22

Estimating Development Time and Effort of Software Projects by using a Neuro_Fuzzy Approach

Venus Marza[1] and Mohammad Teshnehlab[2]
Islamic Azad University South Tehran Branch Faculty of Technical and Engineering[1], Department of Electrical and Computer Engineering of Khaje Nasir Toosi[2] Iran, Tehran

1. Introduction

As software becomes more complex and its scope dramatically increase, the importance of research on developing methods for estimating software development effort has perpetually increased. Estimating the amount of effort required for developing a software system is an important project management concern, because these estimation is a basic for budgeting and project planning, which are critical for software industry. However accurate software estimation is critical for project success. So many software models have been proposed for software effort estimation. Algorithmic models such as COCOMO, SLIM, Multiple Regression, Statistical models,… and non-algorithmic models such as Neural Network Models (NN), Fuzzy Logic Models, Case-Base Reasoning (CBR), Regression Trees,… are some of these models. Here we want to improve software accuracy by integrating the advantages of algorithmic and non-algorithmic models. Also, recent research has tended to focus on the use of function point (FP) in estimating the software development efforts, but a precise estimation should not only consider the FPs, which represent size of the software, but also should include various elements of the development environment which affected on effort estimation. Consequently, for software development effort estimation by Neuro-Fuzzy approach, we will use of all the significant factors on software effort. So the final results are very accurate and reliable when they are applied to a real dataset in a software project.

The empirical validation uses the International Software Benchmarking Standards Group (ISBSG) Data Repository Version 10 to demonstrate the improvement of results. This dataset contains information on 4106 projects of which two thirds were developed between the years 2000 and 2007. The evaluation criteria were based mainly upon MMRE (Mean Magnitude Relative Error), MMER and PRED(20). The results show a slightly better predictive accuracy amongst Fuzzy Logic Models, Neural Network Models, Multiple Regression Models and Statistical Models.

This chapter of book is organized into several sections as follows: In section 1, we briefly review fuzzy logic models and neural network models in software estimation domain.

Section 2 begins with preparing the dataset and this is followed by description of our proposed model. The experimental results are examined in Section 3 in details, and finally Section 4 offers conclusions and recommendations for future research in this area.

1. Survey of fuzzy logic and neural network models

1.1 Fuzzy Logic Model

Since fuzzy logic foundation by Zadeh in 1965, it has been the subject of important investigations [Idri & Abran, 2001]. Fuzzy logic enhances the user's ability to interpret the model, allowing the user to view, evaluate, criticize and possibly adapt the model. Prediction can be explained through a series of rules [Gray & MacDonell, 1997],[Saliu et al., 2004]. After analyzing the fuzzy logic model, experts can check the model to avoid the adverse effects of unusual data, thereby increasing its robustness. Additionally, fuzzy logic models can be easily understood in comparison to regression models and the neural network, thus making it an effective communication tool for management [MacDonell et al., 1999],[Gray & MacDonell , 1999]. In comparison to fuzzy logic, case-based reasoning is similarly easy to interpret, but it requires a high volume of data [Su et al., 2007].

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 chapter 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. The number of fuzzy rules for six input variables and three membership functions is calculated by 3^6, which equals 729. As a result, writing these rules is an arduous task, so based on the statistical model we use two input variables which are demonstrated later. 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 [Moataz et al., 2005].

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 pa rameters (a, b, c) are defined as follows:

MF(x) = 0 if x < a
MF(x) = 1 if x = b
MF(x) = 0 if x > c

Based on the Correlation (r) of the variables, fuzzy rules can be formulated. Correlation, the degree to which two sets of data are related, varies from -1.0 to 1.0. The Correlation Coefficient for the input variables is calculated from the equation below [Humphrey, 2002]:

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

An acceptable correlation should have an absolute value higher than 0.5. The fuzzy inference process uses the Mamdani Approach for evaluating each variable complexity degree when linguistic terms, fuzzy sets, and fuzzy rules are defined. Specifically, we apply the minimum method to evaluate the 'and' operation, and consequently, we obtain one number that represents the antecedent result for that rule. The antecedent result, as a single number, creates the consequence using the minimum implication method. Overall, each rule is applied in the implication process and produces one result. The aggregation using the maximum method is processed to combine all consequences from all the rules and produces one fuzzy set as the output. Finally, the output fuzzy set is defuzzified to a crisp single number using the centroid calculation method [Xia et al., 2007]. This Two-Input-One-Output fuzzy logic system for Effort is depicted in Figure 1. Moreover, the results of this model are shown in Table 7 and Table 9.

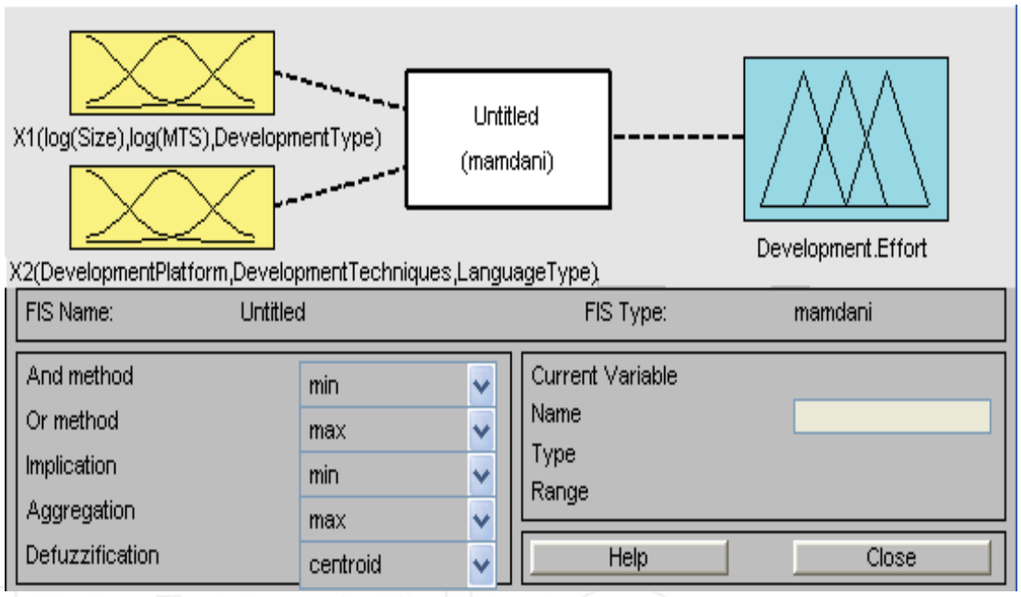

Fig. 1. The Fuzzy Logic System for Effort Estimation

1.2 Neural Network Model

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 [Su et al., 2007]. 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 [Aggarwal et al., 2005] [Huang et al.,2007].

When looking at a neural network, it immediately comes to mind that activation functions are look like fuzzy membership function [Jantzen, 1998].

Our neural network model uses an RBF network, which is easier to train than an MLP network. The RBF network is structured similarly to the MLP in that it is a multilayer, feed-forward network. However, unlike the MLP, the hidden units in the RBF are different from

the units in the input and output layers. Specifically, they contain the RBF, a statistical transformation based on a Gaussian distribution from which the neural network's name is derived [Heiat, 2002]. Since the data of our variables differs significantly, first, we normalized the data and then randomly divided them into two categories: 75% of projects are used for training and 25% of them are used for testing. The trajectory of the training phase is depicted in Figure 2. In particular, we used the Generalized Regression Neural Network Model in MATLAB 7.6, RBF network was created and the data set was applied to it; the results are shown in Table 7-9.

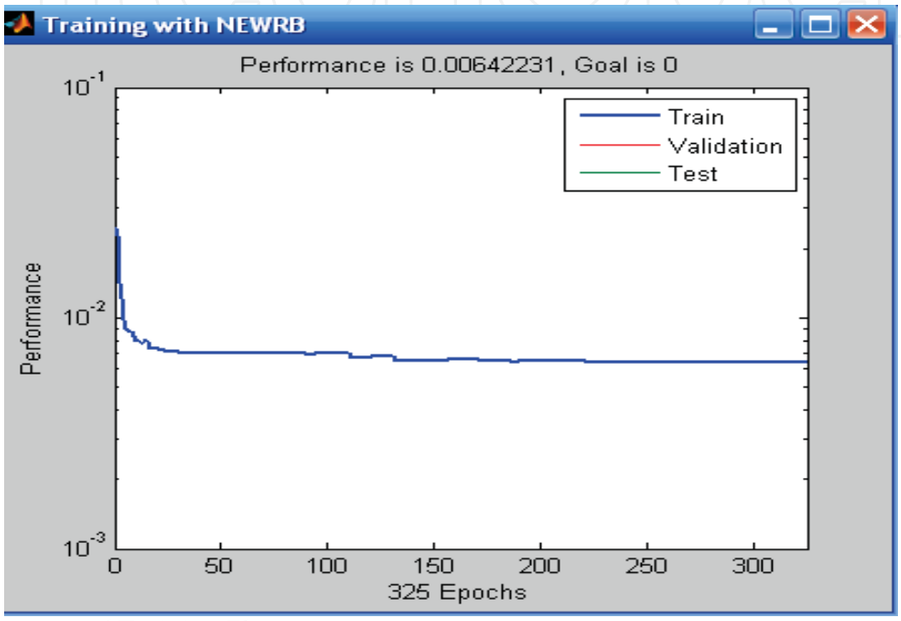

Fig. 2. Progress of Training Phase

2. Proposed Model:

2.1 choosing a Neuro-Fuzzy Model for estimation

By comparison between artificial neural networks (ANN) and fuzzy inference systems (FIS), we find that neural network difficult to use prior rule knowledge, learning from scratch, they have complicated learning algorithms and they are black box structure and also they difficult extract knowledge while fuzzy inference systems can incorporate prior rule-base, they are interpretable by if-then rules, they have simple interpretation and implementation but they can't learn linguistic knowledge and knowledge must be available. Therefore, it seems natural to consider building an integrated system combining the concepts of FIS and ANN modeling. A common way to integrate them is to represent them in a special architecture. Different integrated neuro-fuzzy models implement a Mamdani and Takagi Sugeno fuzzy inference systems, some of them are FALCON, ANFIS, NEFCON, NEFCLASS, NEFPROX, FUN, SONFIN, EFuNN, dmEFuNN and many others [Abraham, 2005].

Due to unavailability of source codes, we are unable to provide a comparison with all the models. In general Takagi-Sugeno fuzzy system has lower Root Mean Square Error (RMSE)

than Mamdani-type fuzzy system but Mamdani fuzzy systems are much faster in compared to Takagi-Sugeno types, our purpose is accuracy so we didn't consider mamdani-type fuzzy system such as FALCON, NEFCON, NEFCLASS, EFuNN. Since no formal neural network learning technique is used in FUN and it randomly changes parameters of membership functions and connections within the network structure, therefore we don't consider it as a neuro-fuzzy system. About other models, Mackey & Glass [Mackey & Glass, 1977] provided a comparative performance of some neuro fuzzy systems for predicting the Mackey-Glass chaotic time series that represented in table 1.

System	Epochs	Test RMSE
ANFIS	**75**	**0.0017**
NEFPROX	**216**	**0.0332**
EFuNN	**1**	**0.0140**
dmEFuNN	**1**	**0.0042**
SONFIN	**1**	**0.0180**

Table 1. Performance of neuro-fuzzy systems

As shown in table ANFIS has the lowest RMSE in compared to NEFPROX (highest RMSE), SONFIN and dmEFuNN which used Takagi-Sugeno fuzzy system. So we use ANFIS as neuro-fuzzy model for predicting effort of software projects.

2.2 Preparing Dataset

In this study we used the latest publication of ISBSG (International Software Benchmarking Standards Group) data repository Release 10 that contains 4106 project's information and two thirds of them were developed between the years 2000 and 2007. One hundred seven metrics were described for each project including data quality rating, project size, work effort, project elapsed time, development type, development techniques, language type, development platform, methodology, max team size,… .

The ISBSG data repository includes an important metric as Data Quality Rating which indicated that the reliability of the reported data. We excluded 141 projects with quality rating D which had little credibility. Project size is recorded with function points and homogeneity of standardized methodologies is very essential for measuring function size. Among different count approaches of function point NESMA is considered to produce equivalent results with IFPUG [NESMA 1996] and most of projects used these approaches for counting function points. So for giving more reliable results, projects with other counting approaches were excluded from the analysis. Also some projects had mistakenly information for example they had 0.5 or 0.95 for Average Team Size or Development Platform was recorded by 'HH' where not acceptable. Finally after cleaning data, 3322 projects remained for predicting effort's projects.

2.3 Suggested model

Our study is based on statistical regression analysis, which is the most widely used approach for the estimation of software development effort. Now we briefly introduce the variables in data repository which will be used as the predicator for the regression analysis [Zhizhong et al.[a], 2007]:

1. Functional Size: It gives the size of the project which was measured in function points.
2. Average Team Size: It is the average number of people that worked on the project through the entire development process.
3. Language Type: It defines the language type used for the project such as 2GL, 3GL, 4GL and ApG. 2GL (two generation languages) are machine dependent assembly languages, 3GL are high-level programming languages like FORTRAN, C ,etc. 4GL like SQL is more advanced than traditional high-level programming languages and ApG (Application Generator) is the program that allows programmers to build an application without writing the extensive code.
4. Development Type: Describes whether the software development was a new development, enhancement or Re-development.
5. Development Platform: Defines the primary development platform. Each project was developed for one of the platforms as midrange, mainframe, multi-platform, or personal computer.
6. Development Techniques: Specific techniques used during software development (e.g. Waterfall, Prototyping, Data Modeling, RAD, etc). A large number of projects make use of various combined techniques.
7. Case Tool Used: Indicates if the project used any CASE (Computer-Aided Software Engineering) tool or not.
8. How Methodology Acquired: Describes whether the development methodology was traditional, purchased, developed in-house, or a combination of purchased and developed.

It is important to point out that [Zhizhong et al [b]., 2007]:

* We did not take into account the factor primary programming language, since each particular programming language (Java, C, etc) belongs to one of the generation languages (2GL, 3GL, etc).
* It is conceivable that senior software developers are more proficient and productive than junior developers. ISBSG data repository does not report this and assumes the developers are all well-qualified practitioners.
* When considering the factor Development Techniques, there exist over 30 different techniques in the data repository and 766 projects even used various combinations of these techniques. Our study considered the ten key development techniques (Waterfall, Prototyping, Data Modeling, Process Modeling, JAD or Joint Application Development, Regression Testing, OO or Object Oriented Analysis & Design, Business Area Modeling, RAD or Rapid Application Development) and separated each of them as one single binary variable with two levels that indicates that whether this variable was used (1) or not (0), also other combinations were labeled by 'Other' as development factor technique.
* The variables Effort, Size and Average Team Size are measured in ratio scales while all others are measured in nominal scales.

Here by fitting a model with Effort as the dependent variable and all the other variables as the predicators, we reduced our inputs for prediction, because for ANFIS with Genfis1 implementation is impossible to write all the rules and the complexity of model will be

increased. So Regression Analysis helps us to use variables effectively. Table 2 gives the summary of the variables used for the regression analysis.

Variables		Scale	Description
Effort		Ratio	Summary Work Effort
Size		Ratio	Functional Size
Average Team Size		Ratio	Average Team Size
Language Type		Nominal	Language Type
Development Type		Nominal	Development Type
Development Platform		Nominal	Development Platform
CASE Tool		Nominal	CASE Tools Used
Methodology		Nominal	How Methodology Acquired
Development Techniques	Waterfall	Nominal	1=Waterfall , 0=Not
	Data	Nominal	1=Data Modelling , 0=Not
	Process	Nominal	1=Process Modelling , 0=Not
	JAD	Nominal	1=JAD , 0=Not
	Regression	Nominal	1=Regression Testing , 0=Not
	Prototyping	Nominal	1=Prototyping , 0=Not
	Business	Nominal	1=Business Area Modelling , 0=Not
	RAD	Nominal	1=RAD , 0=Not
	O.O	Nominal	1=Object Oriented Analysis, 0=Not
	Event	Nominal	1=Event Modelling , 0=Not
Other factors		Nominal	1=Uncommon Development Techniques, 0=Not
Missing		Nominal	1=Missing , 0=Not

Table 2. Summary of the variables for Regression

The variable Missing was added as an indicator variable and indicate that the use of development techniques was recorded for particular project or not (1=recorded, 0=missing). The first step is automatic model selection based on Akaike's information criterion (AIC). AIC is a measure of the goodness of fit of an estimated statistical model. Given the assumption of normally-distributed model errors, AIC is given as [Venables & Ripley, 2002]:

$$AIC = n \log(RSS/n) + 2p \quad (2)$$

Here n is the number of observations, RSS is Residual Sum of Squares, and p is the number of parameters to be estimated. AIC has a penalty as a function of the number of estimated parameters because increasing the number of parameters improves goodness of fit (small RSS), so the preferred model is the one with the lowest AIC value. Based on this criterion, the preferred model with the lowest AIC value is introduced in Table 3.

It is important to point out here that since the original data of Effort and Average Team Size also Effort and Size are extremely skewed, we take the natural log transformation (with base e) to make the data look normally distributed. In scatter plot between each two variables we

can demonstrate that the relationship between them is close to linear. Accordingly we can apply linear model to investigate them.

Regression Terms	Df	Sum of Square	AIC (if variable excluded)
Log(Size)	1	140.6	-161.7
Log(Team Size)	1	134.4	-170.4
Language Type	3	22.2	-357.5
Development Type	2	14.2	-371.1
Development Platform	3	13.8	-373.8
Other	1	1.9	-393.7
RAD	1	1.2	-395.1

Table 3. Regression Results Based on AIC
(The lowest value of AIC is -395.1)

As regression based on AIC tends to overestimate the number of parameters when the sample size is large [Venables & Ripley, 2002], rely fully on the results produced by AIC is not suitable. So AIC should be combined with other statistical criterion such as ANOVA (ANalysis Of VAriance), here we used the ANOVA approach (based on Type I Sums of Squares) to test the significance of the variables. The variables added into the model in order and according to Table 3, the exclusion of the variable size results in the greatest increase of AIC value. Thus the project size factor is most significant to development effort likewise average team size is the second most important factor and etc. Based on Table 3 we can add the variable size to the regression model first, average team size, language type and so forth, then each time the regression was performed, the most insignificant variable was removed and then the model was refitted with the remained variables. By continuing this process we have the model with the final sets of significant terms where represented in Table 4 and significance level is based on p-value <0.05.

Regression Terms	Df	Sum of Sq	F-Value	P-Value
Log(Size)	1	497.8	1026.2	$<10^{-15}$
Log(Average Team Size)	1	173.7	358.1	$<10^{-15}$
Language Type	3	35.9	24.7	$4.8 * 10^{-15}$
Development Platform	3	16.3	11.2	$3.8 * 10^{-7}$
Development Type	2	13.5	13.9	$1.3*10^{-6}$
RAD	1	2.7	5.5	0.019
Other	1	3.9	8.1	$4.6*10^{-3}$
Residuals	573	277.9		

Table 4. ANOVA based on Type I Sums of Squares
(The significance level is based on P-level < 0.05)

By comparing Table 2 and Table 3, we can see that the two methods produced similar significant factors for development effort, although the model based on AIC statistics overestimated additional two variables (OO and Missing) as significant. Considering that

AIC tends to overestimate the number of parameters when the sample size is large, we accept the second model as most appropriate for our study. Summary of the regression results are shown in Table 5.

Regression Terms	Coefficients	Standard Error	P-Value
Intercept	4.24	0.30	$<10^{-15}$
Log(Size)	0.56	0.03	$<10^{-15}$
Log(Average Team Size)	0.68	0.04	$<10^{-15}$
3GL's Language	-0.40	0.27	0.136
4GL's Language	-0.85	0.27	0.002
ApG's Language	-0.71	0.29	0.014
Midrange Platform	-0.12	0.08	0.116
Multi-Platform	-0.15	0.17	0.379
PC Platform	-0.46	0.08	$3.3*10^{-8}$
New Development Type	0.29	0.07	$1.6*10^{-5}$
Re-development Type	0.56	0.15	$2.4*10^{-4}$
RAD	-0.23	0.11	0.027
Other	-0.27	0.09	0.005

Table 5. Summary of the Regression results

It's important to point that the default Language Type is 2GL, the default Development Platform is Mainframe, and the default Development Type is Enhancement. According to Table 5, the model is fitted as (the variable 'Other' is not useful and not included):

$$\begin{aligned}&\log(Effort) = 4.24 + 0.56 * \log(Size) + 0.68 * \log(TeamSize) + \\ &\alpha_i \varphi(Language_i) + \beta_j \varphi(Platform_j) + \gamma_k \varphi(DevType_k) \\ &-0.23\varphi(RAD)\end{aligned} \tag{3}$$

i=1, 2, 3, 4; j=1, 2, 3, 4; k=1, 2, 3

Here the function Φ is the indicator function with binary values of 1 or 0. A value of 1 means the relevant development technique in the parentheses is used, otherwise the value is 0. So the default techniques used are: 2GL for language type (α_1=0), Mainframe for development platform (β_1=0), and Enhancement for development type (γ_1=0). The coefficients α_i, β_j, and γ_k can be obtained from Table5.
By using the obtained coefficient, we assign a value to each variable in our database and these values are corresponding to these coefficients which are shown in Table5.
Our purpose was to apply ANFIS to prepared ISBSG database. Before using ANFIS, we need to have an initial FIS (Fuzzy Inference System) that determines the number of rules and initial parameters, etc. This can be done in three different ways: by using five of the GUI tools, by using Genfis1 that generates grid partition of the input space, and by using Genfis2 that employs subtractive clustering. In other words, if we have a human expert, we can use GUI tools to convert human expertise into rough correctly fuzzy rules, which are then fine-tuned by ANFIS. If we don't have human experts, then we have to use some heuristics embedded in Genfis1 or Genfis2 to find the initial FIS and then go through the same ANFIS

tuning stage. The question is that which of Genfis1 or Genfis2 should be used to generate the FIS matrix for ANFIS, and the answer is when you have less than six inputs and a large size of training data, use Genfis1 and otherwise use Genfis2. GENFIS1 uses the grid partitioning and it generates rules by enumerating all possible combinations of membership functions of all inputs; this leads to an exponential explosion even when the number of inputs is moderately large. For instance, for a fuzzy inference system with 10 inputs, each with two membership functions, the grid partitioning leads to 1024 ($=2^{10}$) rules, which is inhibitive large for any practical learning methods. The "curse of dimensionality" refers to such situation where the number of fuzzy rules, when the grid partitioning is used, increases exponentially with the number of input variables. However, GENFIS1 and GENFIS2 differ in two aspects. First, GENFIS1 produces grid partitioning of the input space and thus is more likely to have the problem of the ``curse of dimensionality'' described above, while GENFIS2 uses SUBCLUST (subtractive clustering) to produces scattering partition. Secondly, GENFIS1 produce a fuzzy inference system where each rule has zero coefficients in its output equation, while GENFIS2 applies the backslash ("\") command in MATLAB to identify the coefficients. Therefore the fuzzy inference system generated by GENFIS1 always needs subsequent optimization by ANFIS command, while the one generated by GENFIS2 can sometimes have a good input-output mapping precision already. Any way since we have six inputs, Genfis2 and then ANFIS is used for our implementation. Also we divided our inputs in two categories and then Genfis1 was used for implementation because we want to compare our results with Fuzzy Model and this model is impossible to implement with six inputs because of its' exponential rules. The other way for preparing FIS for ANFIS is using Genfis3 and its' difference with Genfis2 is that Genfis3 use Fuzzy C-Means Clustering for Clustering inputs data and since our results are almost the same, we have arbitrarily used Genfis2.

For implementation with two inputs, as we say we should divide our six inputs in two categories:

1. Inputs which have the Ratio Scale such as Log(Size) and Log(Average Team Size) given as:

$$Input1 = 4.24 + 0.56\log(Size) + 0.68\log(Average\ Team\ Size)$$

2. Inputs which have the Nominal Scale such as Language Type, Development Platform, Development Type and RAD

$$Input2 = \alpha_i \varphi(Language_i) + \beta_j \varphi(Platform_j) + \gamma_k \varphi(DevType_k) - 0.23\varphi(RAD)$$

The ANFIS structure with six and two inputs are shown in Figure 3 and Figure 4 respectively.

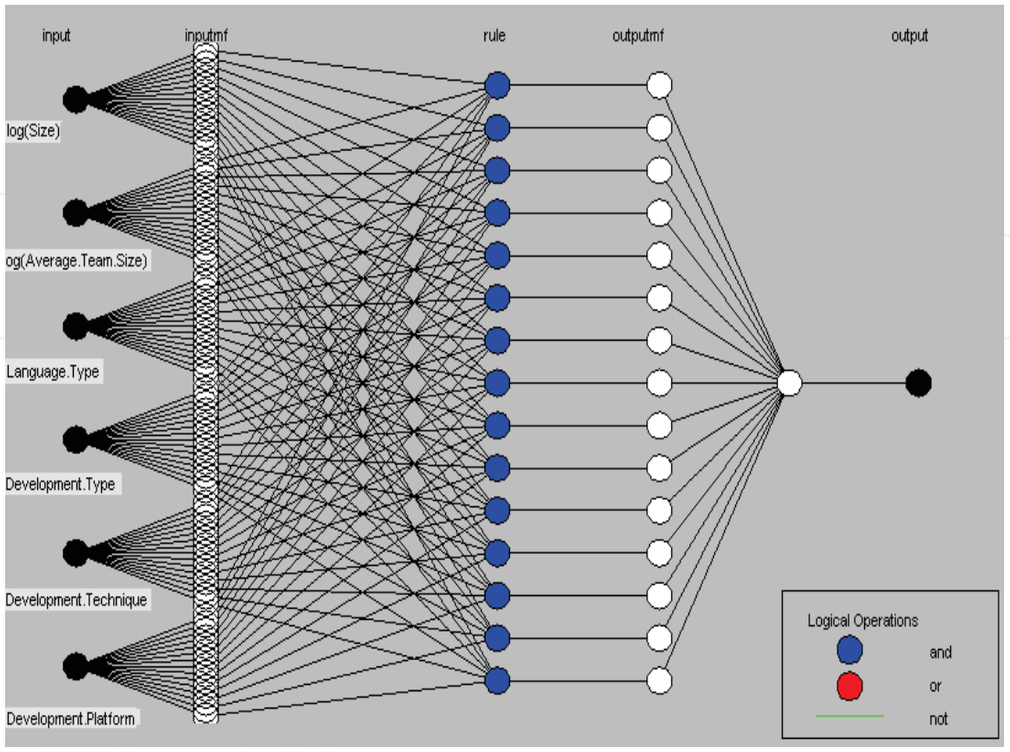

Fig. 3. ANFIS Structure for Six Inputs

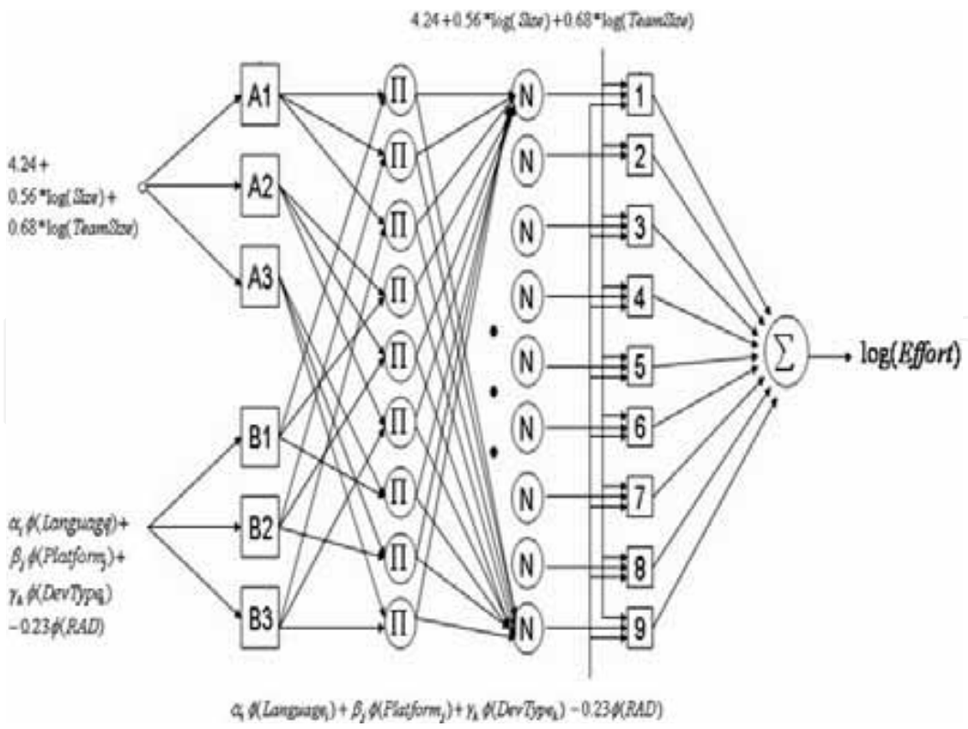

Fig. 4. two-inputs Type III ANFIS with 9 Rules

These inputs and structures are for estimating effort of software projects, but for the elapsed time of software project studies shows that two inputs of log(Effort) and log(Average Team Size) are sufficient for estimating. So by using Genfis1, the subspace of ANFIS Structure is as shown in Figure 5.

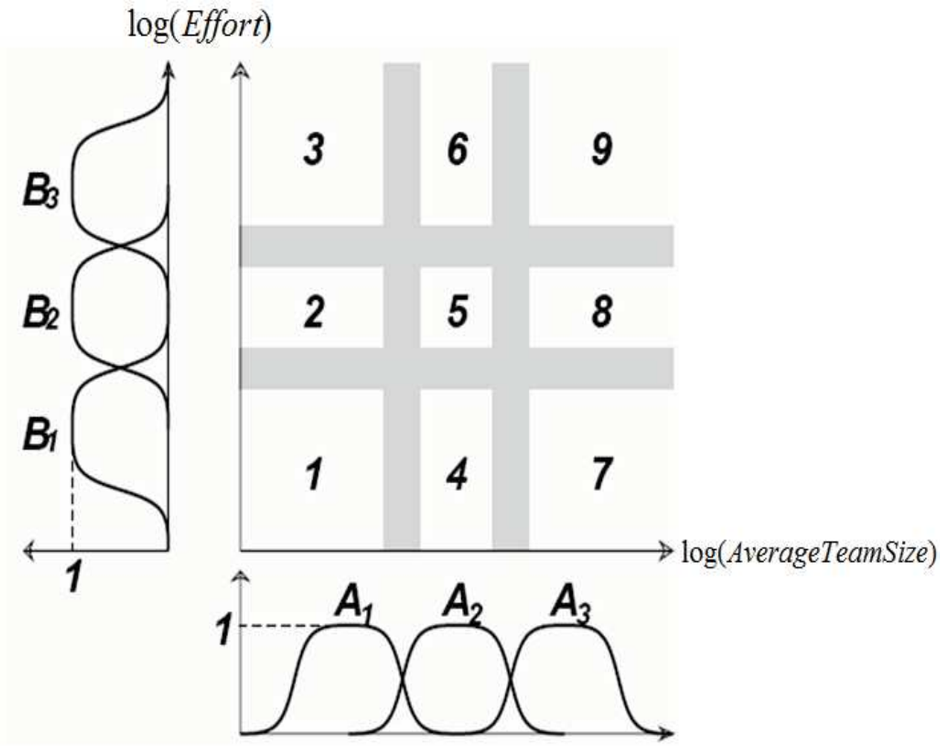

Fig. 5. corresponding fuzzy subspaces with two inputs for time estimation

ANFIS uses a hybrid learning algorithm to identify parameters of Sugeno-type fuzzy inference systems. It applies a combination of the least-squares method and the back-propagation gradient descent method for training FIS membership function parameters to emulate a given training data set. More specifically, in the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. Hybrid learning rule can speed up the learning process and has less error than gradient descent method. Table 6 summarizes the activities in each pass.

-	Forward Pass	Backward Pass
Premise Parameters	fixed	gradient descent
Consequent Parameters	Least Squares Estimate	Fixed
Signals	Node outputs	Error rates

Table 6. Two passes in the hybrid learning procedure for ANFIS

In Figure 6 we demonstrate the membership functions of FIS for time estimation, and Figure 7 shows an output of ANFIS for Time Estimation.

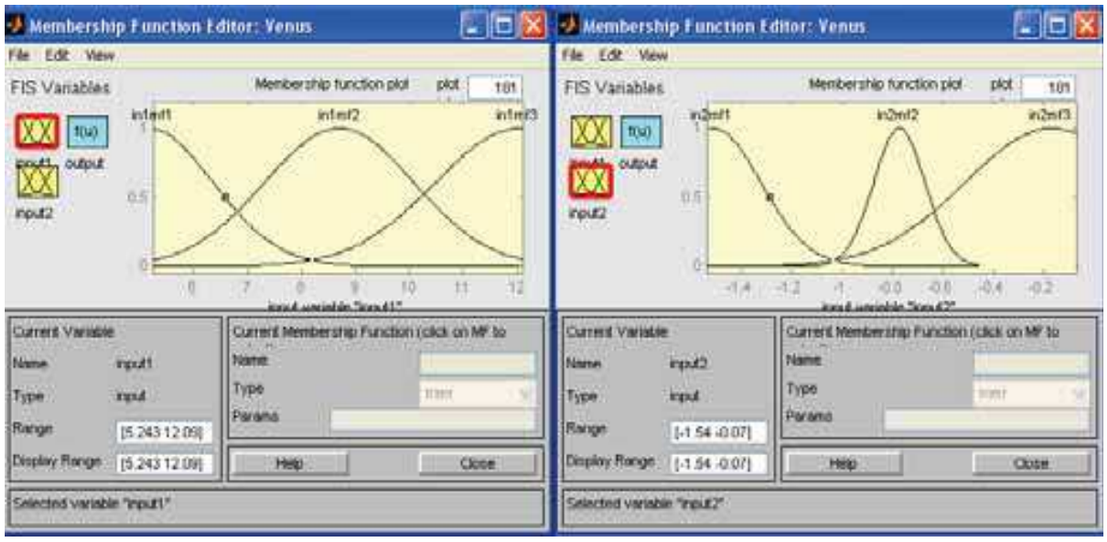

Fig. 6. membership function of FIS for Time Estimation after ANFIS's Training

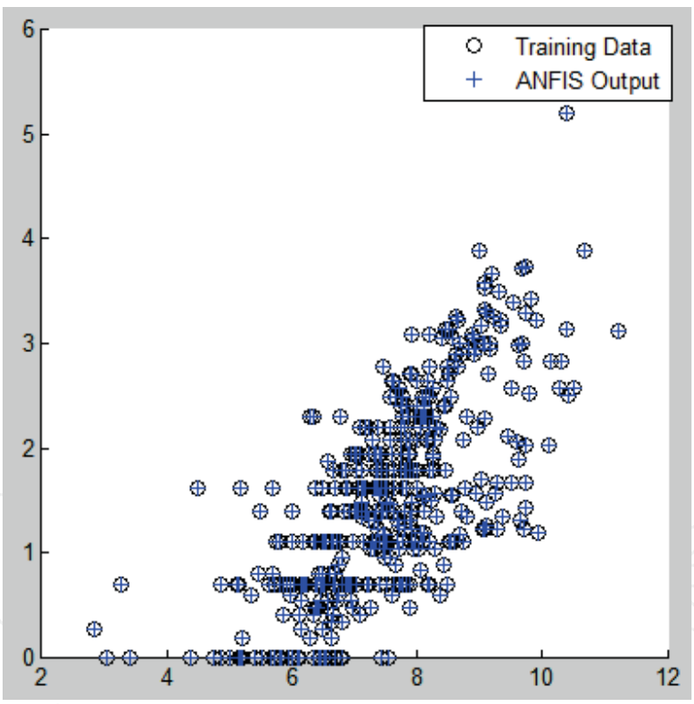

Fig. 7. ANFIS output for Time Estimation

3. Experimental Results

3.1 Evaluation Criteria

We employ the following criteria to assess and compare the performance of effort estimation models. A common criterion for the evaluation of effort estimation models is the relative error (RE) or the magnitude of relative error (MRE), which is defined as [Huang et al., 2007]:

$$RE_i = \frac{Actual\ Effort_i - \Pr edicted\ Effort_i}{Actual\ Effort_i} \quad (4)$$

$$MRE_i = \frac{|Actual\ Effort_i - \Pr edicted\ Effort_i|}{Actual\ Effort_i} \quad (5)$$

The RE and MRE values are calculated for each project i whose effort is predicted. For N multiple projects, we can also use the mean magnitude of relative error (MMRE) [Huang et al., 2007]:

$$MMRE_i = \frac{1}{N}\sum_{i=1}^{N}\frac{|Actual\ Effort_i - \Pr edicted\ Effort_i|}{Actual\ Effort_i} = \frac{1}{N}\sum_{i=1}^{N} MRE_i \quad (6)$$

Intuitively, MER seems preferable to MRE since MER measures the error relative to the estimate. Here we used this. The MER is defined as follows [Lopez-Martin et al., 2008]:

$$MER_i = \frac{|Actual\ Effort_i - \Pr edicted\ Effort_i|}{\Pr edicted\ Effort_i} \quad (7)$$

The MER value is calculated for each observation i whose effort is predicted. The aggregation of MER over multiple observations (N) can be achieved through the mean MER (MMER) as follows [Lopez-Martin et al., 2008]:

$$MMER = \frac{1}{N}\sum_{i=1}^{N} MER_i \quad (8)$$

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

$$\Pr ed(p) = \frac{k}{N} \quad (9)$$

Where k is the number of projects where MRE is less than or equal to p. here we used Pred(25).
In general, the accuracy of an estimation technique is Proportional to Pred(p) and inversely proportional to MMRE and MMER. Any way we used all of these criterions for evaluation of software techniques.
Also the other criterion is coefficient of determination (R^2). Coefficient of determination is used to assess the quality of the estimation models and expressed by R^2. The coefficient R^2 is calculated by [Gu et al., 2006]:

$$R^2 = \frac{\sum_{i=1}^{n}(y - \bar{y})^2}{\sum_{i=1}^{n}(\hat{y} - \bar{y})^2} \quad (10)$$

Here, $\bar{y}$ expresses the mean value of random variables. Obviously, the coefficient R^2 describes the percentage of variability and the value is between 0 and 1; when an R^2 is close to 1, it indicates that this model can explain variability in the response to the predictive variable, i.e. there is a strong relationship between the independent and dependent variables.

3.2 Implementation Results

A software tool (MATLAB 7.6) was used to simulate fuzzy logic system, neural network model and neuro-fuzzy model. Three categories of results are as below:

- First category: Effort Estimation with two inputs data as we discussed above. The results are gathered in Table 7 which showed that Neuro-Fuzzy model has 96% data with less than 20% error. As shown in Figure 8 just four data had more than 25% error and most of them had less than 7% error.
  Since we just have two inputs, we implement ANFIS by Genfis1.

Estimation Models	Pred(20)	Average Error	MMRE	MMER
Neuro-Fuzzy Model	0.96	0.40	0.05	0.05
Fuzzy Logic Model	0.89	0.91	0.12	0.13
Neural Network Model	0.88	0.39	0.04	0.07
Multiple Regression Model	0.78	0.90	0.12	0.14

Table 7. Implementation Results for Effort Estimation with two inputs

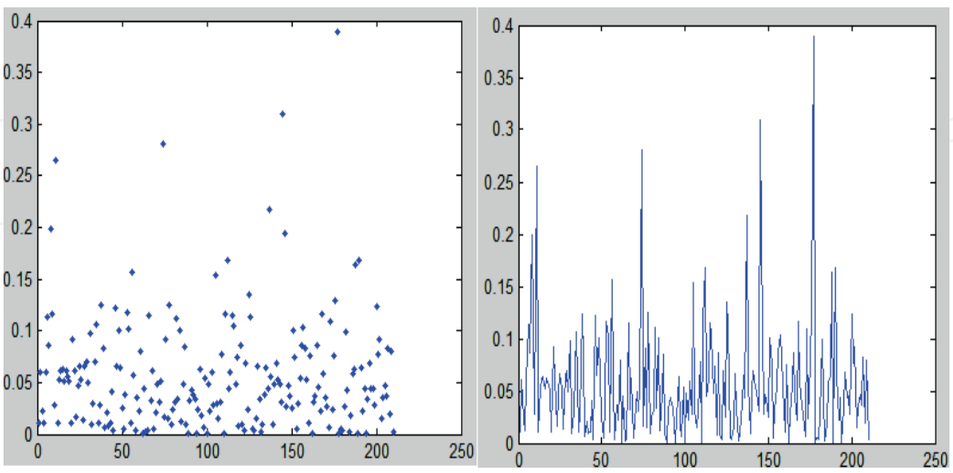

Fig. 8. MMRE Results for ANFIS with two inputs for Effort Estimation

- Second Category: due to the number of inputs we implement ANFIS by Genfis2 here. As we mentioned before the Fuzzy Model is impossible to implement in this category due to large number of inputs, so we have nothing in that row. Here we also had the best results for Neuro-fuzzy Model, these results were shown in Table 8 and were demonstrated in Figure 9.

Estimation Models	Pred(20)	Average Error	MMRE	MMER
Neuro-Fuzzy Model	0.95	0.38	0.05	0.05
Fuzzy Model	It's impossible to implement, Due to very large rule set.			
Neural Network Model	0.89	0.73	0.11	0.11
Multiple Regression Model	0.95	0.50	0.07	0.07
Statistical Model	0.94	0.51	0.08	0.07

Table 8. Implementation Results for Effort Estimation with six inputs

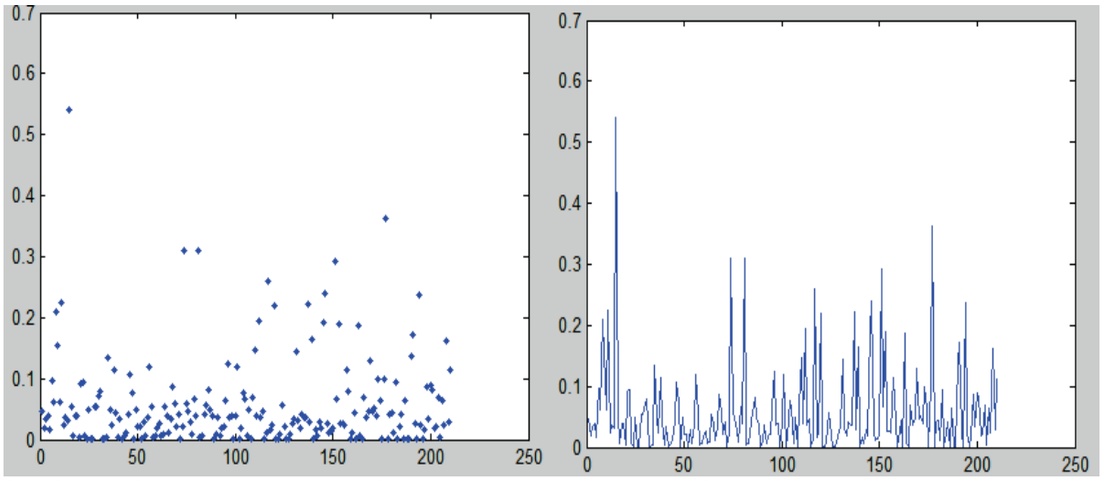

Fig. 9. MMRE Results for ANFIS with six inputs for Effort Estimation

As shown in Figure 7, most of estimations had less than 5% error and this emphasized that the performance of this model is better than the others.

- Third category: Time estimation with two inputs: log (Effort) and log(Average Team Size). The obtained results are organized in Table 9.

Estimation Models	**Pred(20)**	**Average Error**	**MMRE**	**MMER**
Neuro-Fuzzy Model	0.5103	0.4161	0.2594	0.2456
Fuzzy Logic Model	0.3913	0.5561	0.3435	0.3291
Neural Network Model	0.4119	0.5295	0.3266	0.3032
Multiple Regression Model	0.5149	0.4225	38.02	0.2640

Table 9. Implementation Results for Time Estimation

Figure 10 demonstrated that most of results had less than 3% error and it's pointed that this model is very accurate for prediction.

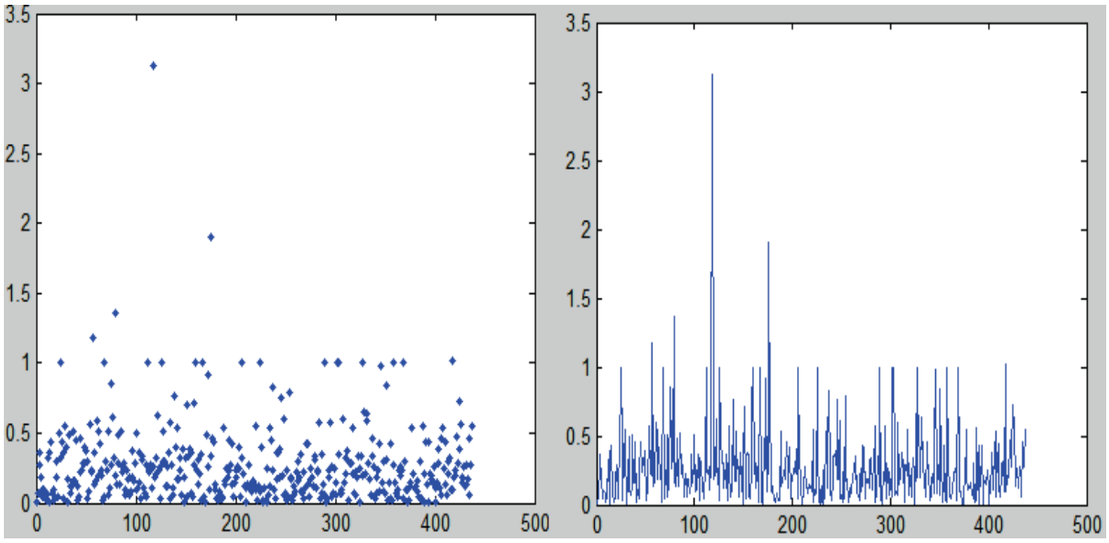

Fig. 10. MMRE Results of ANFIS for Time Estimation

The value of coefficient of determination (R^2) for ANFIS is equal to 0.9828 which indicated that more than 98 % of the variance in dependent variable can be explained by this model thus that's confidenceable.
The comparison plots of these models for Time and Effort estimation are shown in Figure 11 and 12 respectively.

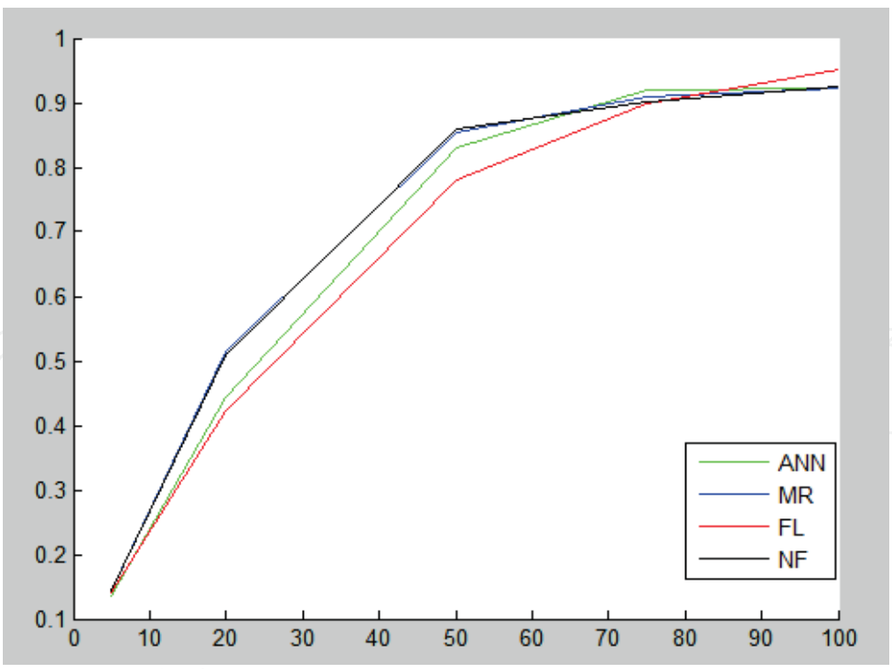

Fig. 11. Comparison plot for Time Estimation

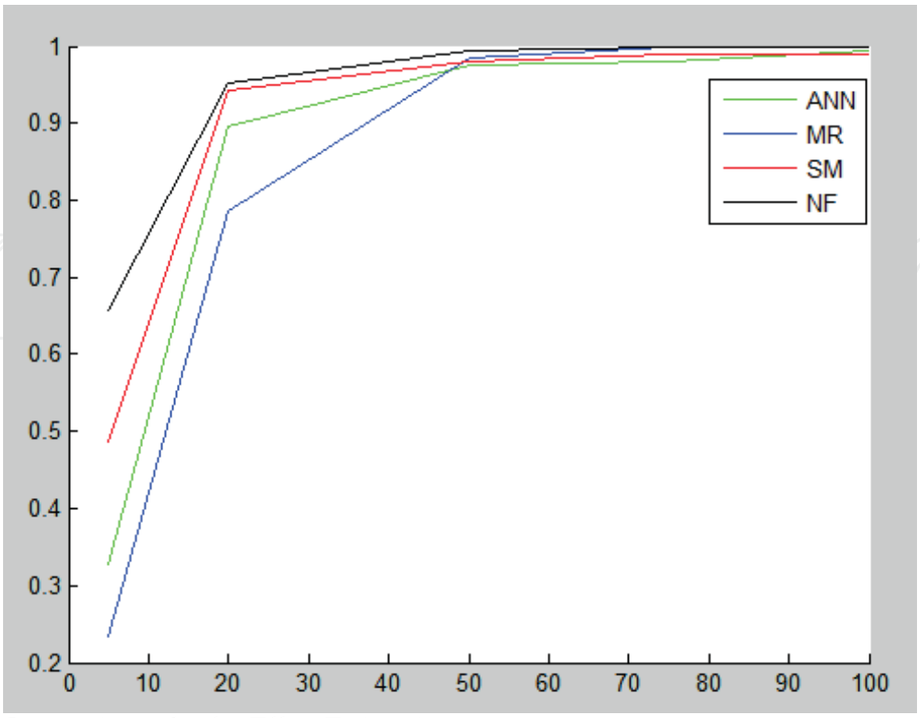

Fig. 12. Comparison plot for Effort Estimation

4. Conclusions and future works

As software development has become an essential investment for many organizations, software estimation is gaining an ever-increasing importance in effective software project management, quality management, planning, and budgeting.

The primary purpose of this study was to propose a precise method of estimation that takes account of and places emphasis on the various software development elements. We compared this neuro-fuzzy based software development estimation model with four other models such as neural network models, fuzzy logic models, multiple regression models, and statistical models.

The main benefit of this model is its good interpretability by using the fuzzy rules. Another great advantage of this research is that they could put together expert knowledge (fuzzy rules), project data and the traditional algorithmic model into one general framework that may have a wide range of applicability in software effort and time estimation. Also recent researches have tended to focus on the use of function points (FPs) in estimating the software development efforts and FPA (Function Point Analysis) assumes that the FP is the only factor which influences software development effort, however, a precise estimation should not only consider the FPs, which represent the size of the software, but should also include various elements of the development environment for its estimation. The factors significant to software development effort are project size, average number of developers that worked on the development, type of development, development language,

development platform, and the use of rapid application development which are used for estimation although FP as a software size metric is an important topic in the software prediction domain.

As a result of comparison, the effort and time estimation model, which is based on the neuro-fuzzy techniques, showed superior results in predictability than the other models mentioned in this study.

This study worked on the latest release of ISBSG data repository which is very large database recording 4106 software projects developed worldwide. Also for comparison of software development techniques we used three evaluation criteria: MMRE (Mean Magnitude Relative Error), MMER and Pred(20).

The proposed model has 98% coefficient of determination (R^2) which emphasize on the best performance of our proposed approach.

Some limitations in this domain are:

- ✓ Estimation of time and effort in earlier phase of software development is very difficult and it depends on lower level of estimation such as Size Estimation which is done by using External Inputs (EI), External Outputs (EO), External Queries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).
- ✓ Many existing research papers have proposed various effort estimation techniques and they still do not have an agreement which technique is the best across different cases.
- ✓ Also we don't have any dynamic learning algorithm for our model to adopt itself with any situation and completed our database in each estimation time. By adding the process maturity in effort estimation models as an input factor, we can improve the accuracy of estimation models.

This limitation gives us motivation to continue this research in our future work.

5. References

A. Abraham, (2005), "Adaptation of Fuzzy Inference System Using Neural Learning", Springer-Verlag Berlin Heidelberg.

K. K. Aggarwal, Y. Singh, P. Chandra, M. Puri, (2005), "Sensitivity Analysis of Fuzzy and Neural Network Models", ACM SIGSOFT Software Engineering Notes, Vol. 30, Issue 4, pp. 1-4.

R. Gray, S. G. MacDonell, (1999), "Fuzzy Logic for Software Metric Models throughout the Development Life-Cycle", in Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society – NAFIPS, New York, USA, 258 – 262.

A. R. Gray, S. G. MacDonell, (1997), "Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation", Fuzzy Information Processing Society 1997 NAFIPS' 97, Annual Meeting of the North American, 394 – 399.

X. Gu, G. Song, L. Xiao, (2006), "Design of a Fuzzy Decision-making Model and Its Application to Software Functional Size Measurement", IEEE, International Conference on Computational Intelligence for Modelling Control and Automation,

and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06) .

A. Heiat, (2002), "Comparison of artificial neural network and regression models for estimating software development effort", Information and Software Technology, 911-922.

X. Huang, D. Ho, J. Ren, L. F. Capretz, (2007), "Improving the COCOMO model using a neuro-fuzzy approach", Applied Soft Computing , Vol.7, Issue 1, pp. 29-40.

W. S. Humphrey, (2002), "A Discipline for Software Engineering", Addison Wesley.

A. Idri, A.Abran, (2001), "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), pp.85-96.

J. Jantzen, (1998), "Neuro-fuzzy modeling", Report no 98-H-874.

C. Lopez-Martin, C.Yanez-Marquez, A. Gutierrez-Tornes, (2008), "Predictive accuracy comparison of fuzzy models for software development effort of small programs", The journal of systems and software, Vol. 81, Issue 6, pp. 949-960.

S. G. MacDonell, A. R. Gray, J. M. Calvert. (1999), "FULSOME: Fuzzy Logic for Software Metric Practitioners and Researchers", in Proceedings of the 6th International Conference on Neural Information Processing ICONIP'99, ANZIIS'99, ANNES'99 and ACNN'99, Perth, Western Australia, IEEE Computer Society Press, 308-313.

M. C. Mackey, L. Glass, Oscillation and Chaos , (1977), in Physiological Control Systems, Science, Vol. 197, pp. 287–289.

A. A. Moataz, O.S.Moshood, A.Jarallah, (2005), "Adaptive fuzzy-logic-based framework for software development effort prediction", Information and Software Technology, Vol. 47, Issue 1, pp. 31-48.

NESMA, (1996), NESMA FPA Counting Practices Manual 2.0: NESMA Association.

M.O. Saliu, M. Ahmed and J. AlGhamdi. (2004), "Towards adaptive soft computing based software effort prediction", Fuzzy Information, Processing NAFIPS '04. IEEE Annual Meeting of the North American Fuzzy Information Processing Society, 1, 16-21.

M. T. Su, T. C. Ling, K. K. Phang, C. S. Liew, P. Y. Man, (2007), "Enhanced Software Development Effort and Cost Estimation Using Fuzzy Logic Model", Malaysian Journal of Computer Science, Vol. 20, No. 2, pp. 199-207.

W.N. Venables, B. D. Ripley, (2002), "Modern Applied Statistics with" New York: Springer.

W. Xia, L. F. Capretz, D. Ho, F. Ahmed, (2007), "A new calibration for Function Point complexity weights", Information and Software Technology, 50 (7-8), 670-683.

Zhizhong[a], Jiang, Craig Comstock, (2007), "The Factors Significant to Software Development Productivity", PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, Vol. 21, ISSN 1307-6884

Zhizhong[b], Jiang, Peter Naudé, (2007), "An Examination of the Factors Influencing Software Development Effort", International Journal of Computer, Information, and Systems Science, and Engineering Vol. 1, No. 3, 182-191.

Advanced Technologies

Edited by Kankesu Jayanthakumaran

ISBN 978-953-307-009-4

Hard cover, 698 pages

Publisher InTech

Published online 01, October, 2009

Published in print edition October, 2009

This book, edited by the Intech committee, combines several hotly debated topics in science, engineering, medicine, information technology, environment, economics and management, and provides a scholarly contribution to its further development. In view of the topical importance of, and the great emphasis placed by the emerging needs of the changing world, it was decided to have this special book publication comprise thirty six chapters which focus on multi-disciplinary and inter-disciplinary topics. The inter-disciplinary works were limited in their capacity so a more coherent and constructive alternative was needed. Our expectation is that this book will help fill this gap because it has crossed the disciplinary divide to incorporate contributions from scientists and other specialists. The Intech committee hopes that its book chapters, journal articles, and other activities will help increase knowledge across disciplines and around the world. To that end the committee invites readers to contribute ideas on how best this objective could be accomplished.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen