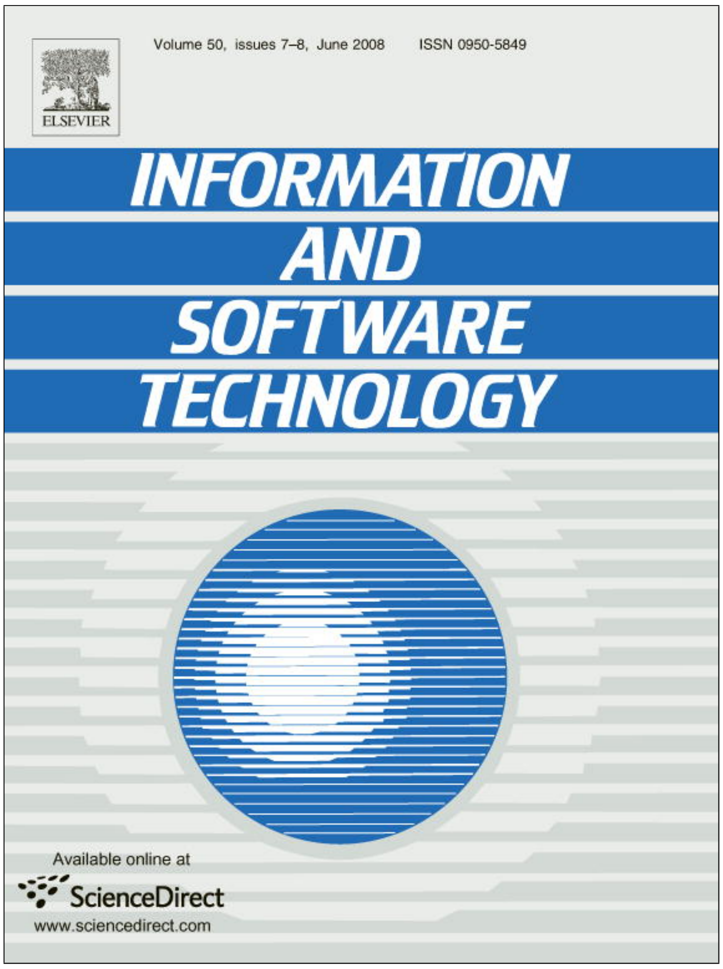

Volume 50, issues 7–8, June 2008
ISSN 0950-5849
ELSEVIER
INFORMATION AND SOFTWARE TECHNOLOGY
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com

ScienceDirect

Information and Software Technology 50 (2008) 670–683

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

A new calibration for Function Point complexity weights

Wei Xia [a], Luiz Fernando Capretz [b,*,1], Danny Ho [c], Faheem Ahmed [d]

[a] HSBC Bank Canada, IT Department, Vancouver, BC, Canada

[b] University of Western Ontario, Department of Electrical and Computer Engineering, London, Ont., Canada

[c] NFA Estimation Inc., London, Ont., Canada

[d] United Arab Emirates University, College of Information Technology, Al-Ain, United Arab Emirates

Received 27 March 2007; received in revised form 25 June 2007; accepted 17 July 2007
Available online 6 August 2007

Abstract

Function Point (FP) is a useful software metric that was first proposed 25 years ago, since then, it has steadily evolved into a functional size metric consolidated in the well-accepted Standardized International Function Point Users Group (IFPUG) Counting Practices Manual – version 4.2. While software development industry has grown rapidly, the weight values assigned to count standard FP still remain same, which raise critical questions about the validity of the weight values. In this paper, we discuss the concepts of calibrating Function Point, whose aims are to estimate a more accurate software size that fits for specific software application, to reflect software industry trend, and to improve the cost estimation of software projects. 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 repository release 8 shows a 22% accuracy improvement of mean magnitude relative error (MMRE) in software effort estimation after calibration.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Neural network; Fuzzy logic; Function Point; Function point analysis; Software size estimation

1. Introduction

Software effort estimation is crucial in software engineering. Accurate software estimation is critical for project success. If done correctly, resources are allocated appropriately, but on the other hand, an inaccurate estimation can ruin even a small software company. Consequently, many models for estimating software development effort have been proposed and some of them are arguably the most popular such as COCOMO [1], SLIM [2]. These models can be considered algorithmic models. The pre-specified formulas for estimating development efforts in those models are calibrated from historical data and all these models utilize software size as a key factor to estimate effort. FP is an ideal software size metric to estimate cost since it can be used in the early development phase, such as requirement, measures the software functional size from user's view, and is programming language independent [3].

Although software engineering has been influenced by a number of ideas from different fields [4], 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 [5]. 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.

Nevertheless, the weight values of FPA method are said to reflect the functional size of software. They have been

[*] Corresponding author. Tel.: +1 519 6612111; fax: +1 519 8502436.
E-mail address: lcapretz@eng.uwo.ca (L.F. Capretz).

[1] Dr. L. F. Capretz is currently spending his sabbatical leave with the Department of Computer Science at the University of Sharjah, in the United Arab Emirates.

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2007.07.004

introduced in 1979, and have been applied universally. In contrast, software development methods have evolved steadily for the last 20 years, and today's software differs drastically from what it was over two decades ago, for example the object-oriented paradigm has incorporated into mainstream of software development. These developments have triggered the creation of Object-Oriented Function Points (OOFP) [6], which substantially improved the accuracy of estimation of object-oriented systems using FP [7]. Algorithmic effort prediction models are limited by their inability to cope with vagueness and imprecision in the early stages of the software life cycle. Srinivasan and Fisher [8] illustrate a neural network learning approach to estimate software development effort known as *Back-propagation*. They indicate possible advantages of the approach relative to traditional models, but also point out limitations that motivate continued research. Furthermore, MacDonell [9] also considers the applicability of fuzzy logic modelling methods to the task of software source code sizing, and suggests that fuzzy predictive models can outperform their traditional regression-based counterparts, particularly with refinement using data and knowledge. The theory of fuzzy sets [10] as been successfully applied to other models of software cost estimation, such as f-COCOMO [11] and NF-COCOMO [12], is sufficiently general to be extended to the well-known FP method.

1.1. Survey of related work

Finnie et al. [13] reported research on the combination of machine learning approach with FP, they compared three estimation techniques using FP as an estimate of system size. The models considered are based on regression analysis, artificial neural networks and case-based reasoning. Although regression models performed poorly on the given data set, the authors observed that both artificial neural networks and case-based reasoning appeared to be valuable for software estimation models. Hence, they concluded that case-based reasoning is appealing because of its similarity to the expert judgement approach and for its potential in supporting human judgement.

Yau and Tsoi [14] introduce a fuzzified FP analysis model to help software size estimators to express their judgment and use fuzzy B-spline membership function to derive their assessment values. The weak point of this work is that they used limited in-house software to validate their model, which brings a great limitation regarding the validation of their model. Lima et al. [15] also propose the use of concepts and properties from fuzzy set theory to extend FP analysis into a fuzzy FP analysis, a prototype that automates the calculation of FPs using the fuzzified model was created, but the calibration was done using a small database comprised of legacy systems developed mainly in Natural 2, Microsoft Access and Microsoft Visual Basic, which compromises this work's generality. Al-Hajri et al. [16] establish a new FP weight system using artificial neural network. Like our work, in order to validate their model, they also used the data set provided by the ISBSG. In their research, tables gathered with the training methods from neural networks replaced the original complexity table. Their results are quite accurate, although the correlation is still unsatisfactory with MMRE over 100%, which originates from the wide variation of data points with many outliers.

Neural network technique is based on the principle of learning from historical data. The neural network is trained with a series of inputs and desired outputs from the training data set [17]. After the training is complete, new inputs are presented to the neural network to predict the corresponding outputs. Fuzzy logic is a technique to make rational decisions in an environment of uncertainty and imprecision [18,19]. It is rich in representing human linguistic ability with the terms such as fuzzy logic and fuzzy rules [20]. Once the concept of fuzzy logic is incorporated into the neural network, the result is a neuro-fuzzy system with learning capacity that combines the advantages of both techniques [21].

Previous research projects have indicated that the combination of machine learning approaches and algorithmic models yields a more accurate prediction of software costs and effort, which is competitive with traditional algorithmic estimators. However, our proposed neuro-fuzzy model goes even further: it is a unique combination of statistical analysis, neural networks and fuzzy logic. Specifically, we obtained an equation from statistical analysis, 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 Function 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. A similar Neuro-Fuzzy approach was applied on COCOMO model [12] and the improvement in software effort estimation proves this approach's validity. In Xishi et al. [12], the fuzzy logic part defines the COCOMO cost driver's membership function and neural network is used to train the weight values of the COCOMO cost drivers.

Abran and Robillard's empirical study [22] demonstrates the clear relationship between FPA's primary component and Work-Effort. In our model, an equation between Unadjusted Function Points and Work effort is used to train the neural network and in estimating the effort. Kralj et al. [23] identified the Function Point Analysis method deficiency of upper boundaries in the rating complexity process that carries the same essences of our research questions. Kralj et al. proposed an improved FPA method by plugging in a dividing process to resolve this problem while we proposed a Neuro-Fuzzy calibration approach.

1.2. Function Points: a short description

FP analysis is a process used to calculate software functional size. Currently, the most pervasive version is regulated in the Counting Practices Manual – version 4.2, which was released by the International Function Point User Group (IFPUG) [3]. Counting FP requires the identification of five types of functional components: Internal Logical Files (ILF), External Interface Files (EIF), External Inputs (EI), External Outputs (EO) and External Inquiries (EQ). Each functional component is classified as a certain complexity based on its associated file numbers such as Data Element Types (DET), File Types Referenced (FTR) and Record Element Types (RET). The complexity matrix for the five components is shown in Table 1. Table 2 illustrates how each function component is then assigned a weight according to its complexity. The Unadjusted Function Point (UFP) is calculated with Eq. 1, where *Wij* are the complexity weights and *Zij* are the counts for each function component.

$$\mathrm{UFP} = \sum_{i=1}^{5}\sum_{j=1}^{3} Zij \cdot Wij \qquad (1)$$

Once calculated, UFP is multiplied by a Value Adjustment Factor (VAF), which takes into account the supposed contribution of technical and quality requirements. The VAF is calculated from 14 General System Characteristics (GSC), using Eq. 2. The GSC includes the characteristics used to evaluate the overall complexity of the software.

$$\mathrm{VAF} = 0.65 + 0.01\sum_{i=1}^{14} Ci \qquad (2)$$

Ci is the Degree of Influence (DI) rating of each GSC.

Finally, a FP is calculated by the multiplication of UFP and VAF, as expressed in Eq. 3.

$$\mathrm{FP} = \mathrm{UFP} \times \mathrm{VAF} \qquad (3)$$

2. Research motivation and problem description

The significant relationship between the software size and cost has been recognized for a long time. In the classical view of cost estimation process, the outputs of *effort* and *duration* are estimated from software *size* as the primary input and a number of *cost factors* as the secondary inputs. There are mainly two types of software size metrics such as Source Lines of Code (SLOC) and FP. SLOC is a natural artifact that measures software physical size, but it is usually not available until the coding phase and difficult to have the same definition across different programming languages. FP has gained popularity over time because it can be used at an earlier stage of software development. Calibrating FP incorporates the historical information and gives a more accurate view of software size. Hence more accurate cost estimation comes with a better software size metric. The FPA method complexity weight system refers to all the complexity calculations and weight values expressed in FP. In other word, we have 15 parameters in the complexity weight system to tune up, which are low, average and high value of External Inputs, External Outputs, External Inquiries, Internal Logical Files, and External Interface Files, respectively. We summarized the objectives of this work in the form of following research questions:

RQ-1: "Does calibration of the function point weight values further enhanced improvements in the software size estimation process?"

RQ-2: "Does a soft computing based approach to calibrate the function point weight values provides improvement in the software size estimation process?"

Regression analysis is the traditional mathematical approach. Our aim is to calibrate the 15 parameter values of FPA method complexity weight system. However, with as many as 15 parameters at hand to calibrate, it is hard to obtain a good equation for all 15 parameters using statistical analysis approach such as regression analysis. Neural network is a relatively new and appealing methodology whose learning ability may lead to good result. However, it has an infamous problem of falling into the black-box trap. It is hard to explain how neural network is able to perform so well, particularly for software engineers. Fuzzy logic is also a hot topic since it can capture human's judgment. Instead of giving an exact number to all 15 Function Points parameters, we could define fuzzy linguistic terms and assign a fuzzy set within numeric range. This can overcome some of the problems exposed in the paper in the next subsection. However, only adopting fuzzy logic cannot benefit from historical project database. Our Neuro-Fuzzy approach presented in this paper is a novel combination of the above three approaches. It obtains a simple equation

Table 2
Function component complexity weight assignment

Component	Low	Average	High
External inputs	3	4	6
External outputs	4	5	7
External inquiries	3	4	6
Internal logical files	7	10	15
External interface files	5	7	10

Table 1
Complexity matrix for fp function components

ILF/EIF	DET			EI	DET			EO/EQ	DET		
RET	1–19	20–50	51+	FTR	1–4	5–15	16+	FTR	1–5	6–19	20+
1	Low	Low	Average	0–1	Low	Low	Average	0–1	Low	Low	Average
2–5	Low	Average	High	2	Low	Average	High	2–3	Low	Average	High
6+	Average	High	High	3+	Average	High	High	4+	Average	High	High

from statistical analysis, defines a suite of fuzzy sets to represent human judgment, 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.

2.1. Problem description and analysis

2.1.1. Ambiguous classification and crisp boundary

In FP counting method, each component, such as Internal Logical File (ILF) and External Interface File (EIF), is classified to a complexity level determined by the numbers of its associated files, such as Data Element Types (DET), Record Element Types (RET) [3]. Table 3 lists the complexity matrix for ILF as an example. Such complexity 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, Table 4 shows a software project with three ILF, A, B and C. According to the complexity matrix, A and B are classified as having the same complexity and are assigned the same weight value of 10. However, A has 30 more DET than B and is certainly more complex. They are now assigned the same complexity, which is recorded as Observation 1: ambiguous classification. Also, B is classified as average and assigned a weight of 10 while C is classified as low and assigned a weight of 7. B has only one more DET than C and the same number of RET as C. However, B has been assigned three more weight units than C. This is recorded as Observation 2: crisp boundary, because there is no smooth transition boundary between two classifications. Processing the number of FP component associated files such as DET, RET using fuzzy logic can produce an exact complexity degree.

Table 3
ILF complexity matrix

ILF	DET		
RET	1–19	20–50	51+
1	Low	Low	Average
2–5	Low	Average	High
6+	Average	High	High

Table 4
Observations on FP complexity classification

	ILF A	ILF B	ILF C
DET	50	20	19
RET	3	3	3
Complexity classification	Average	Average	Low
Weight value	10	10	7
Observation 1	Ambiguous classification		
Observation 2		Crisp boundary	

2.1.2. Calibration to reflect industry trends

The weight values of Unadjusted Function Point (UFP) in Table 2 are said to reflect the functional size of software [24], Albrecht determined them in 1979 based on the study of 22 IBM Data Processing projects. Since 1979, 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 maintains an empirical project data repository. ISBSG data repository release 8 contains 2027 projects, which come from dozens of countries and cover a broad range project types from many industries and business areas, with 75% of the projects being less than 5 years old. Learning UFP weight values from ISBSG data repository using neural network for calibration to reflect the current software industry trend is detailed in the next section.

2.1.3. Calibration to improve cost estimation

The neuro-fuzzy FP model presented in this paper is a unique approach that incorporates FP measurements with the neural networks, fuzzy logic and statistical regression techniques. A technical view of this model is depicted in Fig. 1. The first component, statistical regression is a mathematical technique used to represent the relationship between selected values and observed values from the statistical data. Secondly, 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. Once the concept of fuzzy logic is incorporated into the neural network, the result is a

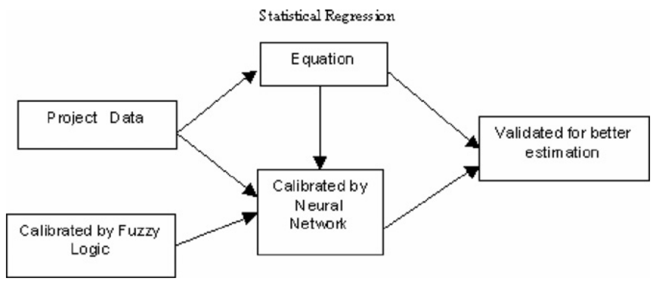

Statistical Regression
Equation
Project Data
Validated for better estimation
Calibrated by Neural Network
Calibrated by Fuzzy Logic

Fig. 1. Neuro-fuzzy Function Points Calibration Model block diagram.

neuro-fuzzy system that combines the advantages of both techniques.

3. Neuro-Fuzzy Function Point Calibration Model

3.1. Model overview

The block diagram shown in Fig. 1 gives an overview of the Neuro-Fuzzy Function Point Calibration Model. The project data provided by ISBSG [25] is imported to extract an estimation equation and to train the neural network. An estimation equation in the form of $\boldsymbol{Effort = A \cdot UFP^B}$, where A, B are all coefficients calculated from the regression process and extracted from the data set by using statistical regression technique. Fuzzy logic is used to calibrate FP complexity degree to fit specific application. Neural network calibrates UFP weight values to reflect the current software industry trend by learning from ISBSG data. The validation results show that the calibrated Function Points have better estimation ability than that of the original.

3.2. Fuzzy logic calibration step

The five FPA elements (ILF, EIF, EI, EO, EQ) are classified according to the complexity matrices. The inputs in the original complexity weight matrices are the numbers of the files associated with each function component, and the output is the component's complexity classification. We define three new linguistic terms: *small*, *medium* and *large*, to express the inputs qualitatively. For example, if an ILF has one Record Element Types (RET), we assume that ILF's RET input is small. Also, we use linguistic terms: *low*, *average* and *high* for the output, which are the same as in the original matrices. To fuzzify the inputs and outputs, we define fuzzy sets to represent the linguistic terms [26]. The fuzzy membership grade is captured through the membership functions of each fuzzy set. The inputs are of the trapezoidal type and the outputs are of the triangular type, because these types of membership functions are appropriate to use in preserving the values in the complexity weight matrices. Fig. 2 shows an example of fuzzy sets for ILF component. Table 5 illustrates the nine fuzzy rules defined based on the original complexity matrices. Each rule has two parts in its antecedent linked with an "AND" operator and one part in its consequence.

Next, the fuzzy inference process using the Mamdani approach [26] is applied to evaluate each component's complexity degree when the linguistic terms, the fuzzy sets, and the fuzzy rules are defined. Once the input, such as small DET or large RET is fuzzified, the degree to which each part of the antecedent for each rule can be calculated. We apply the *min* (minimum) method to evaluate the "AND" operation and obtain one number that represents the result of the antecedent for that rule. 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.

Table 5
Truth table of fuzzy logic rule set

Rule #	Input 1	Input 2	Output
1	Small	Small	Low
2	Small	Medium	Low
3	Small	Large	Average
4	Medium	Small	Low
5	Medium	Medium	Average
6	Medium	Large	High
7	Large	Small	Average
8	Large	Medium	High
9	Large	Large	High

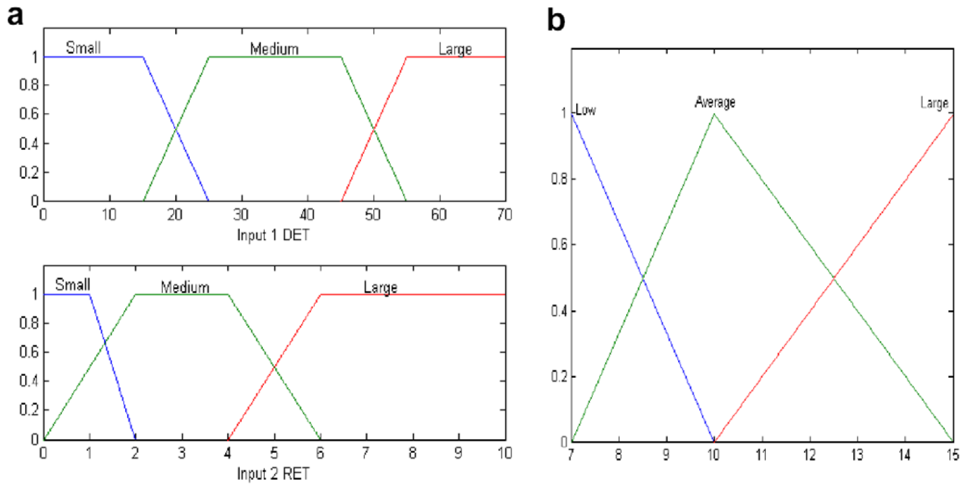


Fig. 2. Neuro-fuzzy Function Points Calibration Model fuzzy sets for ILF. (a) Inputs fuzzy sets (Trapezoidal) (b) Output fuzzy sets (Triangular).

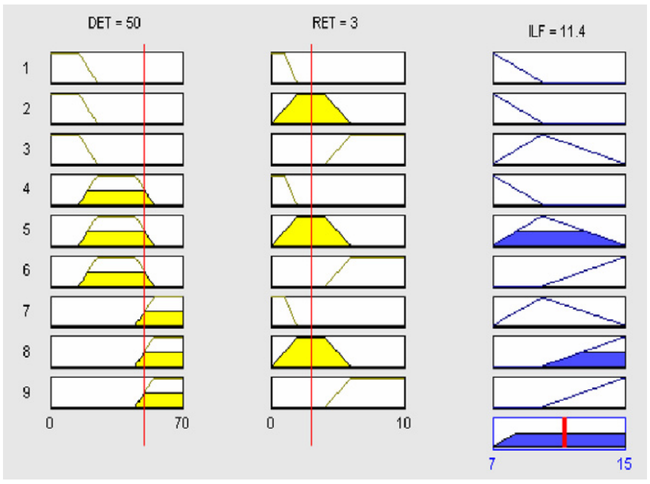
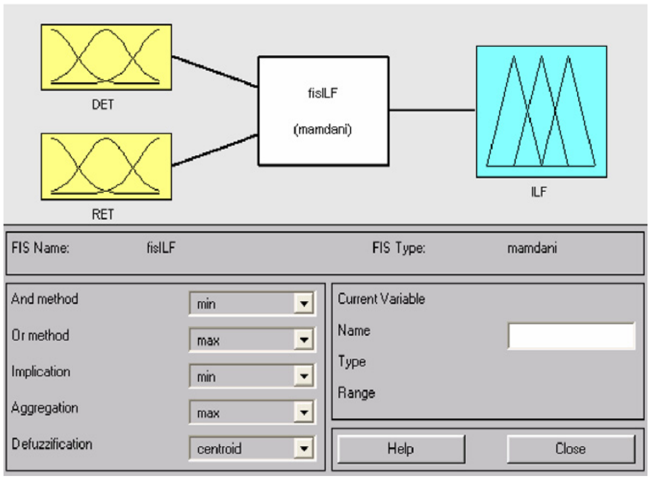


Fig. 3. Fuzzy inference process of neuro-fuzzy Function Points Model.

An example of the complete fuzzy inference process is shown in Fig. 3. Input values are set to DET: 50 and RET: 3. 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 4, 5, 6, 7, 8, 9 are activated for the antecedent part one and the rules 2, 5, 8 are activated for the antecedent part two. The implication is processed using the *min* method, and generates the consequence part of each rule as shown by the dark gray shadings in the right part of the Fig. 3. The aggregation process combines all the consequent fuzzy sets using the *max* method and produces one output fuzzy set, shown by solid dark gray shade at the bottom right of the Fig. 3. Finally, the consequent fuzzy set is defuzzified, using the centroid calculation method, and the output is achieved as a single value of 11.4, the bold black line in the output fuzzy set located in the bottom right of the Fig. 3. Now we can construct a fuzzy logic system for each FPA element (ILF, EIF, EI, EO, EQ). Fig. 4 shows the two-input-one-output fuzzy logic system for ILF as an



FIS Name: fisILF
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name
Type
Range

Fig. 4. Fuzzy logic system for ILF.

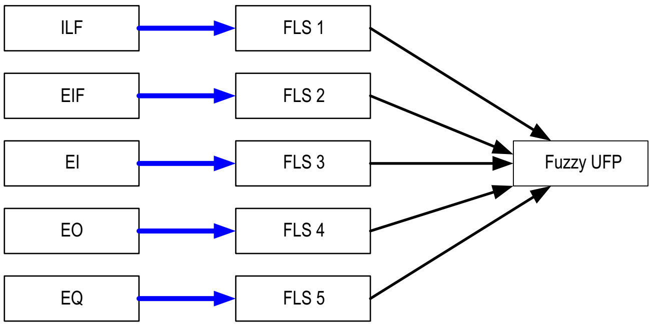


Fig. 5. Fuzzy complexity measurement system of neuro-fuzzy Function Points Model.

example. This approach is applicable to all five FPA elements. Afterwards, a fuzzy complexity measurement system that takes into account all five Unadjusted Function Points function components is built after the fuzzy logic system for each function component is established, as shown in Fig. 5. Each FPA element is into a Fuzzy Logic System (FLS). The outputs of all five FLS are summed up and become the fuzzy Unadjusted Function Points. The new fuzzy Unadjusted Function Points count is the result of the fuzzy complexity measurement system. Table 6 shows the original and the calibrated weight values of ILF using the same example in Section 3.2. We can find that the three different weight values are appropriately weighted.

3.3. Neural network calibration step

The neural network step is aiming at calibrating Function Points to reflect the current software industry trend. By learning from ISBSG data repository, a project data repository which meets the requirement of reflecting the industry trend, this part is achieved by the calibration goal.

3.3.1. Data preparation

In order to reach a reasonable conclusion, the raw ISBSG data set is filtered by several criteria. Recommended by the ISBSG [27], only the data points whose *quality* ratings is "A" or "B" is considered. Function Point has several variations of *counting methods* such as IFPUG [3], COSMIC FFP [28] and Mark II [29], of which the IFPUG method is the most popular (used by 90% of the projects). The work efforts are recorded at different *resource levels* and the projects recorded at the resource level one (development team), a level that covers 70% projects, are chosen.

Table 6
Calibration using fuzzy logic

	ILF A	ILF B	ILF C
DET	50	20	19
RET	3	3	3
Original weight value	10	10	7
Calibrated weight value	11.4	10.4	10.2

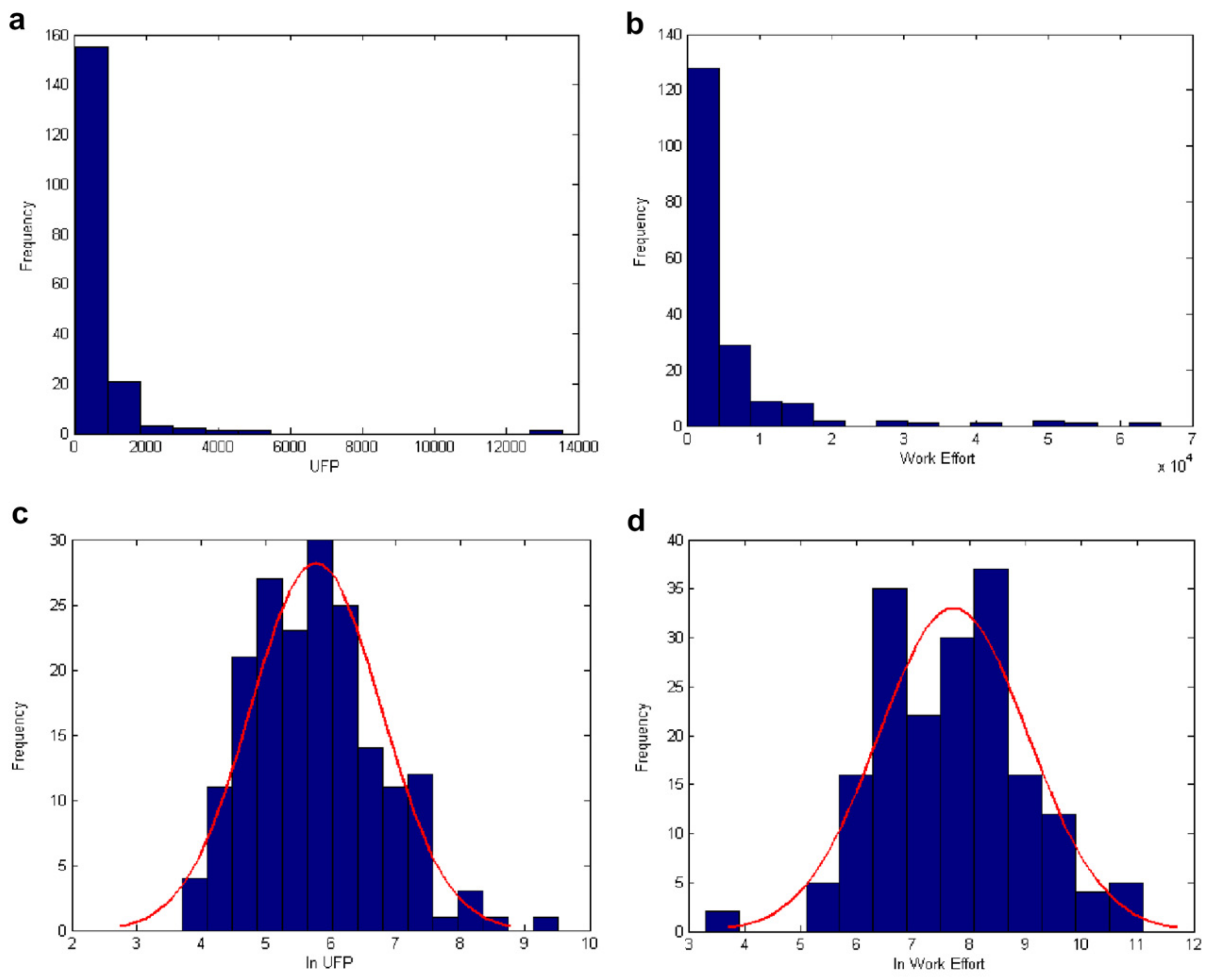


Fig. 6. (a) UFP; (b) work effort; (c) *ln* UFP; (d) *ln* work effort.

There are three major *development types* of the projects: the new development and redevelopment projects are calculated by the original FP equation, thus they are grouped into one type and are selected; whereas the enhancement projects are calculated by another different equation and are excluded.

The calibration requires the 15 *breakdowns of UFP*, that is, the associated file numbers of low, average and high for five function components. Also, 14 *GSC rating values* are chosen to add model's extendibility.[2] Application of all these criteria results in a significant decrease in the number of data points. A subset of 409 projects is obtained of which the quality rating is "A" or "B", the counting method is IFPUG, the effort resource level is one, and the development type is new development or re-development. Further selection of the projects that provide the 15 Unadjusted Function Point breakdowns and 14 GSC rating values results in a 184 projects data set. Angelis et al. [30] conducted research on ISBSG data repository and suffered from the same problem. They use ISBSG Release 6, which contains 789 projects but only 63 projects are left after applying filtering criteria.

An effort estimation equation is extracted based on the data subset using statistical regression analysis. The regression process includes logarithmic transformation, correlation analysis, statistical regression, and post-regression validity analysis. The statistical regression analysis assumes that the underlying data are normally distributed. However, the histograms of effort and size (Figs. 6(a) and (b)) show that they are not distributed normally but are highly skewed. To approximate a normal distribution, we apply a logarithmic transformation on these variables to make the large values smaller and to bring the data closer together. It is observed in the histograms of *ln* UFP and *ln* Work Effort (Figs. 6(c) and (d)) that the transformed variables are approximately normally distributed. The relationship between the work effort and size is visualized, using two-dimensional graphs as shown in Figs. 7(a) and 7(b), before and after logarithmic transformation, respectively.

An obvious positive linear relationship between effort and size after logarithmic transformation is observed and an equation in the form of Eq. 3 is achieved. Its equivalent form: Eq. 4 is used to estimate cost in work effort and

[2] Almost all the projects that provide 15 UFP breakdowns fields provide 14 GSC rating values.

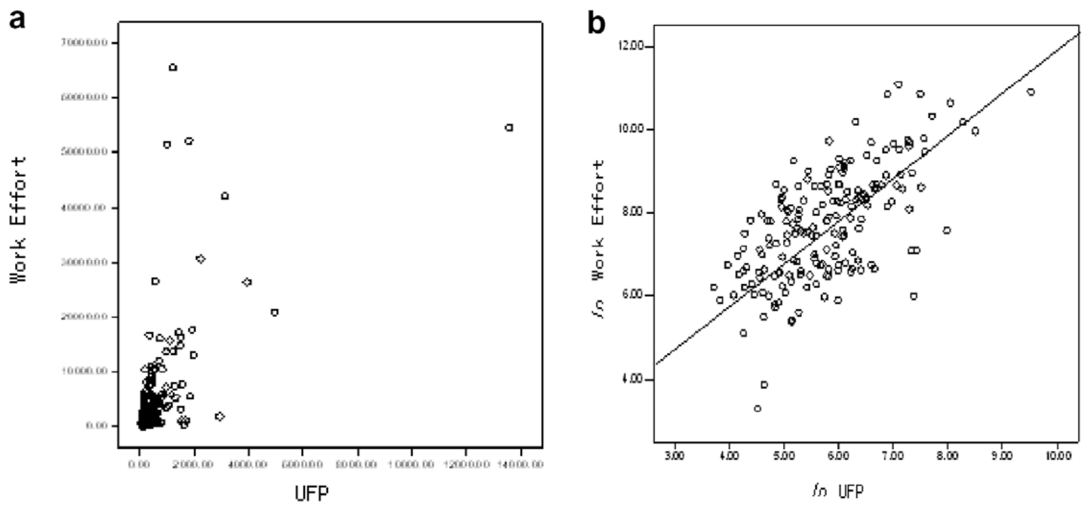


Fig. 7. (a) Work effort and UFP; (b) *ln* Work Effort and *ln* UFP.

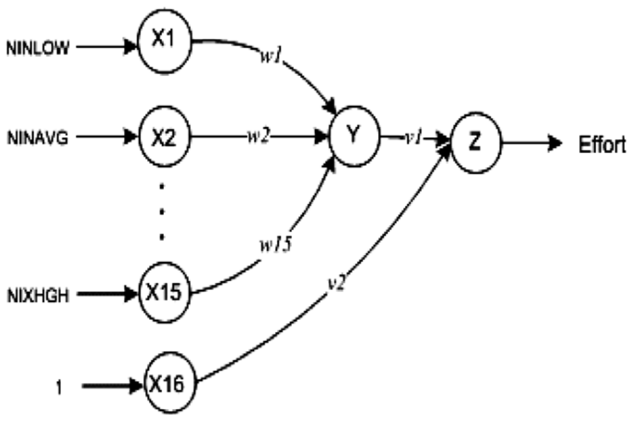


Fig. 8. Neural network structure.

serves to be an activation function of neural network. Eqs. 3 and 4 are shown below, where α, β, A, B are all coefficients calculated from the regression. Though of simple form, Eq. 4 is derived from the reliable filtered data set and analyzed by reliable statistical procedure. It does not include any special ISBSG repository parameters; thus it can estimate any Function Points oriented project and can be extended to include cost drivers for future works.

$$ln\ \text{Effort} = \alpha \cdot ln\ \text{UFP} + \beta \tag{4}$$

$$\text{Effort} = A \cdot \text{UFP}^{B} \tag{5}$$

3.3.2. Neural network structure

The neural network structure used in the Neuro-Fuzzy Function Points Calibration model is shown in Fig. 8. There are 16 input neurons, denoted as *Xi*, *i* from 1 to 16. Among these, neurons X_1 to X_{15} represent the three complexity ratings of five UFP components. The inputs of these 15 neurons are the numbers of their respective components denoted as NINLOW (number of low External Inputs), NINAVG (number of average External Inputs), etc. These 15 neurons are all connected to neuron Y associated with the weights of *wi*, *i* from 1 to 15. Neuron X_{16} is a bias node with a constant input of one and is connected to the output neuron Z with an associated weight of $v2$, which represents the coefficient A in Eq. 2. Neuron Y receives the outputs from the 15 neurons in the input layer and is then connected to neuron Z. The output of neuron Y is $Y = \sum_{i=1}^{15} X_i \cdot W_i$ which is functionally equivalent to the UFP calculation formulae ($\text{UFP} = \sum_{i=1}^{5}\sum_{j=1}^{3} Z_{ij} \cdot W_{ij}$). The activation function of neuron Z is $Z = v2 \cdot Y^{v1}$ which is of the same form as Eq. 4 ($Effort = A \cdot \text{UFP}^{B}$). Thus, neuron Z can be used to estimate the software cost in work effort from software size in UFP. The underlying reason of using work effort as the output to train the UFP weight values is that they are supposed to reflect the software component complexity and the complexity should be proportional to the project work effort which is based on common sense that the more complex the software, the more effort should be put in.

3.3.3. Learning procedure

The goal of the learning procedure of the neural network is to minimize the prediction difference between the estimated and actual efforts. Given *NN* projects, the prediction difference can be expressed as the error signal defined in Eq. 5:

$$E = \sum_{n=1}^{NN} \frac{1}{2} Wn \left[\frac{Zn - Zdn}{Zdn}\right]^2 \tag{6}$$

E is the error signal; Zn is the estimated effort of the nth project; Zdn is the actual effort of the nth project, the desired output; and Wn is the training weight given to the nth project whose default value equals to one.

Learning procedure flowchart (Fig. 9) illustrates the complete neural network learning procedure. Each neuron's associated weight is initialized to a certain value. The associated weights of all neurons *Xi* (*i* from 1 to 15)

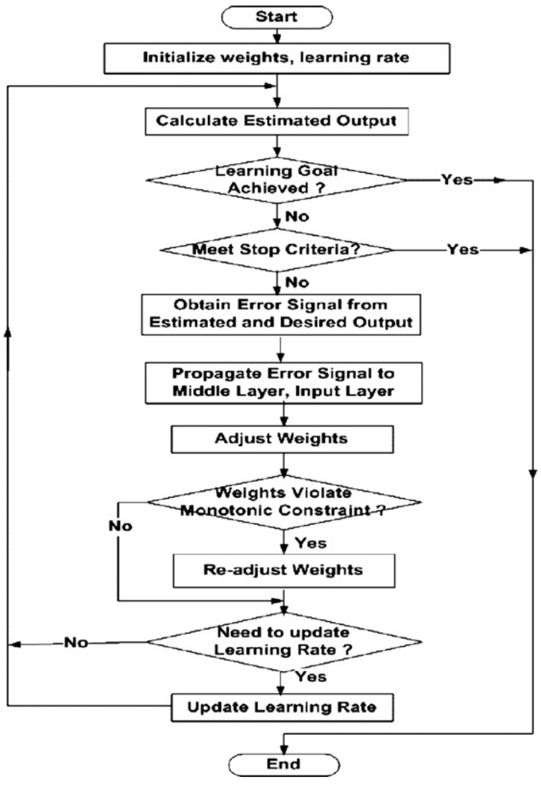

Start
Initialize weights, learning rate
Calculate Estimated Output
Learning Goal Achieved ?
Yes
No
Meet Stop Criteria?
Yes
No
Obtain Error Signal from Estimated and Desired Output
Propagate Error Signal to Middle Layer, Input Layer
Adjust Weights
Weights Violate Monotonic Constraint ?
No
Yes
Re-adjust Weights
Need to update Learning Rate ?
No
Yes
Update Learning Rate
End

Fig. 9. Learning procedure flowchart.

are initialized to the original Function Points weight values as suggested by Albrecht [24], and the weights $v1$ and $v2$ are initialized to the coefficient values obtained in the estimation equation extracted. Certain steps are then repeated until the learning process goal is achieved or the stop criteria are met. The learning process goal is to reduce the error signal E until it has fallen within a desired range. The stop criteria may be that the learning loop is found to diverge or the error signal is found to decrease insignificantly as the iteration continues. The corrective adjustments are applied on the weights to bring the output neuron Z closer to the desired response after iteration. Thus, the error signal E is minimized in a step-by-step manner.

The back-propagation mechanism [21] is adopted in the learning procedure. In the iterations of the learning procedure, the estimated output and the error signal is calculated from the input layer toward the output layer through the feed-forward path. Then the error signal propagates from the output layer to the input layer through the backward path. Adjustment of each neuron's associated weight is obtained when the error signal is back propagated. The learning algorithm is also subject to monotonic constraints in order to be consistent with the definition of Function Points. Monotonic constraints are identified in the definition of the weight values of Unadjusted Function Points. That is, the low weight value must be less than that of the average, and the average weight value must be less than that of the high, i.e., $Low < Average < High$. The learning rate is updated from time to time during the learning process when the error signal is not on the right trend. The weights $w1$ to $w15$ associated with neurons X1 to X15 are assigned new values when the learning procedure is completed. These new weight values are the calibrated weight values for the Unadjusted Function Points function components.

4. Experimental methodology and evaluation

A neuro-fuzzy FP calibration tool, named NFFPCT, has been implemented in Matlab 6.5. The training data set is loaded from a text file and the associated weights of the neurons are initialized to the original FP weight values. The outliers whose Relative Errors (RE) are larger than the threshold specified by the user are identified and excluded from the training data set. Next, the user inputs the neural network training parameters, such as the learning rate and the epoch number, and then clicks the "Start Training" button to launch the training process. Once the training finishes, the calibrated weight values are shown on the right side of the screen. The user can press the "Plot Results" button to depict a training trend graph at the bottom right of the screen. The goal of neural network training is achieved gradually as plotted in the bottom right of the screenshot: MMRE decreases as the training iteration epoch proceeds (MMRE is the Mean Magnitude Relative Error, a performance criterion that measures the estimation accuracy explained below). All of the training results can be saved in text files or in a Microsoft Excel spreadsheet.

In order to evaluate our neuro-fuzzy model, we conducted five experiments. For each experiment, we randomly separated the original data set of 184 projects into 100 training data points and 84 test data points. We selected the number of training data points (100) and test data points (84) in order to balance the numbers relatively evenly between the training and testing points. On the one hand, the number of training data points should be as high as possible to provide accurate learning results. On the other hand, these data points should not be too large so that the neural network functions effectively for the training data but performs poorly for the testing data. The outliers are the abnormal project data points with large noise that may distort the training result; an outlier is defined as a project whose Relative Error is larger than a threshold of 400%, where the threshold excludes less than 10% of data points as outliers.[3] Thus, we exclude the outliers in order to calibrate UFP weight values, but we include them in testing the model.

4.1. Criteria for performance evaluation

Several performance evaluation criteria used to assess the neuro-fuzzy FP model are described below.

[3] The numbers of outliers for Experiments 1–5 are 8, 8, 7, 7, 7 out of 100 training data set.

1. *Relative Error (RE).* For project *i*, Relative Error (RE) measures the estimation deviation and is defined as follows:

$$\mathrm{RE}_i = \frac{\mathrm{Estimated}_i - \mathrm{Actual}_i}{\mathrm{Actual}_i} \tag{7}$$

2. *Magnitude of Relative Error (MRE).* For project *i*, the Magnitude of Relative Error (MRE) measures the absolute estimation accuracy and is defined as follows:

$$\mathrm{MRE}_i = \frac{|\mathrm{Estimated}_i - \mathrm{Actual}_i|}{\mathrm{Actual}_i} \tag{8}$$

3. *Mean Magnitude of Relative Error (MMRE).* For *n* projects, the Mean Magnitude of Relative Error (MMRE) is expressed as follows:

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\mathrm{Estimated}_i - \mathrm{Actual}_i|}{\mathrm{Actual}_i} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MRE}_i \tag{9}$$

MRE_i is the Magnitude of Relative Error (MRE) for project *i*.

4. *Prediction at level p (PRED).* For *n* projects, the prediction at level *p* is defined as follows:

$$\mathrm{PRED}(p) = \frac{k}{n} \tag{10}$$

k is the number of projects wherein the MRE is less than or equal to *p*.

4.2. Discussion of results

The calibrated UFP weight values obtained from Experiments 1–5 are listed in Table 7. The original weight values are listed in the first column, and the average weight values from the five experiments are shown in the last column of the table. In most cases, the calibrated values are smaller than the original ones. The weight values are initialized to the original FP values assigned in 1979 and are then tuned by learning from the ISBSG Data Repository – release 8, which is an updated project data repository. This result accords with the fact that the overall productivity of the software industry has been continuously increasing since FP was invented in 1979. As stated in a report from QSM, for example, the software development productivity trend for a business application company is found to increase one unit on the Productivity Index (PI) every 1.4 years from 1990 to 2000, and the trend is expected to continue [31]. Numerous factors result in productivity improvement, such as improved developer capability, process model adoption like CMM, availability of CASE tools, and the advancement of computer technology. Therefore, lower weight values of FP result because productivity has increased in the last 25 years. In order to accomplish the same number of tasks, less effort is needed today than was the case in 1979, when FP was introduced.

The evaluation results of the five experiments, assessed by the Mean Magnitude Relative Error (MMRE), are listed in Table 8. "Improvement" refers to the MMRE difference between the original and the calibrated weight values. Fig. 10 plots the MMRE with the original weight values, the MMRE with the calibrated weight values, improvement over all five experiments and the average. The neuro-fuzzy calibration model has resulted in an average of 22% improvement of MMRE in estimating effort from calibrated Function Points compared with the original Function Points. After calibration, the MMRE is around 100%, which is still a relatively large percentage, due to the absence of well-defined cost drivers such as COCOMO [1] factors. Unfortunately, the ISBSG – release 8 does not have data on similar types of cost drivers.

The five experiments were assessed by PRED criteria, the evaluation results of which are listed in Table 9. Four PRED criteria are used here, specifically Pred 25, Pred 50, Pred 75 and Pred 100. Fig. 11 plots the comparison of the average original and the calibrated PRED results where overall improvement is observed.

Table 7
Calibrated UFP weights values

Component		Original	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	New average
EI	Low	3	0.65	1.63	0.40	1.06	0.94	0.94
	Average	4	2.11	2.74	1.89	1.97	2.06	2.15
	High	6	4.88	4.51	4.83	4.66	4.60	4.70
EO	Low	4	2.95	3.68	3.00	3.49	3.36	3.30
	Average	5	4.75	4.61	4.54	4.36	4.56	4.56
	High	7	5.97	6.39	6.15	6.51	5.87	6.18
EQ	Low	3	1.63	2.13	1.46	2.08	1.54	1.77
	Average	4	3.06	2.94	2.69	2.89	3.12	2.94
	High	6	5.48	5.21	5.42	5.18	5.45	5.35
ILF	Low	7	5.23	5.72	5.24	5.46	5.36	5.40
	Average	10	9.87	9.68	9.72	9.77	9.85	9.78
	High	15	14.94	14.90	14.88	14.92	14.94	14.92
EIF	Low	5	4.77	4.48	4.58	4.65	4.60	4.62
	Average	7	6.92	6.94	6.92	6.88	6.94	6.92
	High	10	10.00	9.99	10.00	10.00	10.00	10.00

Table 8
MMRE evaluation results

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
MMRE original	1.38	1.58	1.57	1.39	1.42
MMRE calibrated	1.10	1.28	1.17	1.03	1.11
Improvement	0.28	0.30	0.40	0.36	0.31
Average improvement			0.33		
Percentage improvement	20%	19%	25%	26%	22%
Average percentage improvement			22%		

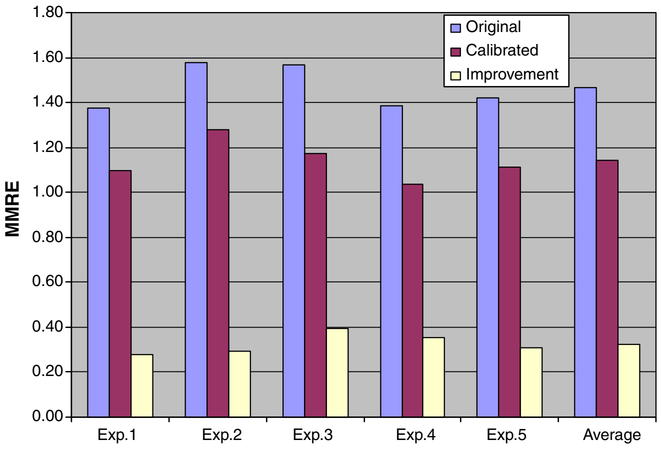


Fig. 10. MMRE evaluation results comparison.

The evaluation results from 15 sample projects are listed in Table 10. There are three projects for each experiment. First of all, there is one mini project, where the work effort is less than 3000 staff-hours or 20 staff-months). Also, there is one small project, where the work effort involves approximately 4500–7500 staff-hours or 30–50 staff-months. Finally, there is one medium to large project; a medium project is defined as one where the work effort is around 15,000 to 22,500 staff-hours or 100–150 staff-months, and a large project where the work effort is more than 30,000 staff-hours or 200 staff-months.[4] In Table 9, other categories include "Actual", which is the actual work efforts measured by staff-hours; "UFP", which is the unadjusted FP; "WE", which is estimated work efforts; "RE", which is the Relative Error; and "Percentage Improvement", which is the percentage of improvement in the Magnitude Relative Error (MRE). The whole data set is unbalanced in terms of the project size distribution: only around 10% of the projects are medium or large projects and the remaining 90% are mini or small; the unbalanced training data leads to uneven results. Although the evaluation of the neuro-fuzzy FP model provides better estimation results for small or mini projects than it does for medium and large ones, the overall estimation is improved. Moreover, ISBSG does not provide data for well-defined cost drivers. Since cost drivers play an important role in estimation, especially for medium and large projects some of the results regarding the medium and large projects are not impressive.

[4] Each staff-month translates to roughly 150 staff-hours.

5. Validity analysis of model

The two most important aspects of precision in experiment-based studies are reliability and validity. Reliability refers to the reproducibility of a measurement, whereas validity refers to the correspondence between the experimental value of a measurement and its true value. Subsequently, this section discusses the reliability and validity of the results observed during and after the calibration of weights for the five FP elements. After applying the average calibrated weights, the reliability of the five FP elements was evaluated by internal-consistency analysis, which was performed using the coefficient alpha. This coefficient of EI, EO, EQ, ILF and EIF was 0.59, 0.53, 0.66, 0.38 and 0.66, respectively; with the original weights, the respective values for coefficient alpha of EI, EO, EQ, ILF and EIF were 0.53, 0.50, 0.63, 0.35 and 0.63. In comparing these two sets of data, it is evident that using the calibrated weights produces a slight improvement in the reliability coefficient.

According to Hunter and Schmidt [32], construct validity is a quantitative question rather than a qualitative distinction between "valid" or "invalid"; it is a matter of degree. The construct validity of the five FP elements was evaluated using principal component analysis with VARIMAX rotation. In this study, we used eigen-value and scree plot as reference points to observe the construct validity. More specifically, we utilized eigen-value-one-criterion, also known as the Kaiser criterion, which means that any component having an eigen-value greater than one was retained. Eigen-value analysis revealed that the FP elements formed a single factor with an eigen-value of 1.87, whereas a second factor was also formed with a value that was slightly higher than the threshold of 1.0 and with an EIF factor loading range of 0.98. The scree plot clearly showed a division at the first component. EI and ILF showed the strongest factor loading range of 0.95 and 0.92, respectively, hence supporting the conclusion of prior research [33,34,35], which maintains that EI and ILF have a strong correlation and that EIF is rarely correlated with other FP elements.

The second type of validity, criterion validity, is concerned with the degree to which the scales under study

Table 9
PRED evaluation results

		Experiment1	Experiment 2	Experiment 3	Experiment 4	Experiment 5	Average
Pred 25	Original	13%	13%	14%	13%	11%	13%
	Calibrated	14%	11%	14%	10%	13%	12%
	Improvement	1%	−2%	0%	−3%	2%	0%
Pred 50	Original	29%	21%	29%	20%	18%	23%
	Calibrated	30%	25%	30%	26%	24%	27%
	Improvement	1%	4%	1%	6%	6%	4%
Pred 75	Original	45%	36%	43%	37%	37%	40%
	Calibrated	54%	37%	45%	50%	42%	46%
	Improvement	9%	1%	2%	13%	5%	6%
Pred 100	Original	66%	57%	66%	57%	51%	60%
	Calibrated	77%	62%	70%	67%	60%	67%
	Improvement	11%	5%	4%	10%	9%	8%

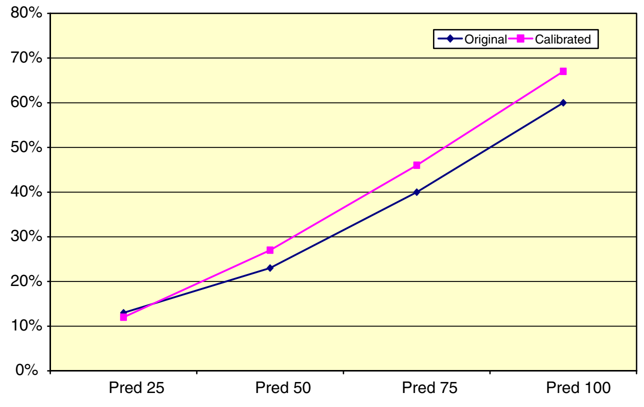


Fig. 11. PRED evaluation results comparison.

are related to an independent measure of the relevant criterion [36]. We used multiple regression analysis to determine the criterion validity of the five FP elements and the UFP; the FP elements were used as predictor variables and the UFP was used as a criterion variable. Subsequently, the multiple correlation coefficients measured was 0.54 and the F-ratio of 70.92 was at $P < 0.001$. Cohen [37] reported that a multiple correlation coefficient higher than 0.51 corresponds to a large effect size. Therefore, we can conclude that the criterion validity of the five FP elements is sufficient.

Table 11 illustrates the regression analysis of the samples utilized in all five experiments and can be used to explain the shape of the data and its variability. The adjusted multiple R^2 measures the proportion of the variability in the dependent variable (work effort), which has been explained by the relationship with the independent variable (function points). The F-ratio statistics were used to estimate the probability that the independent variable does not improve the prediction of the dependent variable; the accompanying p-value indicates this probability. Cohen [37] suggested that a multiple correlation coefficient of 0.14 corresponds to a small effect size, coefficients of 0.36 correspond to a medium effect size, and coefficients above 0.51 correspond to a large effect size. Table 11 shows the "post hoc" power analysis of the five experiments' sample sub-datasets based on regression analysis. In power analysis, we can specify any three of four quantities: alpha, power, sample size and effect size, and then estimate the missing one. In "post hoc" power analysis, we specified everything

Table 10
Evaluation results from sample projects

Experiment	Project type	Actual	Original estimates			Calibrated estimates			Percentage improvement
			UFP	WE	RE	UFP	WE	RE	
1	Mini	1155	519	3297.4	1.85	371.3	2479.8	1.15	38%
	Small	5525	1828	9625.7	0.74	1331.5	7350.5	0.33	55%
	Medium/large	65513	1200	6728.2	−0.90	924.9	5391.1	−0.92	−2%
2	Mini	1691	432	2719.4	0.61	334.7	2133.7	0.26	57%
	Small	7505	1255	9348.6	0.25	1010.1	7757.8	0.03	88%
	Medium/large	54620	13580	72326.8	0.32	10545.9	58204.5	0.07	78%
3	Mini	2041	582	3714.6	0.82	441.3	2885.3	0.41	50%
	Small	5800	1164	6994.6	0.21	819.9	5079.2	−0.12	43%
	Medium/large	21014	4943	26192.5	0.25	3350.3	18363.5	−0.13	48%
4	Mini	1367	383	2577.8	0.89	275.5	1852.9	0.36	60%
	Small	5320	1300	8775.5	0.65	1083	7307.5	0.37	43%
	Medium/large	51527	981	6617.6	−0.87	702.6	4735.7	−0.91	−5%
5	Mini	1367	383	2105	0.54	266.6	1545.6	0.13	76%
	Small	5864	796	7294.2	0.24	508.3	4799.9	−0.18	25%
	Medium/large	13083	1956	16875.1	0.29	1559.4	13659.1	0.04	86%

Table 11
Regression and power analysis of experiment's sample

Experiment #	N	Regression analysis[a]		Power analysis[b]			
		Adjusted R^2	F-ratio	Effect size	Lambda	Critical F	Power
1	84	0.41	58.73	0.694	58.37	3.10	1.000
2	84	0.43	64.92	0.754	63.36	3.10	1.000
3	84	0.42	62.33	0.724	60.82	3.10	1.000
4	84	0.41	59.54	0.694	58.37	3.10	1.000
5	84	0.36	48.36	0.562	47.25	3.10	1.000

[a] Regression analysis: p-value < 0.0001.
[b] Power analysis: $\alpha = 0.05$.

but expected power and subsequently estimated this missing variable. First, we calculated the effect size by using the adjusted R^2, and then we used the quantities of alpha as 0.05, the sample size as 84, and the respective effect size of experiments in order to calculate the power of the test. The effect size ranges from 0.56 to 0.75, which is considered to be a large effect size.

Another aspect of validity is concerned with whether or not the study reports results that correspond to previous findings. Prior investigations [33,34,35] that study the correlation between the five elements of function points concluded that EI and ILF are always correlated, and that EIF is rarely correlated with another FP element; the general findings of these investigations agreed that although most of the correlations are weak, there are still some strong ones. We applied the average calibrated weights proposed by our model to the 184 projects, and then calculated the correlation coefficients among the five FP elements with the purpose of analyzing whether they are inline with previous findings. EI and ILF showed a strong correlation of 0.90 at $P < 0.001$, whereas EIF did not correlate with EI and EQ at $P < 0.05$. Thus, the analysis we obtained is in agreement with previous studies, therefore verifying its validity.

5.1. Threats to external validity

Threats to external validity are conditions that limit the researcher's ability to generalize the results of his/her experiment to industrial practice [38], which was the case with this study. Specific measures were taken to support external validity; for example, a random sampling technique was used to draw samples from the population in order to conduct experiments, and filtering was applied to the ISBSG data set. Five experiments were conducted by drawing five different random samples in order to generalize the results. "Post hoc" analysis of effect size and power reinforced the external validity of the experiments by yielding a large effect size. Furthermore, the correlation among the five FP elements was found to correspond with previous studies, thus, also supporting the external validity of the experiments in the context of previous studies.

The proposed calibration of the FP element's weights were applied to the ISBSG data set to monitor the effectiveness of the approach; a potential threat to the external validity of this study involved the question of whether or not similar results would be obtained with an entirely different sample. In this investigation, we calibrated the weights of the five FP elements using only the ISBSG data set, which has raised a threat to the external validity of the calibration process. The ISBSG data set contains projects using different function point counting techniques, such as IFPUG, COSMIC and MARK II; since 90% of the sample used the IFPUG counting method; we therefore restricted our experiments to IFPUG projects. This decision may lead to the question as to whether the proposed model's outcome will be valid if the model is used with a type of FP counting technique besides IFPUG.

Our study is a data-driven type of research where we extracted a model based on known facts. The proposed model is more meaningful for small projects, which are actually the most common type of projects in the software industry. This limitation is due to the ISBSG data sets' characteristics used in this study, and it has raised threats to external validity, specifically in the case of large projects. In reality, there are more small projects than large ones, and even the large projects tend to be subdivided into smaller projects so that they become easier to manage. Although the proposed approach has some potential to threaten external validity, we followed appropriate research procedures by conducting and reporting tests to guarantee the reliability and validity of the study, and certain measures were also taken to ensure the external validity.

6. Conclusion

The neuro-fuzzy FP calibration model presented in this paper aimed at finding answers to the research questions RQ-1 and RQ-2 of this study and improved the Function Points Analysis Method. The experimental results show a 22% accuracy improvement of MMRE in software effort estimation from Function Points and demonstrate that Function Points need calibration and can be calibrated. This finding provides an answer to research question RQ-1 of this investigation. The fuzzy logic part of the model calibrates the Function Points complexity degree to fit the specific application context. The neural network part of the model calibrates the UFP weight values to reflect the current software industry trend. This part of the model overcomes three problems with the unadjusted FP complexity weight values: obsolete weight values, weight values defined subjectively, and weight values defined locally. We demonstrated in this paper that the use of fuzzy logic and neural network techniques to calibrate FP further improves the cost estimation process of software projects. Thus, this observation provides answer to the research question RQ-2 of this study.

FP as a software size metric is an important topic in the software engineering domain. A practical suggestion for future endeavors in this area could be recalibration of FP. ISBSG – release 8 is a large, new and wide-range project data repository, so the calibrated FP weight values learned from this repository reflect the maturity level of today's software industry. However, software development is a rapidly growing industry and these calibrated weight values will not reflect tomorrow's software. In the future, when modern project data is available, the FP weight values will again need to be re-calibrated to reflect the latest software industry trend. The neuro-fuzzy FP model is a framework for calibration and the neuro-fuzzy FP calibration tool can automate the calibration process when data becomes available.

References

[1] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. Clark, B. Steece, A. Brown, S. Chulani, C. Abts, Software Cost Estimation with COCOMO II, Prentice Hall, Upper Saddle River, NJ, 2000.

[2] L.H. Putnam, W. Myers, Measures of Excellence, Prentice Hall, Upper Saddle River, NJ, 1992.

[3] Function Point Counting Practices Manual, fourth ed. (2004), International Function Point Users Group.

[4] S. Shapiro, Splitting the difference: the historical necessity of synthesis in software engineering, IEEE Annals of the History of Computing 19 (1) (1977) 20–54.

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

[6] G. Antoniol, C. Lokan, G. Caldiera, R. Fiutem, A function point-like measure for object-oriented software, Empirical Software Engineering 4 (3) (1999) 263–287.

[7] G. Antoniol, R. Fiutem, C. Lokan, Object-oriented function points: an empirical validation, Empirical Software Engineering 8 (3) (2003) 225–254.

[8] K. Srinivasan, D. Fisher, Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering 21 (2) (1995) 126–137.

[9] S.G. MacDonell, Software source code sizing using fuzzy logic modeling, Information and Software Technology 45 (2003) 389–404.

[10] W. Pedrycz, F. Gomide, An Introduction to Fuzzy Sets: Analysis and Design, MIT Press, Cambridge, MA, 1998.

[11] P. Musilek, W. Pedrycz, G. Succi, M. Reformat, Software cost estimation with fuzzy models, ACM SIGAPP Applied Computing Review 8 (2) (2000) 24–29.

[12] X. Huang, D. Ho, J. Ren, L.F. Capretz, Improving the COCOMO model with a neuro-fuzzy approach, Applied Soft Computing 7 (2007) 29–40.

[13] G.R. Finnie, G.E. Wittig, J.M. Desharnais, A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models, Journal of Systems Software 39 (1977) 281–289.

[14] C. Yau, H.-L. Tsoi, Modelling the probabilistic behaviour of function point analysis, Information and Software Technology 40 (1998) 59–68.

[15] O.S. Lima, P.F.M. Farias, A.D. Belchior, Fuzzy modeling for function points analysis, Software Quality Journal 11 (2003) 149–166.

[16] M.A. Al-Hajri, A.A.A. Ghani, M.S. Sulaiman, M.H. Selamat, Modification of standard function point complexity weights system, Journal of Systems and Software 74 (2005) 195–206.

[17] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, NJ, 1998.

[18] L.A. Zadeh, Fuzzy logic, Computer (1988) 21.

[19] T.J. Ross, Fuzzy Logic with Engineering Applications, second ed., John Wiley & Sons, Hoboken, NJ, 2004.

[20] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[21] J.S.R. Jang, C.T. Sun, E. Mizutani, Neuro-fuzzy and Soft Computing a Computational Approach to Learning and Machine Intelligence, Prentice Hall, Upper Saddle River, NJ, 1997.

[22] A. Abran, P. Robillard, Function Points analysis: an empirical study of its measurement processes, IEEE Transactions on Software Engineering 22 (12) (1996) 895–910.

[23] T. Kralj, I. Rozman, M. Hericko, A. Zivkovic, Improved standard FPA method – resolving problems with upper boundaries in the rating complexity process, Journal of Systems and Software 77 (2005) 81–90.

[24] A. Albrecht, Measuring application development productivity, in: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium (1979) 83–92.

[25] About ISBSG, International Software Benchmarking Standards Group, 2004.

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

[27] Guidance on the Use of the ISBSG Data, International Software Benchmarking Standards Group, 2004.

[28] COSMICFFP Measurement Manual, Common Software Measurement International Consortium, January 2003.

[29] C. Symons, Function point analysis: difficulties and improvement, IEEE Transactions on Software Engineering 14 (1) (1988) 2–11.

[30] L. Angelis, I. Stamelos, M. Morisio, Building software cost estimation model based on categorical data, in: Proceedings of the 7th International Symposium on Software Metrics (2001) 4–15.

[31] L.H. Putman, Linking the QSM productivity index with the SEI maturity level (2000). Available from: <http://www.qsm.com/provisions.html>.

[32] J.E. Hunter, F.L. Schmidt, Methods of Meta-analysis: Correcting Error and Bias in Research Findings, Sage Publications, Thousand Oaks, CA, 1990.

[33] B. Kitchenham, K. Kansala, Inter-item correlations among function points, in: Proceedings of the 15th IEEE International Conference on Software Engineering (1993) 477–480.

[34] D.R. Jefery, J. Stathis, Function point sizing: structure, validity and applicability, Empirical Software Engineering 1 (1) (1996) 11–30.

[35] C. Lokan, An empirical study of the correlations between function point elements, in: Proceedings of the 6th International Symposium on Software Metrics (1999) 200–206.

[36] T. Dyba, An empirical investigation of the key factors for success in software process improvement, IEEE Transactions on Software Engineering 31 (5) (2005) 410–424.

[37] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Laurence Erlbaum, London, UK, 1988.

[38] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering, Kluwer Academic Publishers, Norwell, MA, 2000.