Shivakumar, N. Balaji & K. Ananthakumar

Thiagarajar College of Engineering, India

Abstract- Software Effort Estimation is highly important and considered to be a primary activity in software project management. The accurate estimates are conducted in the development of business case in the earlier stages of project management. This accurate prediction helps the investors and customers to identify the total investment and schedule of the project. The project developers define process to estimate the effort more accurately with the available mythologies using the attributes of the project. The algorithmic estimation models are very simple and reliable but not so accurate. The categorical datasets cannot be estimated using the existing techniques. Also the attributes of effort estimation are measured in linguistic values which may leads to confusion. This paper looks in to the accuracy and reliability of a non-algorithmic approach based on adaptive neuro fuzzy logic in the problem of effort estimation. The performance of the proposed method demonstrates that there is a accurate substantiation of the outcomes with the dataset collected from various projects. The results were compared for its accuracy using MRE and MMRE as the metrics. The research idea in the proposed model for effort estimation is based on project domain and attribute which incorporates the model with more competence in augmenting the crux of neural network to exhibit the advances in software estimation.

Keywords: *ANFIS, effort estimation, MRE, MMRE.*

GJCST-C Classification : *I.5.1 I.2.3*

Strictly as per the compliance and regulations of:

A Neuro Fuzzy Algorithm to Compute Software Effort Estimation

N. Shivakumar [α], N. Balaji [σ] & K. Ananthakumar [ρ]

Abstract- Software Effort Estimation is highly important and considered to be a primary activity in software project management. The accurate estimates are conducted in the development of business case in the earlier stages of project management. This accurate prediction helps the investors and customers to identify the total investment and schedule of the project. The project developers define process to estimate the effort more accurately with the available mythologies using the attributes of the project. The algorithmic estimation models are very simple and reliable but not so accurate. The categorical datasets cannot be estimated using the existing techniques. Also the attributes of effort estimation are measured in linguistic values which may leads to confusion. This paper looks in to the accuracy and reliability of a non-algorithmic approach based on adaptive neuro fuzzy logic in the problem of effort estimation. The performance of the proposed method demonstrates that there is a accurate substantiation of the outcomes with the dataset collected from various projects. The results were compared for its accuracy using MRE and MMRE as the metrics. The research idea in the proposed model for effort estimation is based on project domain and attribute which incorporates the model with more competence in augmenting the crux of neural network to exhibit the advances in software estimation.

Keywords: *ANFIS, effort estimation, MRE, MMRE.*

I. Introduction

Achieving software economics in large-scale software development projectsare very important today. Software effort estimation is the process of determining the accurate effort required to maintain or develop a software. It is always an important practical problem in software engineering which is still unsolved. Effort estimates are done in initial stages of software engineering to calculate the effort in person-months required for the software development. Accurate effort estimation helps in planning design construction and transition phases of development and prioritize the components in business case. Unreliable estimates is the main important reason for project failure, which is expressed in 2007 Comp TIA survey of thousand IT professionals, finding that three of the four most-cited causes of IT project failure are due to poor estimation.

Author α : Assistant Professor, Department of CSE, Thiagarajar College of Engineering, Madurai, India. e-mail: shiva@tce.edu
Author σ : Professor, Department of IT, K.L.N College of Engineering, Madurai, India. e-mail: balajin@klnce.edu
Author ρ : P.G. Student, Department of CSE, Thiagarajar College of Engineering, Madurai, India. e-mail: ananthakumar.k@outlook.com

[Rosencrance 2007]. Noticing the importance of reliable effort estimation, the software project management contributors are now focusing on developing models to generate accurate effort of software during the earlier stages of software development. Effort estimation for software projects are categorized as non-algorithmic and algorithmic models. Algorithmic models applies the mathematical computation method and the non-algorithmic estimation uses fuzzy, neural network and other machine learning techniques. The effectiveness of project management will be compromised if the Project managers are uncertain to adapt genuine estimation methodologies

Boehm proposed a method called COCOMO that utilizes some experimental equation to estimate the effort using inputs like Kilo lines of code (KLOC), number of functions and other effort drivers. Neural network sare introduced in effort estimation process mainly for the training and learning from previous data. The model identifies a positive correlation between the dependent (effort) and independent variables (effort drivers). The half of the available data sets can be given for training and the remaining can be used to derive effort. The other techniques of software effort estimation are bottom-up, top-down, analogy estimation and expert judgments.

II. Related Work

Cuauhtemoc [1] provides justification that Fuzzy logic can be used to predict the effort of the small programs based on lines of code obtained from new and changed (N&C) and reused code from small programs developed by 74 programmers. This was used as the input for the fuzzy model for estimating effort and the accuracy of output was compared with the accuracy of Statistical regression model using the comparison criterion Mean Magnitude Error Relative to the estimate.

Shiyna [2] compares the frameworks designed by using fuzzy logic and Neural Networks based on the accuracy of effort estimation. COCOMO NASA dataset had been used as the input for both the frameworks. These frameworks are validated using the parameters MMRE (Mean Magnitude of Relative Error) and Pred (Prediction Accuracy). The results show that Fuzzy Logic based framework works better when compared to the Neural Network framework.

Ochodek [3] proposed the usage of Use Case Points (UCP) method to estimate the effort based on the use case model and two adjustment factors (With or Without Unadjusted Actor Weights). The cross-validation procedure has been used to compare the variants of adjustment factors. A group of 14 projects is considered as input which are used to arrive at a conclusion that the UCP method can be simplified without the use of adjustment factors.

Iman [4] compares the software effort estimation computed by the conventional methods like function points, regression models and COCOMO with the model designed using fuzzy logic. The parameter Mean Magnitude of Relative Error (MMRE) is used to compute the accuracy of the considered methods.

Anjana Bawa [5] explains the usage of Artificial Neural Networks to estimate the project effort as it is capable of learning from the previous data. The machine learning algorithms, Back-Propagation and Cascade Correlation are used to learn and classify the dataset and hence estimate the effort using the Neural Networks.

By analyzing the previous work, it is evident that fuzzy logic is better than the conventional methods of effort estimation. By using the package points the complexity of estimating the lines of code for the considered software is reduced. By using the factor refinement, the time taken to compute the effort is less compared to the previous method where 15 attributes where obtained from the programmer to compute the effort.

III. Proposed Approach

We have considered 93 instances of NASA historical project data and also investigated and gathered thirty projects from many case studies and experiments [11][12][14][15] with consists of 15 attributes and actual effort along with domain, area of work, Size. The fifteen attributes art converted to three index valued labelled as Human Perception and Performance Index (HPPI), Machine Requirement and Performance Index (MRPI), Process Requirement and Performance Index (PRPI). Adaptive Neuro-Fuzzy model (The Figure 1) for software development effort estimation is perfect in the learning and good interpretability. Artificial neural networks are made up of processing units in a parallel manner called as neurons these neurons are inter linked by connections. The input for this model is six grouped attributes. Each attribute represents one factor which leads to the development effort. Table 1 describes the refinement of attributes in such a way that 15 effort multipliers are grouped in 3 clusters of refined attributes whose values are obtained from the software project developer.

Table 1: Refined Attributes

REFINED ATTRIBUTES	EFFORT MULTIPLIERS
Human Perception and Performance Index (HPPI)	1. Analyst Efficiency 2. Programmer Efficiency 3. Application Maturity 4. Modern Programming Practices 5. Use of Software tools 6. Virtual Machine Experience 7. Language Experience
Machine Requirement and Performance Index (MRPI)	8. Time Constraint for CPU 9. Turnaround Time 10. Machine Volatility
Process Requirement and Performance Index (PRPI)	11. Process Complexity 12. Storage Space Requirement 13. Schedule Constraint 14. Database Size 15. Required Software Reliability

This proposed model consolidates neural networks and fuzzy-logic principles in a combined ANFIS framework. This inference system correlates to the learning capability of fuzzy IF THEN rules to approximate the nonlinear functions.

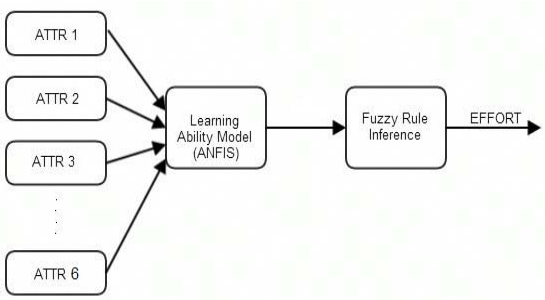

Figure 1 : Adaptive Neuro-Fuzzy model

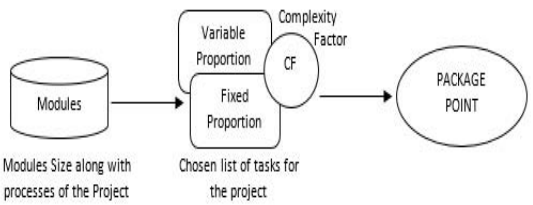

Figure 2 : Package Point Process

a) Package Points

The package point (figure 2) provides an alternate way to estimate the size that needs to be applied in a software project. Unlike function points and class points approach, the package points proves to be highly efficient in aiding to estimate effort and it is proven to work well with the ERP projects. In order to compute the package points, inputs are obtained

namely, scope, tasks and complexities. Package points are defined by standardizing the number of modules in the project primarily. Then the tasks to complete the modules are prioritized and defined by the client as per their requirement. Finally the complexity factors are loaded to arrive at the package point for the considered module.

b) Validation in ANFIS Model

ANFIS is a hybrid supervised method which adopts a hybrid learning algorithm to determine the parameters for fuzzy inference systems. It utilizes both least-squares method and propagation gradient descent method. This is used for training FIS membership function parameters to examine the given training data set. ANFIS can be executed using an optional argument for model validation. This is called as checking model for over fitting. The argument used for this is called as checking data set.

c) Fuzzy Rule Inference

Fuzzy rules are generated using package points, domain, type, Human Perception and Performance Index (HPPI), Machine Requirement and Performance Index (MRPI), Process Requirement and Performance Index (PRPI) and Actual effort. A fuzzy set is illustrated using a membership function that relates with every point in the fuzzy set that comprises of numbers in the interval [0, 1], known as degree or grade of membership. Membership function used in this research work is Triangular Membership Function. Triangular Fuzzy Number (TFN) is defined using a triplet (α, m, β), where m denotes modal value, α and βsignify the right and left boundary correspondingly and is expressed as:

$$\mu(x) = \begin{cases} 0, & x \le \alpha \\ \frac{x-\alpha}{m-\alpha}, & \alpha \le x \le m \\ \frac{\beta-x}{\beta-m}, & m \le x \le \beta \\ 0, & x \ge \beta \end{cases} \quad (1)$$

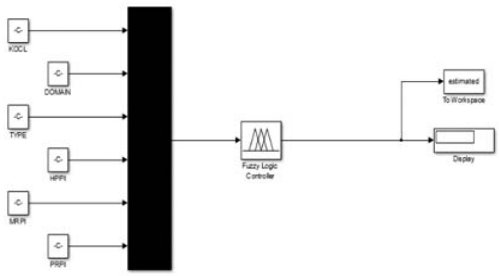

Figure 3 : Fuzzy Rules generated

The equivalent characteristics of the rules the most significant aspect of fuzzy logic systems. Instead of sharp swapping among modes based on breakpoints, logic flows efficiently from sections where the system's performance is governed by one rule or another.

Defuzzification converts from fuzzy to crisp conversions. The process converts the fuzzy value to the estimated value for the single data set. This is similar to a "rounding off" method. Defuzzification converts the collection of membership function data to a single sealer quantity in corresponding membership degrees. It is characterized instructure of rules that convert variables to a fuzzy result, that is, the outcome is defined in terms of membership in fuzzy sets.

$$R_{alv} = \begin{cases} \mu^* w_1 & 0 < \text{ALV} \le 1 \\ \mu^* w_1 + (1-\mu) * w_2 & 1 < \text{ALV} \le 2 \\ \mu^* w_1 + (1-\mu) * w_1 & 2 < \text{ALV} \le 3.5 \\ \mu^* w_2 + (1-\mu)\, w_3 & 3.5 < \text{ALV} \le 5 \\ \mu^* w_3 + (1-\mu)\, w_2 & 5 < \text{ALV} \le 6.5 \\ \mu^* w_3 + (1-\mu)\, w_4 & 6\,5 < \text{ALV} \le 8 \\ w_4 & \text{ALV} > 8 \end{cases} \quad (2)$$

IV. Experimental Design

a) Evaluation Criteria

1. Mean Magnitude Relative Error (MRE), is an error ratio between the absolute deviations of prediction to the actual effort in each of the referred project

$$MRE = |(Actual_i - Estimated_i)|/(Actual_i) \quad (3)$$

2. Mean Magnitude Relative Error (MMRE) is the average value of MER of all the referred projects

$$MMRE = 1/n \sum MREi \quad (4)$$

The Table 2 indicates the Package point and its subsequent domain with its actual effort and the non-algorithmic estimated effort using the proposed method. The effort in the dataset is compared with this estimated effort

Table 2 : Domain based Metrics

Area Of Domain	Package Point	Actual Effort	Estimated Effort
Avionics monitoring	25.9	117.6	138.4
Mission planning	31.5	60	112
Simulation	66.6	352.8	402
Monitor Control	70	458	561
Real Data Processing	177.9	124	397
Communications	240	192	322
Batch Data Processing	25.9	117.6	119
Data Capture	31.5	60	67.7
Launch Processing	66.6	352.8	360.9
Application Ground	70	458	459
Utility	177.9	124	128
Operating Systems	240	192	193.2

The preceding list of algorithmic models are tested including KLOC (Kilo Lines Of Code) value from the data sets and efforts were estimated and these estimates are compared with the results obtained from Adaptive Neuro Fuzzy model.

- Halstead Model- This model developed by Halstead, concerning the supplied lines of source code from the programmer and formulates a relation,

$$Effort = 0.7 * (KLOC)^{2} \quad (5)$$

- Bailey-Basili Model - This model developed by Bailey-Basili, between delivered lines of source code and formulates a relation,

$$Effort = 5.5 * (KLOC)^{1.16} \quad (6)$$

- Doty Model - This model developed by Doty, between delivered lines of source code and formulates a relation,

$$Effort = 5.288 * (KLOC)^{1.047} \quad (7)$$

- COCOMO - It was the first model suggested by Barry Boehm. This model has been widely accepted in practice. In the COCOMO model, the code-size S is given in thousand LOC (KLOC) and Effort is in person-month.

$$Effort = a * (KLOC)^{b} \quad (8)$$

Where a, b are complexity factors. This model uses three sets of a, b depending on the complexity of the software. The basic COCOMO model is simple and easy to use.

- COCOMO II - It comprises of three variants, namely, early design model, Application composition model, and Post architecture model. This is an augmentation of intermediate COCOMO model and defined as,

$$Effort = 2.9 * (KLOC)^{1.10} \quad (9)$$

b) Numerical Results

This section reports the experimental results of predictions obtained from soft computing models and other algorithmic models. The actual effort existing in the dataset is associated with the estimated effort and finally Mean Relative Error is calculated. The following graphs (figure 4) shows the Mean Relative Error variations in Doty, Bailey, COCOMO I and COCOMO II, Halstead and Neuro Fuzzy models.

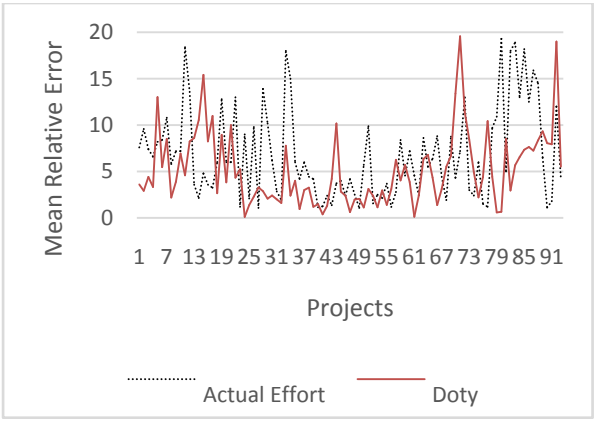

Figure 4 : Doty Estimation Vs Actual Effort

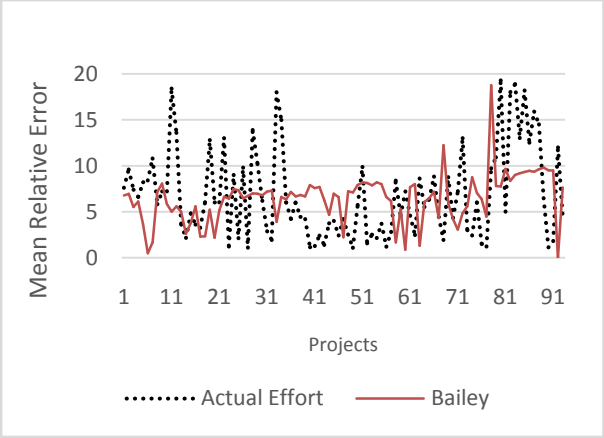

Figure 5 : Bailey Estimation Vs Actual Effort

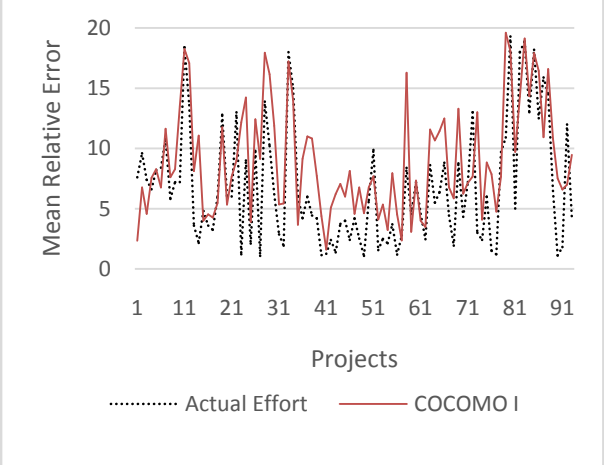

Figure 6 : COCOMO I Estimation Vs Actual Effort

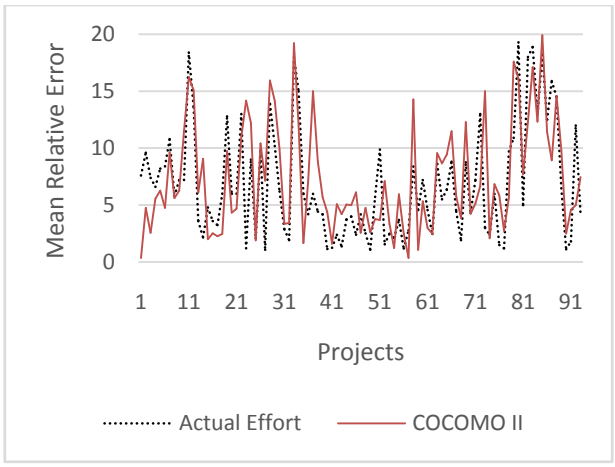

Figure 7 : COCOMO II Estimation Vs Actual Effort

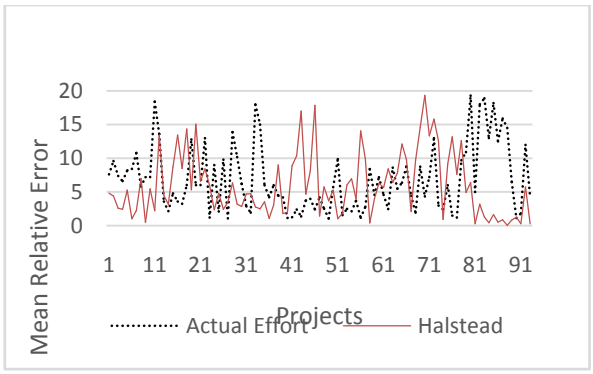

Figure 8 : Halstead Estimation Vs Actual Effort

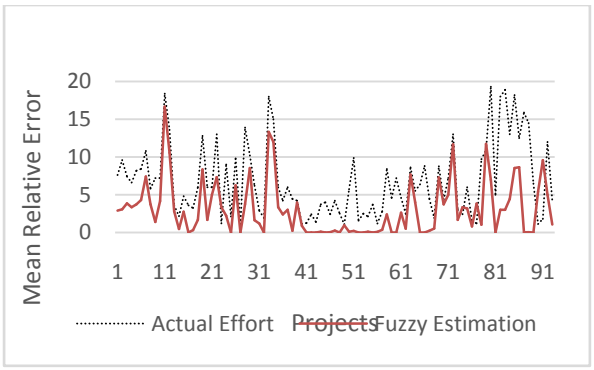

Figure 9 : Fuzzy Based Estimation Vs Actual Effort

The following table summarizes the MMRE value of all the algorithmic and non-algorithmic models discussed above for all the referred projects.

Table 3 : MMRE comparison

	Doty	Bailey	Halstead	COCOMO I	COCOMO II	Fuzzy Estimation
MMRE	5.15	6.47	6.06	9.14	7.54	3.11

V. Conclusion

Early effort estimation in software development lifecycle is an important activity for project planning and resource allocation. This research work proposes an efficient model in estimating the software effort. The outcomes of the estimation obtained using the direct algorithmic methods indicates the divergence between the actual and the estimated effort. The outcome of non-algorithmic method comprising of the adaptive neuro technique based estimation decreases the Mean Magnitude of Relative Error (MMRE). Hence the examination of effort from algorithmic method and non-algorithmic method prove that adaptive neuro fuzzy based estimation is more efficient than the algorithmic methods for the estimation process. The success of estimation depends upon the accuracy and stability of the method in various measures. Future work is planned to investigate the clustering algorithms in estimation process and apply Neuro Fuzzy model on large datasets.

References Références Referencias

1. Cuauhtemoc 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, 2010.
2. Shiyna Kumar, Vinay Chopra, "Neural Network and Fuzzy Logic based framework for Software Development Effort Estimation", International Journal of Advanced Research in Computer Science and Software Engineering, Vol.3, No. 5, 2013.
3. M. Ochodek, J.Nawrocki and K.Kwarciak, "Simplifying effort estimation based on Use Case Points", Information and Software Technology, Vol.53, 2011.
4. Iman Attarzadeh and Siew Hock Ow, "Software Development Effort Estimation Based on a New Fuzzy Logic Model", International Journal of Computer Theory and Engineering, Vol. 1, No. 4, 2009.
5. Anjana Bawa and Rama Chawla, "Experimental Analysis of Effort Estimation Using Artificial Neural

Network", International Journal of Electronics and Computer Science Engineering, Vol.1, No.3, 2011.
6. Mohammad Azzeh, Daniel Neagu and Peter I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers", The Journal of Systems and Software, Vol.84, 2011.
7. Sun-Jen Huang, Nan-Hsing Chiu and Li-Wei Chen, "Integration of the grey relational analysis with genetic algorithm for software effort estimation", European Journal of Operational Research, Vol.188, 2007.
8. M. Kazemifard, A. Zaeri, N. Ghasem-Aghaee, M.A. Nematbakhsh and F. Mardukhi , "Fuzzy Emotional COCOMO II Software Cost Estimation (FECSCE) using Multi-Agent Systems", Applied Soft Computing, Vol.11, 2011.
9. Emad A. El-Sebakhy, "Functional networks as a novel data mining paradigm in forecasting software development efforts", Expert Systems with Applications, Vol.38, 2011.
10. Magne Jørgensen, "Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation", Information and Software Technology, Vol.53, 2011.
11. Bente Anda, Endre Angelvik, and Kirsten Ribu,"Improving Estimation Practices by Applying Use Case Models", product focused software process improvement, Springer,2002. Kirsten Ribu, "Estimating Object-Oriented Software Projects with Use Cases, University of Oslo. 2001
12. AP Subriadi, Sholiq, A P Ningrum. "Critical review of the effort rate value in use case point method for estimating software development effort" Journal of Theoretical and Applied Information Technology, January 2014
13. M. Azzeha, Ali Bou Nassifb, Leandro L. Minkuc, "An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation" Journal of Systems and Software, may 2015
14. A.B Nassif, L.F. Capretz, "Software Effort Estimation in the Early Stages of the Software Life Cycle Using a Cascade Correlation Neural Network Model" 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012
15. Nassif, A.B.; Capretz, L.F.; Ho, D. "Estimating Software Effort Using an ANN Model Based on Use Case Points", Machine Learning and Applications (ICMLA), 2012 11th International Conference on, On page(s): 42 - 47 Volume: 2, 12-15 Dec. 2012
16. El Bajta, M. "Analogy-Based Software Development Effort Estimation in Global Software Development", Global Software Engineering Workshops (ICGSEW), 2015 IEEE 10th International Conference on, on page(s): 51 – 54.