A New Method for Enhancing Software Effort Estimation by Using ANFIS-Based Approach

The-Anh Le[1,2], Quyet-Thang Huynh[1(✉)], Tran-Tuan-Nam Nguyen[1], and Minh-Hoa Tran Thi[3]

[1] School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
thanghq@soict.hust.edu.vn
[2] Faculty of Information Technology, People's Police University, Bacninh, Vietnam
[3] University of Social Sciences and Humanities, VNU Hanoi, Hanoi, Vietnam

Abstract. The accurate estimation of the effort and cost becomes one of the important issues of project management. There are some algorithmic and non-algorithmic techniques are already developed to tackle the challenges of estimation tools in software project management such as Bayes probability-based approach, classification and regression, semantic analysis of software requirements, artificial neural networks, fuzzy logic, and hybrid methods. These techniques are unable to satisfy the management of modern and dynamic software development process. The aim of this paper is to propose a method of estimation by using the fuzzy logic's related functions and the fuzzy algorithm of the Adaptive Neuro-Fuzzy Inference System (ANFIS) model to improve the accuracy of Functional Point Analysis (FPA) to estimate the cost and effort of software development. A tool called ALBRE is also developed to support the calculation of the proposed method. Experimental results show that the proposed method based on the ANFIS model produces positive results, with less errors. The accuracy of VAF increases by up to 80% compared to the method proposed by Albrecht in Function Point Counting Practices Manual 4.2.1 by considering the Mean Magnitude of Relative Error (MMRE).

Keywords: ALBRE · Effort estimation · ANFIS · Function point · FPA · Software estimation · Neural-fuzzy

1 Introduction

The software industry has been developing rapidly in recent times, as a result, minimizing the development cost becomes a subject of interest [1]. Software development effort estimation is the process of evaluating the cost required to develop software in its early stages of development. It goes without saying that the more accurate estimation is, the more likely the project will succeed. Nevertheless, calculating accurate estimations is a difficult task [2]. Therefore, many software cost estimation models have been researched

N.-S. Vo et al. (Eds.): INISCOM 2021, LNICST 379, pp. 195–210, 2021.
https://doi.org/10.1007/978-3-030-77424-0_16

and developed for decades such as COCOMO, COCOMO II, UCP, SLIM, and FPA [2–4]. The methods provide an algorithmic model in the form of steps and calculation formulas drawn from the study of historical data of projects.

In recent years, the application of FPA has also been strongly researched and developed in many companies, organizations, and software enterprises. In 2009, NESMA published a document that applied FPA to the later phase of the project at maintenance and improvement phases with many positive results [22]. David W. Russel et al. [20] designed a variant of FPA to estimate software costs. The value of 14 GSC is adjusted to suit the specific settings. Archana Srivastava [21] changed the way to calculate VAF by adding the end-user programming with the same calculation as the VAF but with 16 GSC elements and new weighting with 0–5 weighting range. This has enabled freelance programmers or those who want to develop their own applications to estimate the exact cost of the application without having to know the entire system [21]. E. Praynlin and P. Latha [25] applied ANFIS to COCOMO estimation method of effort multiplier, the goal is to select the appropriate membership function through price MMRE treatment. The comparative results concluded that the paper selected the trapezoid function to give the lowest MMRE [25]. Shivakumar [23] proposed to apply ANFIS to the Class Point method to estimate software costs. The authors in [27] proposed the method that applied the modified ANFIS to improve effort estimation results and achieve accuracy up to 96% with training data and 89% with test data. Al-Hajri et al. [24] set up a new system of FP parameters using neural networks. The results were quite accurate although the correlation was still unsatisfactory with MMRE above 100%, derived from the wide variety of data points with much noise. Empirical studies by Abran and Robillard [26] show each relationship between the main part of FPA and effort. They proposed a model to use FPA in combination with ANFIS to make each relationship with effort. Therefore, effort estimation is supposed to be fateful because it is an infrequent, unique task with underestimation bias and different goals [3]. There are some factors that may impact the efficiency of software engineering such as product complexity, quality requirements, time pressure, process capability, team distribution, interrupts, feature churn, tools, and programming language [4]. The main purpose of this paper is to design an early effort estimating method based on FPA using ANFIS with customized Albrecht dataset to improve Function Point Counting Practices Manual 4.3.1 equation [6], compare between ANFIS model 6 membership functions to choose the best one and calculate the Value Adjustment Factor (VAF) of a specific software project with FPA by using results of the research.

The rest of this paper is organized as follows: In Sect. 2, the literature review will be presented. The proposed model and dataset description will be discussed in Sect. 3. In Sect. 4 and 5, the results and discussion of the proposed model will be presented.

2 Literature Review

2.1 Function Points Analysis (FPA)

Function points are proposed by a team under Allan Albrecht at IBM which is published in a book in 1979 [5, 10]. Albrecht states that function points are "an effective relative measure of function value delivered to our customer" [5]. Researchers have also found a strong relationship between the amount of function points and work effort. Function Points Analysis (FPA) is a common method for effort estimation [17]. It has many objectives and benefits that can be used as a vehicle to estimate the cost and resources during the phases of software development and maintenance.

Step 1. Determining the Unadjusted Function Point (UFP).

The UFP is a term used to describe the information of the specific countable functionality of the project or application to the user. The specific user functionality is evaluated what is delivered by the application, not how it is delivered. Only user-requested and defined components are counted with two types of functions: data types (Data Functions) and transactional types (Transactional Functions). Each of them is evaluated by assigning one of these three functional complexities: Low, Average and High. This calculation also depends on the quantity of each types-data and they will be translated to UFP. Table 1 will interpret this transformation. The total of these five UFPs will determine the overall UFP of this step.

Table 1. Transformation of functions to UFP

No	Type of functions		Functional complexity		
			Low	Average	High
1	*Transactional Functions*	External Input	3	4	6
2		External Output	4	5	7
3		External Inquiry	3	4	6
4	*Data Functions*	Internal Logical File	7	10	13
5		External Interface File	5	7	10

Step 2. Determining the Value Adjustment Factor (VAF).

The UFP will be adjusted by 14 technical factors or 14 General System Characteristics (GSC). These GSC will be interpreted in the Table 2 below with their brief description [5, 10].

$$VAF = (TDI \times 0,01) + 0,65 \tag{1}$$

There are 6 system influences, rated from to 5 to evaluate the weight of each GSC. These weights will be described in Table 3. In this research, we believe our VAF calculation will be at higher accuracy than the Eq. (2). The method will be demonstrated in later sections.

Table 2. General system characteristics

	General system characteristics	Brief description
1	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2	Distributed data processing	How are distributed data and processing functions handled?
3	Performance	Did the user require response time or throughput?
4	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6	On-Line data entry	What percentage of the information is entered On-Line?
7	End-user efficiency	Was the application designed for end-user efficiency?
8	On-Line update	How many ILF's are updated by On-Line transactions?
9	Complex processing	Does the application have extensive logical or mathematical processing?
10	Reusability	Was the application developed to meet one or many user's needs?
11	Installation ease	How difficult are conversion and installation?
12	Operational ease	How effective and/or automated are start-up, backup, and recovery procedures?
13	Multiple sites	Was the application specifically designed, developed and supported to be installed and multiple sites for multiple organizations?
14	Facilitate change	Was the application specifically designed, developed and supported to facilitate change?

Step 3. Overall calculation – determining the Development project Function Point count (DFP).

The final calculation of the FPA process can be calculated by the following equation:

$$\mathrm{DFP} = \mathrm{UFP} \times \mathrm{VAF} \tag{2}$$

Where DFP is the Development project Function Point count, UFP is Unadjusted Function Point and VAF is Value Adjustment Factor.

The FPA evaluation method is based on the opinion of the end-user, outside domain of the considering system, hence, the FPA is not depended on the technology used in the

Table 3. Degree of influence

Score as	System influence
0	Not present or no influence
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Strong influence throughout

system and not required the detail description of the system. It can be applied in the early stage of system development which based only on the system specification from the user so the project can be estimated and planned correctly. This will help the project manager in terms of cost management and control [8]. However, the FPA cannot be calculated automatically because of the end-user opinion-based method. Every input of the process is depended on the sense of the user and sometimes it is vague and imprecise. This can be adjusted by the user as an expert or a specialist. Thus, the FPA is a strong method for the design of the function-oriented system but it is not recommended for object-oriented systems design or scientific system with complicated algorithms and calculations [8].

2.2 Fuzzy Logic Membership Function

2.2.1 Classical Logic, Classical Set and Its Problems

Classical logic is used in the daily life of a human with only 2 values in opposition such as "no" – "yes", "0" – "1", "true" – "false", etc. In mathematics, the classical logic comes with the classical set. There are only 2 states of an element, "belong" or "not belong" to a specific set which is presented by the symbol $\in$ and $\notin$. By considering a membership function with an element x and a set A, the relationship of this logic will be interpreted by the following equation.

$$\mu A(x) = \begin{cases} 1 \; if \; x \in A \\ 0 \; if \; x \notin A \end{cases} 0 \leq \mu A(x) \leq 1 \tag{3}$$

Hence, classical logic is absolute. However, if the other number sets, for instance, a set B is a set that contains rational numbers which are less than 9 and a set B' is a set that contain also rational numbers but much less than 9 or if the set of tall, average and short people are considered, there will be a problem. Is 1.51 m average enough, is 1.72 m tall enough. At this time, the classical logic is useless due to its limitation can never be cover all of the life situations of human. Human technology such as self-drive technology, artificial intelligence or human creative activities, the classical logic cannot support them to operate. Especially the machine learning technology, the knowledge is plenty of vague, uncertain situations with fuzziness in semantics and the machine needs an absoluteness. Thus, they cannot understand. That is the reason why fuzzy logic is

needed to confront fuzzy problems. With fuzzy logic, the problem of the air-conditional, self-drive car and machine learning is resolved.

2.2.2 Fuzzy Logic, Fuzzy Set

Fuzzy logic is used to adjust the absoluteness of classical logic. If the classical logic states the certain theses as 100% or 0%, the fuzzy logic states the uncertain. Fuzzy logic is used to solve difficult problems that cannot be handled by a mathematical model due to its complex or unfeasible to construct. It is also applied to reduce the complexity of existing solutions and increase the accessibility of direct hypothesis [16]. The development of software has been measured by parameters that possess a certain level of fuzziness [18]. The fuzzy logic is applied to solve the problems that are inherent in existing effort estimation techniques [19].

In general, we can only give probabilistic rules for certain cases that are relatively perfect and cannot give only one rule to just one case. Now the rules are used together for continuous overlapping cases and what people do is perfect, round or approximate them. Combining rules can be typical when cases are repeated with the same or approximate state. People can do that through the flexibility of the language used to create rules as well as abstract thinking, logic coming from the human brain. However, that does not mean that we cannot use mathematics to model fuzzy logic theory. As shown in Sect. 2.2.1 on classical logic, we can use the membership functions to represent fuzzy logic theory. Membership function associates each point in the fuzzy set a real number in the interval [0,1] is called degree or grade of membership. Several types of fuzzy membership functions including Triangular, Gaussian bell, Trapezoidal, Sigma, S function and the function of Z. Among them, Triangular, Gaussian, and Trapezoidal are commonly used in software estimation models. These equations below will interpret the general type of membership function [11, 14].

$$\mu(x) = \begin{cases} \frac{x-l_1}{m-l_1}, & x \in [l_1, m] \\ \frac{l_2-x}{l_2-m}, & x \in [m, l_2] \\ 0, & x \notin [l_1, l_2] \end{cases} \tag{4}$$

Where l_1 is the left boundary value of membership, m is the value of capital and l_2 is the right limit membership value ($l_1 < \mathrm{m} < l_2$). Since all information contained in a fuzzy set is described by its membership function, it is useful to develop a lexicon of terms to describe various special features of this function [9]. This is a firm base for ANFIS training and represents the sense of "fuzzy", modeling from reality to logic model.

2.3 Anfis

ANFIS is an integrated model of Fuzzy Inference System (FIS) with Neural networks [12]. A generic rule in a Sugeno fuzzy pattern has the form: If Input1 = x and Input 2 = y, then output is z = ax + by + c [12].

It needs 7 constrains:

Single output, defuzzied by the average center of weight method.

Membership functions of output must be the same type, linear or constant.

The number of output membership function and the number of the rule must be equal.

Each rule has its particular weight.

FIS structure will have an error if these constraints above are not satisfied.

Fuzzy membership functions and defuzzied method cannot be created by the user.

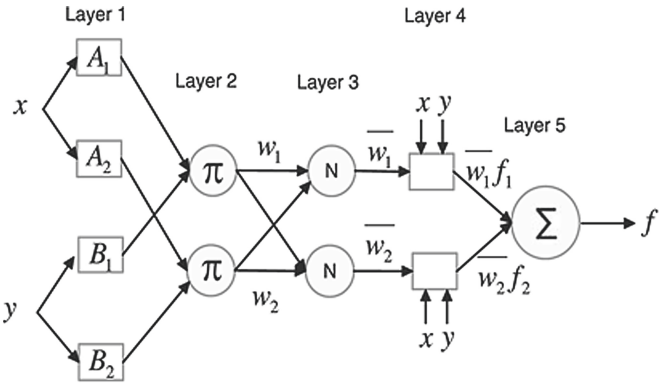

Fig. 1. ANFIS structure

ANFIS is a hybrid supervised method that leverages a hybrid learning algorithm to determine the parameters for fuzzy inference systems. It utilizes the least-squares method and propagation gradient descent method which is used for training FIS membership function parameters to investigate the given training data set. ANFIS can be executed by using an optional argument for model validation. This is called checking the model for overfitting. The argument used for this is called checking data set. The key is finding the FIS rule base. FIS converts human knowledge into a rule base in order to maximize performance and minimize the model error output [13]. ANFIS structure is described in Fig. 1.

The function of each node is described as follows:

Layer 1 Each node k in a square layer node with the function:

$$O_k^1 = \mu_{A_k}(x) \quad k = 1, 2 \tag{5}$$

$$O_k^1 = \mu_{B_{k-2}}(y) \quad k = 3, 4 \tag{6}$$

Where x, y is the input of node k, and A_k, B_k is a fuzzy set of rating the node function. O_k^1 is a membership function that determines the degree of membership B_{k-2} and A_k.

Layer 2 Each node labeled Π is a multiplier value, used AND operator to fuzzily inputs.

$$O_k^2 = w_k = \mu_{A_k}(x) \times \mu_{B_{k-2}}(y) \tag{7}$$

Layer 3 Normalized firing strength. Each node labeled N is calculated by taking the ratio of the k-th rule firing strength of the overall total of all rules firing strength.

$$O_k^4 = \overline{w}_k f_k = \overline{w}_k CD_{ik} \tag{8}$$

Layer 4 Setting the consequent parameters. Each node k in a square layer node with function:

$$O_k^4 = \overline{w}_k f_k = \overline{w}_k (p_k x + q_k y + r_k) \quad k = 1, 2 \tag{9}$$

where $\overline{w}_k$ is the output of Layer 3 and $p_k x + q_k y + r_k$ is the consequent parameters.

Layer 5 There is a single node with symbol Σ which calculates the overall output of the incoming signal. This stage is also called defuzzification.

$$O_k^5 = \sum_k \overline{w}_k f_k = \frac{\sum_k w_k f_k}{\sum_k w_k} \tag{10}$$

3 The Proposed Approach

3.1 Approach Description

According to Eq. (1), the calculation and estimation of the project cost based on FPA estimation depend on the two parameters UFP and VAF. These are the two factors that can cause relatively high errors in the cost and effort estimation. In this study, the VAF parameter will be fine-tuned to minimize errors through ANFIS in order to reduce the cost bias. The VAF value is calculated based on 14 GSC factors with the weights of DI influence (0–5). These DIs will also be tuned for accuracy in this study. To fine-tune the deviation of VAF, we use the adaptive neural-fuzzy network ANFIS to blur the VAF value calculated by Eq. (1). At the same time, the values of DI will also be improved through the introduction of a set of questions that are more semantically clear than the given CPM documents and Albrecht's research results.

The approaching model is illustrated in Fig. 2 below.

ANFIS is suitable for use in the proposal of this study. ANFIS learns from historical data sets and adjusts to minimize the factors that cause errors between the training model and real data of organizations. Users can provide the parameters of previous projects as input to ANFIS and from the training results, they can consider and select the most suitable results. The strength of ANFIS over conventional neural networks is its adaptability. ANFIS does not have a fixed model structure as a basis for the attributes of the variables in the system so users no need to know about the core technical issues nor waste time and effort to monitor the training process. However, ANFIS, like other neural networks, requires data from the past that is large enough to underpin an effective learning process. The greater the amount of data to learn, the more ANFIS will have accurate and closest calculations to reality.

The objective of the MMRE value achieved is minimum. For MMRE, it is calculated through MRE according to the following two-equation [7]. The relative error compared

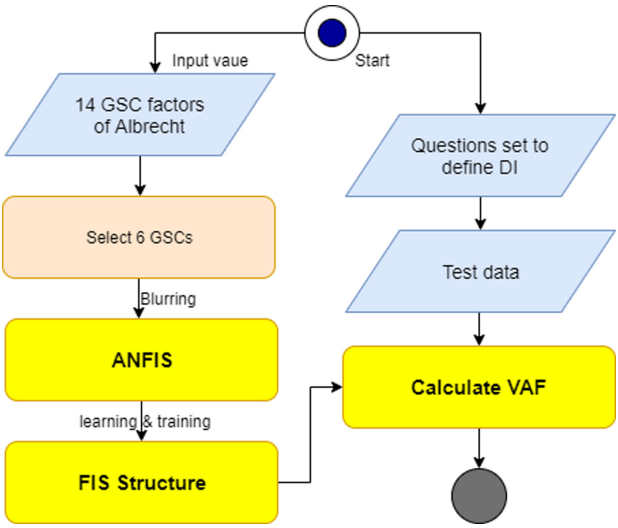

Fig. 2. Workflow of Software Effort Estimation by using ANFIS-based Approach

to the actual (5) and (6) average relative error. The algorithm for this method consists of 4 steps.

$$MRE_i = \frac{|Actual\ Effort_i - Estimated\ Effort_i|}{Actual\ Effort_i} \tag{11}$$

$$MMRE = \frac{1}{n}\sum_i^n MRE_i \tag{12}$$

Step 1. Reduce the Number of Input Values from 14 to 6 GSC
The reduction in numbers helps reduce ANFIS execution time as well as reduce resource utilization and exceed the computational power of the system. However, the VAF results remain the same to ensure that the inference from Eq. (1) does not change when ANFIS is trained. In fact, training with a data set of 5 GSCs takes only 15 min but when increasing to 6, the training time is 12 times. Based on recent research results, UCP estimation when comparing each GSC and based on practical experience, we choose 6 GSCs: 2, 3, 7, 9, 10 and 13. They are Distributed Data Processing (GSC2), Performance (GSC3), End-User Performance (GSC7), Complex Processing (GSC9), Reusability (GSC10), and Multi-Platform (GSC13). In the UCP estimation, the 13 factors and their weights, outlined in Table 4 below, affect the TCF value of a project under UCP. The 6 factors that are relevant and compared with the 6 GSCs are in bold.

Table 4. Technical Complexity Factor (TCF)

Factor	Description	Weight
T1	**Distributed system**	**2.0**
T2	**Response time/performance objectives**	**1.0**
T3	**End-user efficiency**	**1.0**
T4	**Internal processing complexity**	**1.0**
T5	**Code reusability**	**1.0**
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	**Portability to other platforms**	**2.0**
T9	System maintenance	1.0
T10	Concurrent/parallel processing	1.0
T11	Includes special security objectives	1.0
T12	Provides direct access for third parties	1.0
T13	Special user training facilities are required	1.0

Step 2. Fuzzy 6 DI Values of 6 GSCs

6 DI values and VAF results calculated according to Eq. (1) will be put into ANFIS to implement fuzzy and training with different related functions according to 3 fuzzy language variables "small", "medium" and "large". From the recommendations and scoring criteria of CPM, this study will provide a more explicit set of questions and answers to assist the estimator. At the same time, the responses are weighted according to the CPM but will be relatively blurred through the statistical method of Albrecht's project data sets. The blurring is done by statistics across 240 projects, considering the percentage of occurrences of influences from 0 to 5. The calculation is rounded to 2 decimal places and the compensation is selected (Table 5).

Table 5. Statistics of the frequency offset ratio of DI of 6 GSCs

DI/GSC	2	3	7	9	10	13
0	0,98	0,97	0,92	0,83	0,83	0,83
1	0,82	0,94	0,88	0,87	0,78	0,63
2	0,7	0,76	0,69	0,78	0,76	0,75
3	0,69	0,74	0,62	0,72	0,8	0,88
4	0,85	0,79	0,88	0,86	0,88	0,93
5	0,97	0,8	0,99	0,93	0,94	0,98

$$\text{DI} = \text{Point i is given by CPM} \times \text{d} \tag{13}$$

Where d is % of the compensation of the occurrence of point I.

Step 3. Estimate VAF Values
From the FIS data of ANFIS training in step 2 and the values in the data set, VAF values will be estimated using a new method. The ANFIS input values are 18 fuzzy sets of 6 GSCs and 3 linguistic variables. The output is the VAF value required. Thus, ANFIS will develop a total of 36 or 729 rules according to the "if-then" format as shown in the Table 6 below.

Table 6. The truth table of the ruleset

Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Output
Small	Small	Small	Small	Small	Small	Small
…	…	…	…	…	…	…
Small	Small	Large	Large	Small	Small	Medium
…	…	…	…	…	…	…
Large	Large	Large	Large	Large	Large	Large

Step 4. Use the VAF to Calculate Effort
After the VAF value has been calculated and with the UFP value available, we proceed to calculate the DPF value according to Eq. (2).

$$Effort = -13,39 + 0,0545 \times DFP \tag{14}$$

3.2 Dataset Description

Albrecht dataset is a famous and reliable dataset. It is an excel file sized 15 × 250 contained 240 actual software projects, estimated follow FPA method. Thus, it is included 14 columns of 14 GSCs and 1 column of VAF-calculated by Eq. (1). ANFIS requires a dataset for training as input and export the result to a test dataset as an output. The input is created by separating the original dataset–Albrecht dataset sized 15 × 240 into 3 types of a smaller train-test dataset, named A, B, and C, which will be interpreted as in Table 7.

Each dataset is considered as a metric of m x n. The training dataset will take more than the test dataset 1 column of VAF. The purpose of the train datasets is to 'teach' ANFIS with their input (GSCs) and output (VAF) then apply the knowledge (FIS) to calculate the output VAF with given GSCs in the test dataset. Reminded that the test and FIS metric columns must be exactly the same and the FIS will calculate only 1 project as its size of the row. These VAFs are the final results of this study that we are expected. Grouping the original 14 GSCs into smaller groups is considered because of the problem's complexity and computer capability issues. During the experiment,

Table 7. Dataset separation

#	Dataset	Size	Type	GSCs
1	A	6 × 70	Train	1–5
2		5 × 40	Test	
3		6 × 70	Train	6–10
4		5 × 40	Test	
5		6 × 70	Train	11–14
6		5 × 40	Test	
7	B	7 × 70	Train	2, 3, 7, 9, 10, 13
8		6 × 50	Test	
9	C	7 × 120	Train	
10		6 × 120	Test	

ANFIS cannot generate rules or 'learn' if the number of columns is greater than 8 or 9 due to the fact that a more powerful computer is required and the training time will also significantly increase, for example: training a dataset with 5 columns took 15 min, whereas training a dataset with 6 columns took 3 h, which means it possibly took up to 4 h to train with 7 columns. On the other hand, 2–4 grouping-columns are considered too few inputs for the result to be accurate. Therefore, 5 and 6 grouping-column has been chosen for this study (see step 2 in Sect. 3.1, Table 4).

In Table 7, the TCF and FPA have correlative factors with GSC2-T1, GSC3-T2, GSC7-T3, GSC9-T4, GSC10-T5, GSC13-T8, and factor T1 and T8 have the highest weight of 2 and the other weight of 1. It is believed that grouping 6 above GSCs can give out highly accurate results. As the pairing correlative factors above, there would be a considered pair: GSC11-T6, GSC12-T7 each has a weight of 0,5. Therefore, GSC11 and GSC12 have been eliminated from the grouping. It is reasonable to eliminate the remaining 6 GSCs, by studying the Albrecht dataset 150/240, it means that 62,5% projects are rated 0 or 1 with GSC14. So, GSC14 has also been eliminated. GSC6 and GSC8 are mostly applicable to online applications, which do not seem to be significant to every application, so we also eliminated these 2 GSCs. Besides, GSC1, GSC4, GSC5 can be chosen to replace any GSC from 6 chosen GSCs above (except GSC2 and GSC13). Therefore 6 GSCs that have been mentioned will be chosen as the basis of the dataset for this study (Fig. 3).

3.3 ANFIS Training

Firstly, in this study, every 10 datasets would be the input of ANFIS for training and testing with 6 different types of membership function, with 40, 60, 80, 90 and 100 epochs. Secondly, ANFIS will 'learn' through these training datasets then will generate rules and membership function parameters. Finally, ANFIS finishes training and generates the FIS file by grid partitioning method. The file was used for estimating the projects, which are

numbers of the row in the test dataset and give the result as VAF value. It takes 3 h to obtain the expectation in the training. In the first stage, we conducted ANFIS training by using data from the previous stage. The training process allows the system to adjust the parameters as input/output. This process stops when the number of epochs is reached, or the number of error-rates achieved. In the second stage, a vector is created with the number of dimensions N where N is the number of membership functions that contains the parameters of membership functions. In each iteration, one of the parameters of the membership function will be updated [15]. Based on what ANFIS has studied, it will generate the VAF result with each project (row) in the test dataset. The FIS structure will show the result of the ANFIS training process, including the model of 36 fuzzy rules and a model of input and output of the structure. The simulation is conducted in MATLAB R2016a, Windows 10, 4 GB of RAM and INTEL Pentium G4400 3.3 GHz CPU. These following figures and tables will interpret the FIS structure (Table 8).

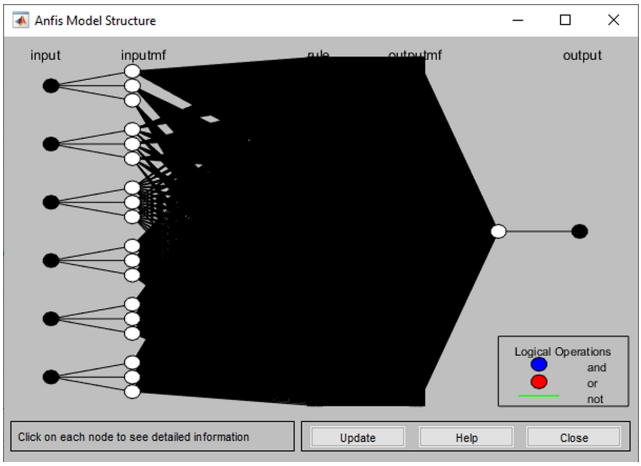

Fig. 3. FIS structure

3.4 ALBRE- Tool for Effort Estimation

ALBRE application is developed on MATLAB R2016a IDE. It is easy to deploy in an enterprise environment, company, small and medium organization. Individuals can also use the application to self-estimate the cost but requires a certain knowledge of software engineering. The main functions will be presented in this section: VAF Calculation, UFP Calculation, DI Calculation, Question set. Users can enter data into numeric fields in a variety of ways and perform calculations. In particular, entering data through the question set will have its own interface.

4 Results and Discussion

Evaluating the accuracy of estimation can be done by comparing the results of the production effort and the actual effort to calculate MRE (Magnitude of Relative Error) [7].

Table 8. The overall data of the training process with Gauss membership function

FIS structure	Number of inputs	6
	Number of outputs	1
	Type of membership function input	Gauss
	Type of membership function output	Linear
	Total fuzzy rule	729
Training structure	Number of epochs	80
	Error tolerance	0
Learning method		Hybrid
Training time		3 h

MRE can be calculated by the following equation where i is the number of observations:

$$MRE_i = \frac{|Actual\ Effort_i - Estimated\ Effort_i|}{Actual\ Effort_i} \tag{15}$$

The MRE value is calculated for each observation whose effort is estimated. The aggregation of MRE over multiple observations (N) can be achieved through the mean MMRE as follows:

$$MMRE = \frac{1}{n}\sum_{i}^{n} MRE_i \tag{16}$$

The Table 9 below presents the MMRE values for the set of inputs Tripmf and Gbellmf which has a very low MMRE value compared to all other membership functions with a group of 5 and 6 columns dataset.

Table 9. MMRE for various membership function with a group of 5 and 6 columns dataset

Membership function	MMRE (group 5)	Epochs	MMRE (group 6)	Epochs
trimf	0.1203	40	0.2309	40
trapmf	0.2965	60	0.2862	60
gbellmf	0.1259	80	0.4594	100
gaussmf	0.1183	100	0.2183	80
gauss2mf	0.2767	90	0.3690	90
pimf	–	–	0.3735	100

MMRE value of 0.1183 and 0.2183 for the Gaussian curve membership function is the lowest among all the membership functions of 2 groups prediction is based on available data. If there are errors in the actual data, then the result's accuracy will be

affected. The fuzzification is based on certain membership functions. The selection of a particular membership function depends on the nature of data value to be used. If the selected membership function is incorrect, the input fuzzy data will be wrong and as a result, the output fuzzy data will also be wrong and the defuzzied output data will be imprecise and the error rate will be very high. Hence the selection of membership functions plays a very important role in the Estimation model.

5 Conclusion and Future Work

Software effort estimation involves dealing with the uncertainty of inputs. ANFIS is able to handle this uncertainty, and selecting correct membership functions plays a crucial role in effort estimation. Comparison between membership functions such as the Gaussian curve, Gaussian combination membership, and generalized bell-shaped membership, Trapezoidal membership and Triangular membership was done. The results show the Gaussian curve membership function has produced the smallest MMRE value compared to the other member function. The purpose of this study is to identify the membership function is to be used in ANFIS and give the most accurate effort estimation and claim the improvement of the Function Point Counting Practices Manual 4.2.1 equation (Eq. (2)). The proposed method is effective in estimating the cost development of new software. However, the current study was limited by computer capability issues and still has much remained to be improved. The study can point out the representation of this 6 selected GCSs but still cannot be 100% precise about the limitation of the random columns above. MATLAB is not a good platform for developing interfaces and applications, so it is very limited in targeting users, although it is heavily used in science or in a fuzzy neural network. by the simulation capabilities.

For future works, similar studies can be done to estimate software cost based on fuzzy logic and neural network by using ANFIS such as:

1) Analyzing the performance of the model by varying the number of epochs, the number of membership functions.
2) Analysis can also be done using an artificially generated dataset.
3) Another improvement for training by ANFIS to gain a higher VAF accuracy.

Acknowledgments. This research is funded by Hanoi University of Science and Technology under Grant number T2018-TĐ-009.

References

1. Grimstad, S., Jorgensen, M.: The impact of irrelevant information on estimates of software development effort. In: Proceedings of the 2007 Australian Software Engineering Conference, ASWEC 2007, pp. 359–368. IEEE Computer Society (2007)
2. Chulani, S.: Bayesian analysis of software cost and quality models. Ph.D. Dissertation, University of Southern California, Los Angeles (1999)

3. Sinhal, A., Verma, B.: Software development effort estimation: a review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **III**(6), 1120–1135 (2013)
4. Bourque, P., Fairley, R., (eds.): SWEBOK 3.0: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Press (2014)
5. Albrecht, A.J.: Measuring application development productivity. In: Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, 14–17 October, pp. 83–92. IBM Corporation (1979)
6. Timp, A.: Function point counting practices manual. International Function Point Users Group (2010). Release 4.3.1. ISBN 978-0-9753783-4-2
7. Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, L.: A simulation study of the model evaluation criterion MMRE. IEEE Trans. Softw. Eng. **29**, 985–995 (2003)
8. Thang, H.Q.: Software Engineering Economics. Hanoi University of Science and Technology Publishing House (2016). (in Vietnamese)
9. Ross, T.J.: Fuzzy Logic with Engineering Applications, 3 ed, 606 p. Wiley (2010)
10. Jones, C.: Software economics and function point metrics. In: Thirty years of IFPUG Progress. http://www.ifpug.org/wp-content/uploads/2017
11. Kaur, M.: A fuzzy logic approach to software development effort estimation. Int. J. Adv. Res. Dev. **3**(1), 125–127 (2018). ISSN 2455-4030
12. Iraji, M.S.: Hypermedia web software effort estimate with adaptive neuro fuzzy inference system. J. Theoret. Appl. Inf. Technol. **93**(1), 133–142 (2016). ISSN 1992-8645
13. Bedi, R.P.S., Singh, A.: Software cost estimation using fuzzy logic technique. Indian J. Sci. Technol. **10**(3), 1–5 (2017)
14. Batra, G., Trivedi, M.: A fuzzy approach for software effort estimation. Int. J. Cybern. Inform. (IJCI) **2**(1), 9–15 (2013)
15. Sweta, K., Shashankar, P.: Comparison and analysis of different software cost estimation methods. Int. J. Adv. Comput. Sci. Appl. (IJACSA) **4**(1), 153–157 (2013)
16. Razaz, M., King, J.: Introduction to Fuzzy Logic - Information Systems - Signal and Image Processing Group (2004). http://www.sys.uea.ac.uk/king/restricted/boads/
17. Symons, C.R.: Function point analysis: difficulties and improvements. IEEE Trans. Softw. Eng. **14**(1), 2–11 (1988)
18. Moon Ting, S., Ling, T., Phang, K., Liew, C., Man, P.: Enhanced software development effort and cost estimation using fuzzy logic model. Malaysian J. Comput. Sci. **20**(2), 199–207 (2007)
19. Agustin Gutierrez, T., Cornelio Yanez, M., Pasquier, J.L.: Software development effort estimation using fuzzy logic: a case study. In: Proceedings of the Sixth Mexican International Conference on Computer Science (ENC 2005) (2005)
20. Russell, D.: A metric for rating the effectiveness of industrial automation systems using a derivative of function point analysis. Robot. Comput.-Integr. Manuf. **26**(6), 551–557 (2010)
21. Srivastava, A., Qamar Abbas, S., Singh, S.K.: Enhancement in function point analysis. Int. J. Softw. Eng. Appl. (IJSEA) **3**(6), 129–136 (2012)
22. NESMA: Function point analysis for software enhancement guidelines version 2.2.1 (2009)
23. Shivakumar, N., Balaji, N., Ananthakumar, K.: A neuro fuzzy algorithm to compute software effort estimation. Glob. J. Comput. Sci. Technol.: C, **16**(1) (2016). Version 1.0
24. Mokri, F.D., Molani, M.: Software cost estimation using adaptive neuro fuzzy inference system. Int. J. Acad. Res. Comput. Eng. **1**(1), 34–39 (2016)
25. Praynlin, E., Latha, P.: Estimating development effort of software projects using ANFIS. In: International Conference on Recent Trends in Computational Methods. Communication and Controls, Ongole, Andhra Pradash, India, pp. 15–20 (2012)
26. Abran, A., Robillard, P.N.: Function points analysis: an empirical study of its measurement processes. IEEE Trans. Softw. Eng. **22**(12), 895–910 (1996). https://doi.org/10.1109/32.553638
27. Mokri, F.D., Molani, M.: Software cost estimation using adaptive neuro fuzzy inference system. Inter. J. Acad. Res. Comp. Eng. **1**(1), 34–39 (2016)