# Object Oriented Software Effort Estimate with Neuro Fuzzy Use Case Size Point (NFUSP) Based on Exact Weights

M. Kia
Department of Computer Science
Islamic Azad University
Sari Branch
Sari, Iran
iraji.ms@Gmail.com

***ABSTRACT:*** *Use case size point (USP) method has been proposed to estimate object oriented software development effort in early phase of software project and used in a lot of software organizations. Intuitively, USP is measured by counting the number of actors, preconditions, postconditions, scenarios included in use case models.This paper describes the idea to automatically classify the weight of this factors from use case model.*

*Even though several estimation procedures are available the Neural Network and fuzzy models presents advantages over normal estimation procedure. In this paper we develop a fuzzy Neural Network model to estimate the effort of object oriented software using Use Case size Point approach. In our proposed system fuzzy neural network use case size point has less error and system worked more accurate and appropriative than prior methods.*

**Keywords:** Use case point, Effort, Neuro fuzzy, Neuro Fuzzy Use Case Point, Object Oriented Software Effort

**Received:** 16 October 2011, Revised 21 December 2011, Accepted 26 December 2011



## 1. Introduction

Through industry collaboration we have experienced an increasing interest in software effort estimation based on use cases [1].

Estimation of object oriented software cost and effort is an important and hard management activity. This is due to the lack of information to making decisions in the early phases of the development, frequently characterized by uncertainty. To help the managers in this task, there are in the literature many estimation models that usually include two main metrics: Lines of Code (LOC) and Function Points (FP) [2], both of them with skills and limitations. LOC is dependent on the programming language and the FP Analysis (FPA) is subjective and based on human decisions [3].

As the most popular technique for object-oriented software cost estimation, Use Case Points (UCP) method, however, has two major drawbacks: the uncertainty of the cost factors and the abrupt classification [4]. Software cost estimation is vital for project bidding, budgeting, controlling and planning. Although in the literature many estimation models like Constructive Cost Model (COCOMO), Function Points(FP) have been proposed to help manager in estimation task, there is no obvious evidence shows that the accuracy is improved in last decades [5]. Achieving a highly accurate estimation is still a challenging issue in software engineering [4].

The subjects of estimation in the area of software development are size, effort invested, development time, technology used and quality. Particularly, development effort is the most important issue. So far, several effort models [6][7][8] have been proposed and most of them include software size as an important parameter. Function point is a measure of software size that uses logical functional terms business owners and users more readily understand [9]. Since it measures the functional requirements, the measured size stays constant despite the programming language, design technology, or development skills involved. To estimate the effort in earlier phase, use case point method has been proposed [10]. Use case point (UCP) is measured from a use case model that developes the functional scope of the software system to be developed. It is inuenced by the function point methods and is based on analogous use case point. There are several experience reports that show the usefulness of use case point for early estimation of software size.

Unified Modeling Language (UML) is a graphical modeling language that is used for visualizing, specifying, constructing and documenting software systems. To capture the functional requirements of a software project use case models are often employed. Use case modeling is a technique that has been widely used throughout the industry/research to describe and capture the functional requirements of a software system [11]. Since use cases and scenarios are developed as a normal part of requirements gathering and analysis they capture an accurate representation of the user's requirement.

The use case points method adjusts the size of the functionality of the system based on a number of technical and environmental factors. The technical factors are related to non-functional requirements on the system, while the environmental factors characterize the development team and its environment. The influence of these factors on the estimate was in this case much smaller (a 16% increase in the estimate) than the increase in actual effort spent by the companies that emphasized the development process and the quality of the code (an increase in actual effort of more than 100%) [1].

Software estimation models combining algorithmic models with machine learning approaches, such as neural networks and fuzzy logic, have been viewed with scepticism by the majority of software managers [12]. Briefly, 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 the neural network, the result is a neuro-fuzzy system that combines the advantages of both techniques [13]. Many metrics exist based on use case model in object oriented software .we Consider use case size point(USP) metric in this paper.

However, our proposed neuro-fuzzy model goes even further: it is a unique combination of neural networks and fuzzy logic. Specifically, we obtained equation 10**,** defined a suite of fuzzy sets to represent human judgement, and used a neural network to learn from a comprehensive historical database of software projects. A Neuro-Fuzzy use case size Points Calibration model that incorporates the learning ability from neural network and the ability to capture human knowledge from fuzzy logic is proposed and further validated in this paper.

The paper is organized in six sections. After the introduction in Section 1, Section 2 which also introduces the related works of effort estimation. Section 2 continues with explanations of use case point and use case size point approches effort estimation in section 3,4. in Section 5 is proposed Effort estimate with neuro fuzzy use case size point based on calibrate weights . It continues with discussions on the architecture of hybrid learning and fuzzy model validation, the error of observations for training data sets. Section 6 presents the conclusions of the research. The paper ends with a list of references.

**2. Literature review**

The use case points method was proposed by Karner in 1993, who also validated it on three projects [14].The method is an extension of MKII Function Points Analysis [15]. The use case points method adjusts the use case points based on a number of technical and environmental factors. This is similar to MKII Function Points, which also adjusts the size based on a number of calibration factors [1, 15]. These factors have been criticized for not improving the precision of the estimate [16]. The criticism relates both to the chosen set of factors and to their influence, but little investigation has been done on the effects on effort of the individual factors. Use cases are often used as input for estimating software development effort. Several studies show that a particular estimation method based on use cases, the use case points method, performs well early in aproject [17-18, 19, 42, 21].

In [22, 23, 24, 25] authors have used different neural network models for cost estimation. In [26, 27, 28, 29] the authors have used different case studies for estimating the effort of the software development using use case point approach. Neural Network is an

area which is leading the promise of producing consistently accurate estimate. The system effectively learns how to estimate from training set of completed projects [11]. In [11] authors have used neural network models for effort estimation using use case point approach.

Dealing with simple '*black*' and '*white*' answers is no longer satisfactory enough; degree of membership (suggested by Prof. Zadeh in 1965) became a new way of solving problems by treating data as imprecise or in a fuzzy form, there rule-based allowing the fuzzy system to handle certain degree of randomness without compromising on the efficiency of the system.

Fuzzy set is more powerful than classic set, because in real life most of the membership of the set is not a simply absolute "in or out" and fuzzy set mimic the way in which human interprets the terms, so it make is possible to deal with vagueness, imprecise and uncertainty when identifying the category. Fuzzy set theory has been applied software cost estimation for a long time.

Belchior et al [30] show that the metric FP does not have a continuous classification of the functionalities. They propose the Fuzzy Function Point Analysis (FFPA), that introduces a continuous and gradual classification of the system functionalities by using fuzzy numbers to perform the role of the traditional classification tables.

Rodrigo extended Use case Size Points (USP) to Fuzzy Use case Size Points (FUSP) by using fuzzy set theory [31]. Ryder researched on the application of fuzzy logicto COCOMO and Function Points models [32].Some techniques like fuzzy set and BBNs are introduced for software cost model. Fuzzy set use the degrees of membership in set to replace the absolute "*in or out*" membership in classic set, which results more precise assessments in software cost estimation [31,32,33].

Neuro-fuzzy systems are one of the most successful and visible directions of that effort. Neuro fuzzy hybridization is done in two ways [34]: a neural network equipped with the capability of handling fuzzy information (termed fuzzy neural network) and a fuzzy system augmented by neural networks to enhance some of its characteristics like flexibility, speed, and adapt-ability (termed neuro-fuzzy system(NFS) or ANFIS). An adapted neuro-fuzzy system (NFS) is designed to realize the process of fuzzy reasoning, where the connection weights of network correspond to parameters of fuzzy reasoning[34, 35]. These methodologies are thoroughly discussed in the literature [34].A second and distinct approach to hybridization is the genetic fuzzy systems (GFSs) [36]. A GFS is essentially a fuzzy system augmented by a learning process based on genetic algorithms (GAs). The parameter optimization has been the approach used to adapt a wide range of dissimilar fuzzy systems,as in genetic fuzzy clustering or genetic fuzzy systems [36]. However, genetic fuzzy systems are not subject of this work.

Marcio Rodrigo Braz, Silvia Regina Vergilio in [3] porposed FUSP (Fuzzy Use Case Size Points), considers concepts of the Fuzzy Set Theory to create gradual classifications that better deal with uncertainty. Results from an empirical evaluation show the applicability and some advantages of the proposed metrics.

Wei xia et al. [13] introduce a new calibration for Function Point complexity weights. A FP calibration model called Neuro-Fuzzy Function Point Calibration Model (NFFPCM) that integrates the learning ability from neural network and the ability to capture human knowledge from fuzzy logic is proposed. The empirical validation using International Software Benchmarking Standards Group (ISBSG) data.

**3. Use Case Point Approach**

Use case point (UCP) is calculated from use case model in [26]. Table 1 gives a brief overview of the steps of the method. In step 4, there are 13 technical factors, which are basically non-functional requirements on the system, see Table 2. There are also eight environmental factors that relate to the efficiency of the project in terms of the qualifications and motivation of the development team, see Table 3. The weights and the formula for technical factors are borrowed from the Function Points method proposed by Albrecht [20].

Karner himself proposed the weights and the formula for the environmental factors based on interviews with experienced developers and some estimation results. In step 6, the adjusted use case points (UCP) is multiplied by a productivity factor. The literature on the UCP proposes from 20 to 36 person hours per use case point (PHperUCP) depending on the values of the environmental factors [14, 37].

**4. Use Case Size Point (USP)**

USP measures the functionality by considering the structures and sections of a UC, counting the number and weight of

scenarios, actors, precondition and postconditions. To determine the USP, the sections of an expanded UC are analysed and its elements must be classified. The USP for the whole system is given by the sum of the USP for each UC. The following steps show how to calculate USP [3].

| Step | Rule | Output |
|---|---|---|
| 1 | Classify actors:<br>a)Simple , WF (Weight Factors)=1<br>b)Average , WF=2<br>c)Complex , WF=3 | Unadjusted Actor Weight (UAW)=Σ(#Actors * WF) |
| 2 | Classify Use Cases:<br>a)Simple- 3 OR Fewer transactions, WF=5<br>b)Average- 4 to 7 transactions, WF=10<br>c)Complex-more then7 transactions, WF=15[2] | Unadjusted Use Case Weight (UUCW)=<br>Σ(#Use Cases * WF) |
| 3 | Calculate the Unadjusted Use Case Point (UUCP) | UUCP = UAW + UUCW |
| 4 | Assign values to the technical and environmental factors [0..5], multiply by their weights[ -1..2], and calculate the weighted sums (TFactor and Efactor). Calculate TCF and EF as shown | Technical Complexity Factor (TCF) = 0.6 + (0.01*TFactor)<br>Environmental Factor (EF) = 1.4 + (-0.03 + EFactor) |
| 5 | Calculate the Unadjusted Use Case Point (UCP) | UCP = UUCP * TCF * EF |
| 6 | Estimate Effort (E) in Person-hours | E = USP * PHperUCP |

Table 1. The UCP estimation method

1. Actors classification: Each actor has its complexity (CA) determined according to the data provided to or received from the UC being classified (Table 4). The total complexity of actors in the UC (TPA) is calculated by Equation 1.

$$TPA = \sum_{i=1}^{n} CA_i \quad (1)$$

n - Number of actors in the UC

2. Precondition classification: Each precondition of the UC has the complexity (CPrC) determined according to the number of logical expressions tested by the condition (Table 5). The total complexity of the preconditions (TPPrC) is given by Equation 2.

$$TPPrC = \sum_{i=1}^{n} CP_r C_i \quad (2)$$

3. Main scenario classification: The main scenario must be classified according to its number of entities and to the number of elementary steps needed to the scenario conclusion. The complexity of the scenario (PCP) is given by the sum of both values (number of entities + number of steps), according to Table 6.

| Factor | Description | Weight |
|---|---|---|
| T1 | Distributed System | 2 |
| T2 | Response or throughputPerformance objectives | 2 |
| T3 | End-user efficiency | 1 |
| T4 | Complex internal processing | 1 |
| T5 | Reusable Code | 1 |
| T6 | Easy to install | 0.5 |
| T7 | Easy to use | 0.5 |
| T8 | Portable | 2 |
| T9 | Easy to change | 1 |
| T10 | Concurrent | 1 |
| T11 | Includes security features | 1 |
| T12 | Provides access for third parties | 1 |
| T13 | Special user trainingfacilities are required | 1 |

Table 2. Technical factor

| Factor | Description | Weight |
|---|---|---|
| F1 | Familiar with RationalUnified Process | 1.5 |
| F2 | Application experience | 0.5 |
| F3 | Object-oriented experience | 1 |
| F4 | Lead analyst capability | 0.5 |
| F5 | Motivation | 1 |
| F6 | Stable requirements | 2 |
| F7 | Part-time workers | -1 |
| F8 | Difficult programminglanguage | -1 |

Table 3. Enviromental factor

4 . Alternative scenario classification: All the alternative scenarios are classified similarly to the main scenario. Each alternative scenario receives a number of points (PCA) according to Table 6. The total complexity of the alternative scenarios (TPCA) is given by Equation 3.

$$TPCA = \sum_{i=1}^{n} PCA_i \quad (3)$$

n - Number of alternative scenario in the UC

5. Exception classification: Each exception present in the UC must also be analysed according to its complexity (CE), determined by the number of logical expressions tested to detect the exception occurrence. The total points added by exceptions (TPE) are determined by Equation 4. Table 7 helps in this classification.

$$TPE = \sum_{i=1}^{n} CE_i \quad (4)$$

n - Number exceptions in the UC

6. Postcondition classification: The complexity of the postconditions (CPoC) is determined according to the number of related

| Complexity | Entities | UUSP |
|---|---|---|
| Simple | ≤ 5 | 2 |
| Average | 6 to 10 | 4 |
| Complex | > 10 | 6 |

Table 4 . UCP actor classification

| Complexity | Texted expression | UUSP |
|---|---|---|
| Simple | 1 logical expression | 1 |
| Average | 2 or 3 logical expressions | 2 |
| Complex | 3 logical expressions | 3 |

Table 5. USP Precondition Classification

| Complexity | Entities + steps | UUSP |
|---|---|---|
| Very Simple | ≤ 5 | 4 |
| Simple | 6 to 10 | 6 |
| Average | 10 to 15 | 8 |
| Complex | 16 to 20 | 12 |
| Very complex | > 20 | 16 |

Table 6. USP Scenario Classification

| Complexity | Texted expression | UUSP |
|---|---|---|
| Simple | 1 logical expression | 1 |
| Average | 2 or 3 logical expressions | 2 |
| Complex | > 3 logical expressions | 3 |

Table 7. USP Exception Classification

| Complexity | Entities | UUSP |
|---|---|---|
| Simple | ≤ 3 | 1 |
| Average | 4 to 6 | 2 |
| Complex | > 6 | 3 |

Table 8. USP Postcondition Classification

| Factor | Requirement | Influence |
|---|---|---|
| F1 | Data Communication | I1 |
| F2 | Distributed Processing | I2 |
| F3 | Performence | I3 |
| F4 | Equipment utilization | I4 |
| F5 | Transaction Capacity | I5 |
| F6 | On-line Input of data | I6 |
| F7 | User efficiency | I7 |
| F8 | On-line update | I8 |
| F9 | Code reuse | I9 |
| F10 | Complex processing | I10 |
| F11 | Easiness of deploy | I11 |
| F12 | Easiness operation | I12 |
| F13 | Many places | I13 |
| F14 | Facility of change | I14 |

Table 9. Technical Factors

| Factor | Description | Influence |
|---|---|---|
| E1 | Formal development process existence | I1 |
| F2 | Experience with the application being developed | I2 |
| F3 | Experience of the term with the used technologies | I3 |
| F4 | Presence on an experienced analyst | I4 |
| F5 | Stable requirements | I5 |

Table 10. Environmental Factors

entities (Table 8). The total complexity of postconditions (TPPoC) is given by Equation 5.

$$TPPoC = \sum_{i=1}^{n} CPoC_i \quad (5)$$

n - Number of post conditions in the UC

7. Calculation of the unadjusted value: The Unadjusted Use-case Size Point (UUSP) is given by the sum of the complexity values of all sections of the UC (Equation 6).

$$UUSP = TPA + TPPrC + PCP + TPCA + TPE + TPPoC \quad (6)$$

8. Application of the adjustment factor: The way USP calculates the adjustment factor is derived from FP and UCP

• Technical Adjustment Factors: The technical factors represent the influence that some technical characteristics (existent in the software being developed and inherent to all UCs of the system) could have upon the software. Each adjustment factor in Table 9 receives a value between 0 and 5 according its influence in the UC, where 5 represents the greatest influence. The technical adjustment factor is calculated by Equation 7.

$$FTA = 0.65 + ( 0.01 * \sum_{i=1}^{14} I_i ) \quad (7)$$

• Environmental Adjustment Factors: The environmental factors represent some characteristics existent at the development environment that could influence the software cost. Each factor from Table 10 receives a value and the Environmental Adjustment Factor (FAA) is given by Equation 8.

$$FFA = 0.01 * \sum_{i=1}^{5} I_i \quad (8)$$

9. Concluding the calculation: the final value for a UC is given by Equation 9.

$$USP = UUSP * (FTA - FAA) \quad (9)$$

**5. Study Objectives and Method**

The objectives of this study are:

1. calibration of the use case point factor weight values Exactly to fuzzy further enhanced improvements in the software effort estimation process

2. Artificial neural network approach to calibrate the function point weight values provides improvement in the software size estimation process.

The weight values of Unadjusted use case size Point (UUSP) in Tables 4-8 are said to reflect the functional size of software [24], Karner determined them in 1993. Since 1993, software development has been growing steadily and is not limited to one organization or one type of software. Thus, there is need to calibrate these weight values to reflect the current software industry trend. The ISBSG Development and Enhancement repository has over 5,600 projects from 29 countries and 11 major industry types. This industry data can be used to estimate, benchmark and improve the planning and management of projects. Learning UUSP weight values from ISBSG data repository using neural network for calibration to reflect the current software industry trend.

Our Neuro-Fuzzy approach presented in this paper is a novel combination of the above three approaches. It obtains a simple equation 10, defines a suite of fuzzy sets to represent human judgment Exactly, and uses neural network to learn the calibrated parameters from the historical project database.

The equation from statistical analysis is fed into neural network learning. The calibrated parameters from neural network are then utilized in fuzzy sets and the users can specify the upper and lower bounds from their human judgment.

The first, the neural network technique is based on the principle of learning from previous data. This neural network is trained with a series of inputs and desired outputs from the training data so as to minimize the prediction error. Once the training is complete and the appropriate weights for the network links are determined, new inputs are presented to the neural network to predict the corresponding estimation of the response variable. The final component of our model, fuzzy logic, is a technique used to make rational decisions in an environment of uncertainty and imprecision. It is rich in its capability to represent the human linguistic ability with the terms of fuzzy set, fuzzy membership function, fuzzy rules, and the fuzzy inference process (figure 1). PHperUSP proposes 2.5 person hours per use case Size point.

**5.1 Neural network**

Developing a neural net solution means teaching the net a desired behavior. This is called the learning phase. Either sample data sets or a "*teacher*" can be used in this step. A teacher is either a mathematical function or a person that rates the quality of the

neural net performance. Since neural nets are mostly used for complex applications where no adequate mathematical models exist and rating the performance of a neural net is difficult in most applications, most are trained with sample data(figure 2).

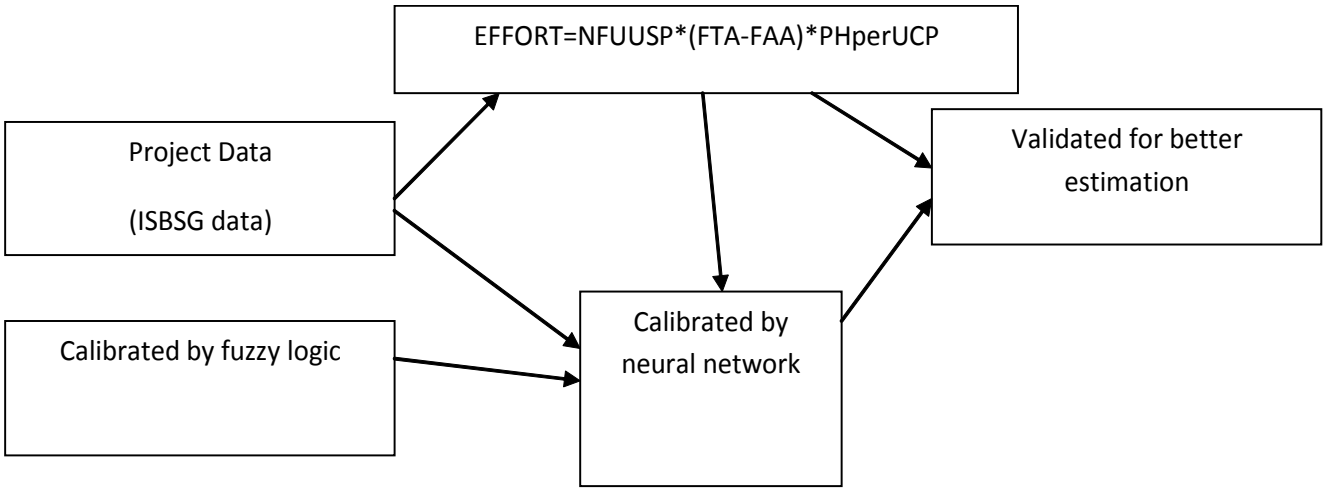


Figure 1:neuro fuzzy use case size point model

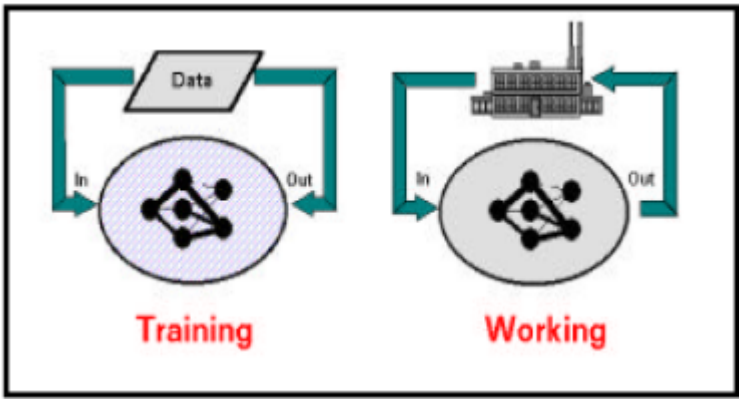


Figure 2. Training and working phase for supervised learning

**5.1.1 Neuron Model**

An elementary neuron with R inputs is shown figure 3. Each input is weighted with an appropriate w. The sum of the weighted inputs and the bias forms the input to the transfer function f. Neurons may use any differentiable transfer function f to generate their output.

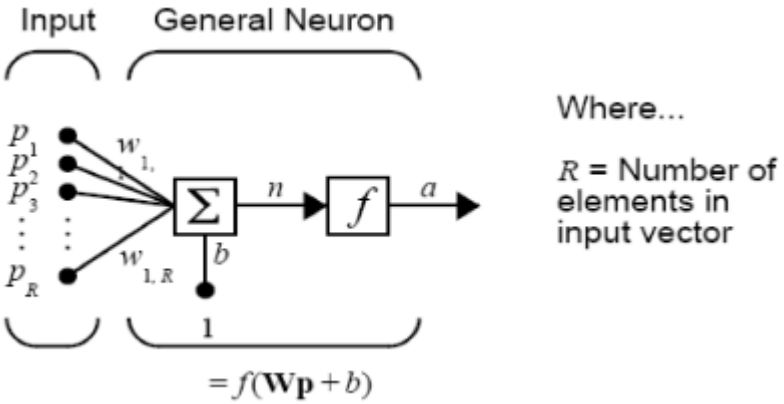


Figure 3. Structure a neuron

Feedforward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neuronswith nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors (figure 4). The linear output layer lets the network produce values outside the range –1 to +1. On the other hand, if you want to constrain the outputs of a network (such as between 0 and 1), then the output layer should use a sigmoid transfer function (such as logsig). This network can be used as a general function approximator. It can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer [38].

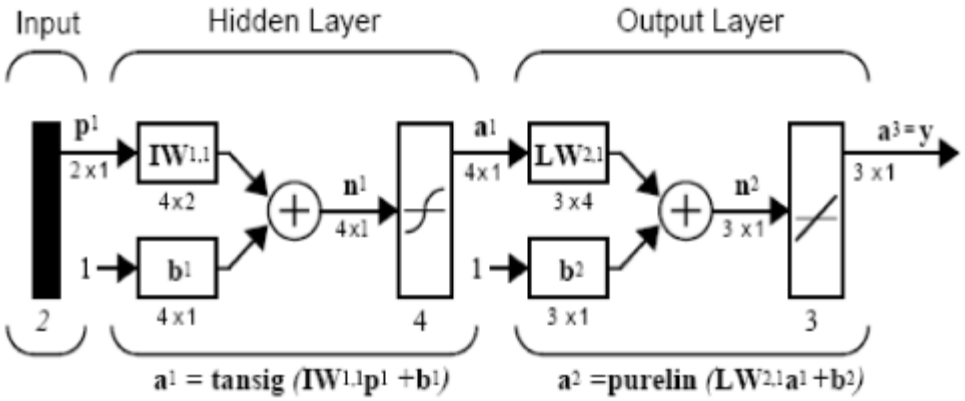


Figure 4. Feed forward neural network with two layers

**5.1.2 Neural Network Step For Estimating Use Case Weight**

Back-propagation feed forward neural network approach is used to predict the size of the software using UCP approach. A neural network is constructed with 27 inputs and 1 output effort. The input nodes represent the distinguishing parameters of use case point approach and the output nodes represent the effort. The network is constructed by using MATLAB. In our experiment the training function we have considered is traingda, the adaptation learning function considered was learngdm, and the performance function used was Mean Square Error (MSE). The projects considered for this research are taken from the use case point ISBSG data. The input nodes represent the following features of software projects:

1. Number of simple actor
2. Number of average actor
3. Number of complex actor
4. Number of simple Precondition
5. Number of average Precondition
6. Number of complex Precondition
7. Number of very simple Scenario
8. Number of simple Scenario
9. Number of average Scenario
10. Number of complex Scenario
11. Number of very complex Scenario
12. Number of simple Exception
13. Number of average Exception
14. Number of complex Exception
15. Number of simple Post condition
16. Number of average Post condition
17. Number of complex Post condition
18. Technical cost Factor(FTA)
19. Environmental Factor(FAA)
20. Data communication - F1
21. Distributed processing- F2

22. Performance –F3
23. Equipment utilization - F4
24. Transaction Capacity - F5
25. On-line input of data - F6
26. User efficiency -F7
27. On-line update - F8
28. Code reuse - F9
29. Complex processing - F10
30. Easiness of deploy - F11
31. Easiness operation -F12
32. Many places -F13
33. Facility of change- F14
34. Formal development process existence - E1
35. Experience with the application being developed- E2
36. Experience of the team with the used technologies- E3
37. Presence on an experienced analyst -E4
38. Stable requirements -E5

**5.2 Designing by neuro fuzzy**

**5.2.1 Neuro fuzzy**

It immediately comes to mind, when looking at a neural network, that the activation functions look like fuzzy membership functions. Indeed, an early paper from 1975 treats the extension of the McCulloch-Pitts neuron to a fuzzy neuron (Lee & Lee, 1975; see also Keller & Hunt, 1985).

The one neuron in the output layer, with a rather odd appearance, calculates the weighted average corresponding to the *center of defuzzification* in the rule base. Backpropagation applies to this network since all layers are differentiable. Two possibilities for learning are apparent. One is to adjust the weights in the output layer, i.e, all the singletons weight until the error is minimized. The other is to adjust the shape of the membership functions, provided they are parametric. Also rule weight can be changed with training data. In Neuro fuzzy model is not necessary output be linear [43].

**5.2.2 Sugeno model**

A typical rule in a Sugeno fuzzy model has the form
If Input 1 = $x$ and Input 2 = $y$, then Output is $z = ax + by + c$
For a zero-order Sugeno model, the output level $z$ is a constant ($a = b = 0$).

The output level $z_i$ of each rule is weighted by the firing strength wi of the rule.For example, for an AND rule with Input 1 = $x$ and Input 2 = $y$, the firing strength is $w_i$ =AndMethod (F1(x), F2(y));where $F1, 2$ (.) are the membership functions for Inputs 1 and 2. The final output of the system is the weighted average of all rule outputs, computed as:

$$Final\ Output = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \qquad (1)$$

A Sugeno rule operates as shown in the following diagram(Figure 6)

ANFIS (Adaptive Neuro Fuzzy Inference System) is an architecture which is functionally equivalent to a Sugeno type fuzzy rule base (Jang, Sun & Mizutani, 1997; Jang & Sun,1995). Under certain minor constraints the ANFIS architecture is also equivalent to a radial basis function network. Loosely speaking ANFIS is a method for tuning an existing rule base with a learning algorithm based on a collection of training data. This allows the rule base to adapt.

**5.2.3 Anfis**

Consider the fuzzy neural network in figure 7. The output of the first layer nodes are the degree of membership of linguistic

Figure 5. Neuro fuzzy model control system

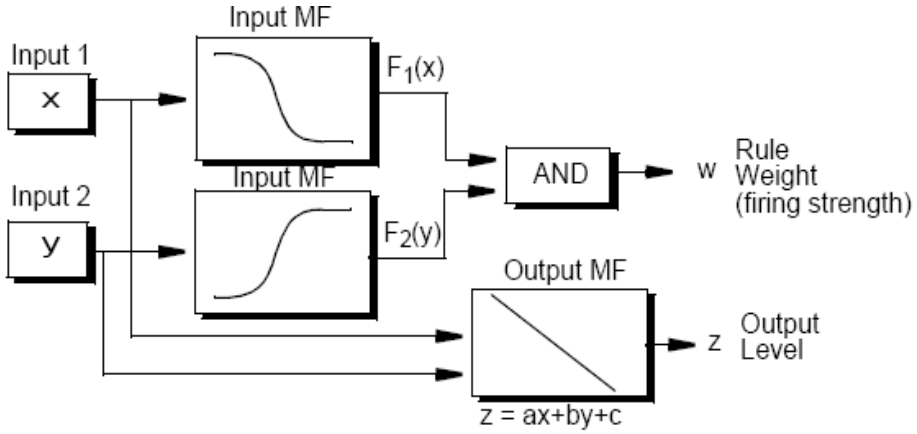


Figure 6. Sugeno model

variables. Typically, in this layer bell –shaped functions are used. Bell-shaped function is shown in Relationship 6. The purpose of learning in this layer is adjusting the parameters of membership function of inputs.

$$f(x) = \exp\left[\frac{-1}{2}\left(\frac{x - a_{i_1}}{b_{i_1}}\right)^2\right] \qquad (2)$$

The second layer is '*rules layer*'. In this layer, the condition part of rules is measured by usually Min fuzzy logic operator, and the result will be the degree of activity of rule resultant. Learning, in this layer, is the change of the amount of activity of rules resultant, regarding to the '*training data*', given to the network. In the third layer, we'll get the linear combination of rules

resultant rate, and in order to determine the degree of belonging to a particular category, Sigmund function is used in layer 4 [44].

If a series of training vectors is given to the network in the form of the formula 7:

$$\{(x^k, y^k), k = 1,..., K\} \tag{3}$$

where $x^k$ refers to the K- th input pattern, then we have:

(4)

The error function for K pattern can be defined by relationship 9:

$$E_k = \frac{1}{2}[(O_1^k - y_1^k)^2 + (O_2^k - y_2^k)^2)] \tag{5}$$

where is $y^k$ desired output, and $O^k$ is computed output.

$$y^k = \begin{cases} (1,0) \; if \; x^k \; belongs \; to \; class \; 1 \\ (0,1) \; if \; x^k \; belongs \; to \; class \; 2 \end{cases}$$

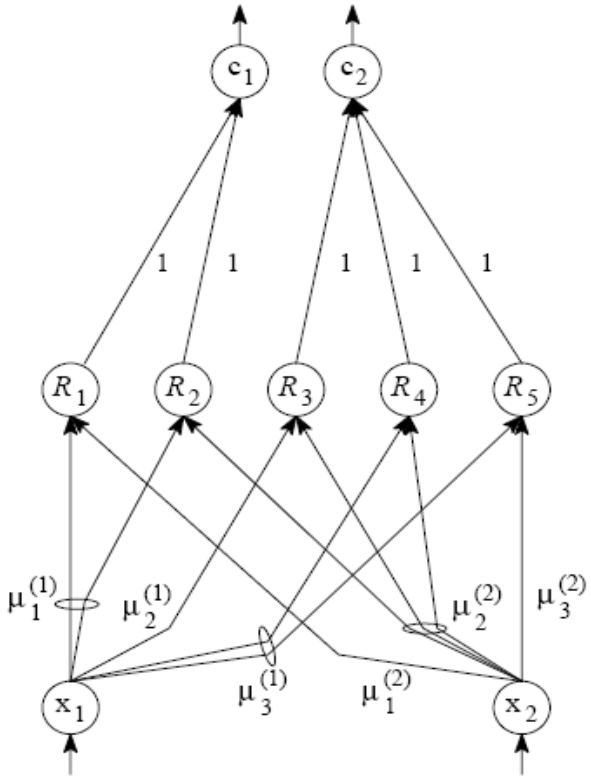


Figure 7. adaptive neuro fuzzy Network(anfis)

**5.2.4 Fuzzy Use Case SIZE Point (FUSP)**

In use case size counting method, each component, such as actors is classified to a weight level determined by the numbers of its entities (step 1 of section 4), Such weight classification is easy to operate, but it may not fully reflect the true color of the software complexity under the specific software application. For example, a software project with two use cases, A with 10 entities and B with 6 entities. According to the weight matrix, A and B are classified as having the same complexity and are assigned the same weight value of 4. However, A has 4 more transaction than B and is certainly more complex. They are now assigned the same complexity, which is recorded as Observation 1: ambiguous classification? [13]

The metric USP presents new elements for measuring the functionality of the UCs. However, it also uses a discrete classification

of the functionalities complexity, like UCP and FPA. The use of the classification tables does not allow a gradual change from one complexity category to another. To allow such gradual change, we extended the metric USP by adopting the same FFPA steps [39] and introduced a metric named NFUSP (Neuro Fuzzy Use Case Size Points).

**5.2.5 Fuzzy logic calibration step**

The classification tables are transformed into a continuous classification, this process is called fuzzyfication (a more formal definition to fuzzyfication could be found in [40]). This can be made through the generation of a trapezoidal fuzzy number to each complexity category found on the classification tables. Then, each classification table for a UC (actors, preconditions, exceptions, etc) is represented by a graph, To generate the graph, that is the trapezoidal number, the following variables are calculated, for each category in the classification tables ($1 \leq i \leq n$), and n is the number of linguistic terms in the classification table being analysed).

Shortly [3]:

- mi = lower value of the linguistic term Ti in the classification table
- ni = (mi + mi + 1) / 2
- ai = ni - 1
- bi = mi + 1

Table 11 shows the values of the above variables for the USP classification tables. For example, the table for use case Classification has three linguistic terms: Simple (T1), Average (T2) and Complex (T3). The graphs obtained for each USP classification table are present in Figure 8.

| Table | m1 | n1 | a1 | b1 | m2 | n2 | a2 | b2 | m3 | n3 | a3 | b3 | m4 | n4 | a4 | b4 | m5 | n5 | a5 | b5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 3.5 | | 6 | 6 | 8.5 | 3.5 | 11 | 11 | | 8.5 | | | | | | | | | |
| 5 | 1 | 1.5 | | 2 | 2 | 3 | 1.5 | 4 | 4 | | 3 | | | | | | | | | |
| 6 | 1 | 3.5 | | 6 | 6 | 8.5 | 3.5 | 11 | 11 | 13.5 | 8.5 | 16 | 16 | 18.5 | 13.5 | 21 | 21 | | 18.5 | |
| 7 | 1 | 1.5 | | 2 | 2 | 3 | 1.5 | 4 | 4 | | 3 | | | | | | | | | |
| 8 | 1 | 2.5 | | 4 | 4 | 5.5 | 2.5 | 7 | 7 | | 5.5 | | | | | | | | | |

Table 11. Values for the Fuzzyfication of the Terms

Membership functions for output actor weight are the triangular type, because these types of membership functions are appropriate to use in preserving the values in the complexity weight matrices (Figure 9).

The fuzzy inference process using the Mamdani approach [41] is applied to evaluate use case component weight degree when the linguistic terms, the fuzzy sets, and the fuzzy rules are defined.

Fuzzy Logic Rules:
If Input =simple entities then weight output = simple
If Input = average entities then weight output = average
If input = complex entities then weight output = complex

The antecedent result as a single number implies the consequence using the min (minimum) implication method. Each rule is applied in the implication process and produces one consequence. The aggregation using the max (maximum) method is processed to combine all the consequences from all the rules and gives one fuzzy set as the output. Finally, the output fuzzy set is defuzzified to a crisp single number using the centroid calculation method.

An example of the complete fuzzy inference process is shown in Figure 10. Input value are set to entities: 10. The antecedent parts of the fuzzy rules whose degrees are not equal to zero are activated and represented by the light gray shades. Here, rules 2, 3 are activated for the antecedent part inpute. Finally, the consequent fuzzy set is defuzzified, using the centroid calculation method, and the output is achieved as a single value of 4.36 the bold black line in the output fuzzy set located in the bottom right of the Figure 10.

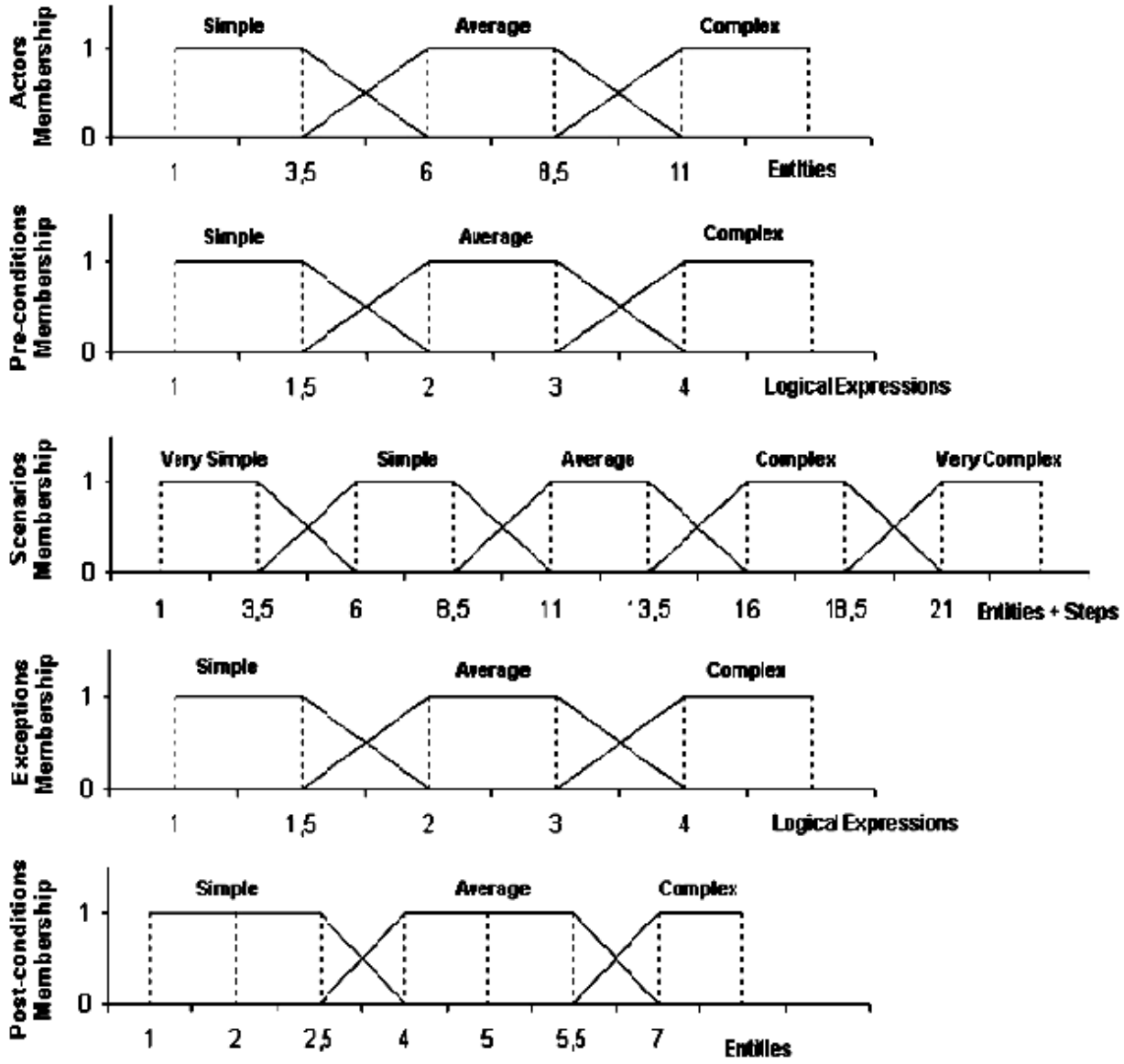


Figure 8. Fuzzy graph for USP tables

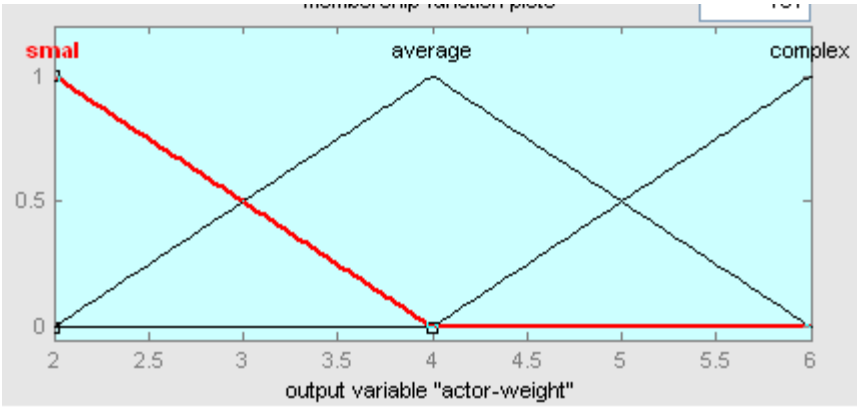


Figure 9. Membership function for actor weights

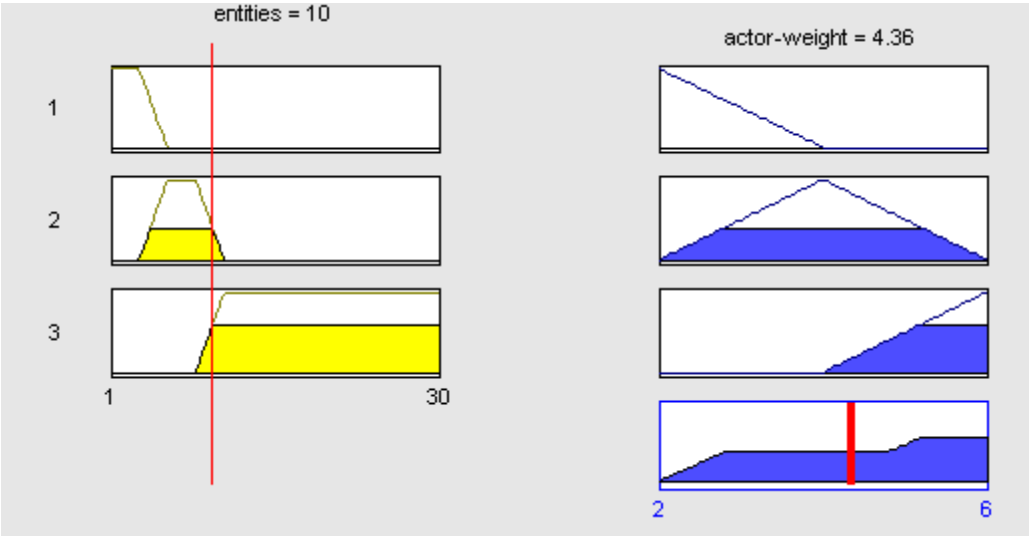


Figure 10. Fuzzy inference process of neuro-fuzzy use case size point model

Afterwards, a fuzzy complexity measurement system that takes into account all elements of use case size point in tables 4-8 is built after the fuzzy logic system for each use case size point component is established, as shown in Figure 8. Each USP element is into a Fuzzy Logic System (FLS). The outputs of all five FLS are summed up and become the neuro fuzzy Unadjusted Use Case Size Points(NFUUSP) Which is used to calculate effort(figure 11).

$$EFFORT = NFUUSP * (FTA - FAA) * PHperUSP \quad (10)$$

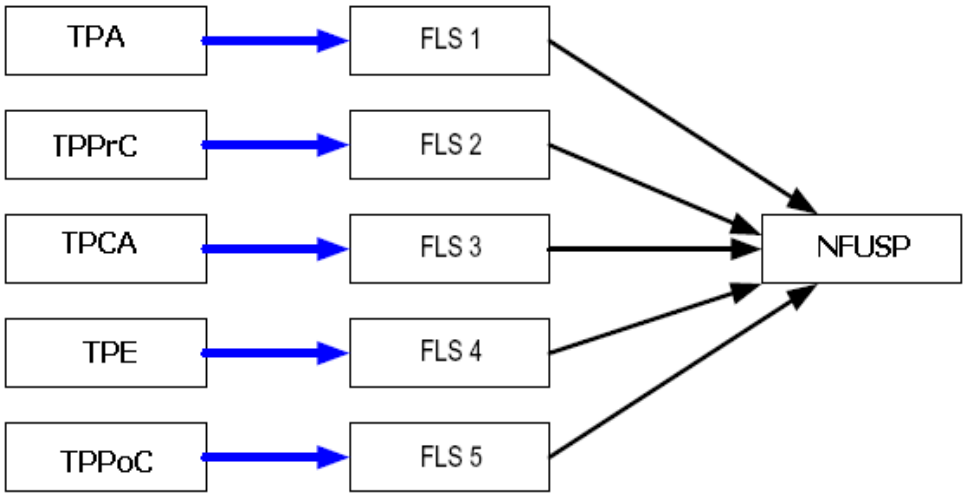


Figure 11. Fuzzy inferece process of neuro-fuzzy use case size point model

**6. Discussion and Conclusion**

Several methods exist to compare cost estimation models. Each method has its advantages and disadvantages. In this work, The Magnitude of Relative Error (MRE) will be used. MRE for each observation i can be obtained as:

$$MER_i = \frac{|\, Actual\ Effort_i - \Pr edictedEffort_i \,|}{\Pr edictedEffort_i} \quad (11)$$

MMRE can be achieved through the summation of MRE over N observations:

$$MMER = \frac{1}{2}\sum_{1}^{N} MER_i \qquad (12)$$

After training neural network by ISBSG data, we have applied Our proposed neuro fuzzy system on seven samples of projects. Results as the table below, respectively.

| PROJECT # | Actual effort | effort in Use case size point | Effort in Neuro fuzzy use case point | MER (USP) | MER(NFUSP) |
|---|---|---|---|---|---|
| 1 | 2905 | 2610 | 2873 | 0.113027 | 0.011138 |
| 2 | 2200 | 2176 | 2196 | 0.011029 | 0.001821 |
| 3 | 1759 | 1591 | 1638 | 0.105594 | 0.073871 |
| 4 | 1542 | 1364 | 1538 | 0.130499 | 0.002601 |
| 5 | 1025 | 992 | 1015 | 0.033266 | 0.009852 |
| 6 | 2784 | 2600 | 2725 | 0.070769 | 0.021651 |
| 7 | 2961 | 2891 | 2927 | 0.024213 | 0.011616 |
| MMER | | | | 0.069771 | 0.018936 |

Table 12. Evaluation results from sample projects

As you can see in the table 12 Thus effort estimation with neuro fuzzy Unadjusted Use Case Size Points(NFUUSP) MMER is less and NFUUSP accuracy further. For future work can be other different types of membership functions, different types of neural network and optimization algorithms like genetic algorithm considered.

However, software development is a rapidly growing industry and these calibrated weight values will not reflect tomorrow's software. The advantage of Neural Networks is it has the learning capability to adapt new data. On the other hand, Fuzzy Systems has the capability to handle numerical data and linguistic knowledge simultaneously.In the future, when modern project data is available, the USP weight values will again need to be re-calibrated to reflect the latest software industry trend. The neuro-fuzzy USP model is a framework for calibration and the neuro-fuzzy USP calibration tool can automate the calibration process when data becomes available.

**7. Acknowledgments**

This work received support from the Department of Computer Engineering, Islamic Azad University, Sari Branch.

**References**

[1] Bente Anda, Hans Christian Benestad Siw Elisabeth Hove. (2005). A Multiple-Case Study of Software Effort Estimation based on Use Case Points.

[2] Albrecht, A. (1979). Measuring Application Development Productivity. *In*: Proc. of IBM Applications Development Symposium, p. 83–92, October.

[3] Marcio Rodrigo Braz, Silvia Regina Vergilio. (2006). Software Effort Estimation Based on Use Cases, *In*: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06). IEEE.

[4] WANG Fan,YANG Xiaohu, ZHU Xiaochun, CHEN Lu. (2009). Extended Use Case Points Method For Software Cost Estimation, IEEE.

[5] Molokken, K., Jorgensen, M. (2003). A review of software surveys on software effort estimation. *In:* Empirical Software Engineering, *In:* ISESE. Proceedings. International Symposium on, p. 223–230.

[6] Basili, V. R., Freburger, K. (1981). Programming measurement and estimation in the Software Engineering Laboratory, *Journal of Systems & Software*, 2, p. 47-57.

[7] Boehm, B. W. (1981). Software Engineering Economics, Prentice-Hall.

[8] Walston,C. E., Felix, C. P. (1977). A method of program measurement and estimation, *IBM Systems Journal*, 16 (1) 54-73.

[9] Albrecht,A. J. (1994). Function Point Analysis, Encyclopedia of Software Engineering, 1, 518-524 .

[10] Schneider, S., Winters, J. P. (2001). Applying Use Cases, Second Edition, Addison Wesley.

[11] Ajitha, S., Suresh Kumar, T. V., Evangelin Geeth, Rajani Kanth, K. (2010). Neural Network Model For Software Size Estimation Using Use Case Point Approach, 5th International Conference on Industrial and Information Systems, ICIIS , Jul 29 - Aug 01, India.

[12] Idri, A., Khosgoftaar, K. H., Abran, A. (2002). Can neural networks be easily interpreted in software cost estimation? *In*: Proceedings of the IEEE International Conference on Fuzzy Systems,1162–1167.

[13] Wei Xia , Luiz Fernando Capretz , Danny Ho , Faheem Ahmed . (2008). A new calibration for Function Point complexity weights, Information and Software Technology 50, 670–683.

[14] Karner. (1993). Resource Estimation for Objectory Projects. Objective Systems SF AB. 17. September.

[15] Symons, P. R. (1991) Software Sizing and Estimating MK II FPA (Function Point Analysis). John Wiley & Sons.

[16] Lokan., C. and Abran, A. (1999). Multiple Viewpoints in Functional size Measurement. *In*: Proc. of the International Workshop on Software measurement (IWSM'99), Lac Supérieur, Canada, September 8- 10.

[17] Anda, B. (2002). Comparing Use Case based Estimates with Expert Estimates. *In*: Proc. of Empirical Assessment in Software Engineering (EASE), Keele, United Kingdom, April 8-10.

[18] Anda, B., Dreiem, H., Sjøberg, D. I. K., Jørgensen, M. Estimating Software Development Effort Based on Use Cases – Experiences from Industry. The 4th International Conference on the Unified Modeling Language, Concepts, and Tools (UML), Canada, October 1-5, LNCS 2185,Springer-Verlag.

[19] Arnold, P. and Pedross, P. (1998). Software Size Measurement and Productivity Rating in a Large- Scale Software Development Department. The 20th International Conference on Software Engineering (ICSE), Kyoto, Japan, April 19-25, p. 490-493.

[20] Albrecht, A. J. (1979). Measuring Application Development Productivity. *In*: Proceedings of the IBM Applic. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA, USA, p. 83-92.

[21] Ribu, K. (2001). Estimating Object-Oriented Software Projects with Use Cases. MSc thesis, November.

[22] Smith Alice, E., Mason Anthony, K. Cost Estimation Predictive Modeling: Regression versus Neural Network

[23] Tirimula Rao, B., Sameet, B., Kiran Swathi, K., Vikram Gupta, K., RaviTeja, Ch., Sumana, S. (2009). A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN) *IJCSNS International Journal of Computer Science and Network Security,* 9 (6) June.

[24] Aggarwal, K. K. Yogesh Singh, Pravin Chandra, Manimala Puri. (2005). Evaluation of various training algorithms in a neural network model for software engineering applications. ACM SIGSOFT Software Engineering Notes Page 1 July Volume 30 Number 4.

[25] Man-Yi Chen, Ding-Fang Chen. (2002). Early cost estimation of strip-steel coiler using BP neural network, Proceedings of the first international conference on machine learning and cybernetic, Beijing 4-5 Nov.

[26] Schneider, G., Winters, J. P. (2001). Applying Use Cases, SecondEdition, Addison Wesley.

[27] Kirsten Ribu, Estimating object-oriented software projects with use cases.

[28] Laird Linda, M., Carol Brennan, M. Software Measurement and Estimation. A prctical approach.

[29] Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S., Maegawa, Y. (2004). Estimating effort by Use Case Points: method, tooland case study, Proceedings of the 10th International Symposium on Software Metrics.

[30] Belchior, A. D., Junior, O. S. L., Farias, P. (2003). Fuzzy modeling for function points analysis. *Software Quality Journal*, 11(2) 149–166, June.

[31] Braz, M., Vergilio, S. (2004). Using fuzzy theory for effort estimation of object-oriented software. *In*: 16th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2004, p.196–201.

[32] Ryder, J. (1998). Fuzzy modeling of software effort prediction. *In*: IEEE Information Technology Conference, p. 53–56.

[33] J´unior, O., Farias, P., Belchior, A. (1999). A Fuzzy Model for Function Point Analysis to Development and Enhancement Project Assessments. *CLEI Electronic Journal*, 5:2.

[34] Mitra, S., Hayashi, Y. (2000). Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Networks*, 11 (3) 748–768.

[35] Berenji, R, H., Khedkar, P. (1992). Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3, 724–740.

[36] Cordon, O., Gomide, F., Herrera, F., Hofmann, F., Magdal na, L. (2004). Ten years of genetic fuzzy systems: Current framework and new trends. Fuzzy Sets and Systems, 141, 5–31.

[37] Schneider, G. and Winters Jason P. (2001). Applying Use Cases – A practical guide. 2nd ed. Addison-Wesley.

[38] Howard Demuth, Mark Beale. (2000). Neural Network Toolbox User's Guide.

[39] Belchior, A. D., Junior, O. S. L., Farias, P. (2003). Fuzzy modeling for function points analysis. *Software Quality Journal*, 11 (2) 149–166, June.

[40] Klir, G., and Folger, T. (1998). Fuzzy Sets, Uncertainty and Information. Prentice-Hall.

[41] Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis, *IEEE Transactions on Computers* 26 (12) 1182–1191.

[42] Mohagheghi, P., Anda, B. and Conradi, C. (2005). Effort Estimation of Use Cases for Incremental Large-Scale Software International Conference on Software Engineering (ICSE), St Louis, Missouri, USA, May 15-12, p. 303-311.

[43] N. N. (1998). Fuzzy Logic Benchmarks for MCUs , http:// www.fuzzytech.com/e_ftedbe.htm.

[44] Ashish Ghosh, Uma Shankar, B., Meher Saroj, K . (2009). A novel approach to neuro-fuzzy classification, Neural Networks, 22 (1) 100-109, January.