Neuro-Fuzzy Approach to Calibrate Function Points

WEI XIA[1], LUIZ FERNANDO CAPRETZ[2], DANNY HO[3]
[1]HSBC Bank Canada, IT Department, Vancouver, BC, Canada
[2]University of Western Ontario, Dept. Electrical & Computer Engineering, London, ON, Canada

[3]NFA Estimation Inc., London, ON

Abstract: - Function Points is an important and well-accepted software size metric. However, it is absolutely essential to accurately calibrate Function Point (FP), whose aims are to fit specific software application, to reflect software industry trend, and to improve cost estimation. Neuro-Fuzzy is a technique that incorporates the learning ability from neural network and the ability to capture human knowledge from fuzzy logic. We developed a Neuro-Fuzzy model to calibrate Function Points. The empirical validation using ISBSG data repository Release 8 shows a 22% improvement in software effort estimation after calibration using Neuro-Fuzzy technique.

Key-Words: - Neuro-fuzzy, Neural networks, Fuzzy logic, Software cost estimation, Function Points

1 Introduction

Function Points is a metric of measuring software size that was proposed by Albrecht [1] at IBM in 1979. It was a big step forward compared with the use of counts of Source Lines of Code (SLOC) that focuses on program ”functionality”, or ”utility rather than counting LOC”. Function Point Analysis is a process to count Function Points, of which the most pervasive version is regulated in the Counting Practices Manual version 4.2, released by the International Function Point User Group (IFPUG) [2].

Counting Function Points requires identifying five types of function components, namely Internal Logical Files (ILF), External Interface File (EIF), External Inputs (EI), External Outputs (EO) and External Inquiries (EQ). Each function component is classified to a certain complexity based on its associated number such as Data Element Types (DET), File Types Referenced (FTR) and Record Element Types (RET). Each function component is then assigned a weight according to its complexity. (See in Table 1).

Table 1: FP Component Weight Values.

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

The Unadjusted Function Points (UFP) is calculated by Equation 1, where *Wij* are the complexity weights and *Zij* are the counts for each function component.

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

Then UFP is multiplied by a Value Adjustment Factor (VAF) which takes into account the supposed contribution of technical and quality requirements. Finally Function Points is reached by multiplication of Unadjusted Function Points (UFP) and Value Adjustment Factor (VAF), as expressed in the following equation.

$$FP = UFP \times VAF$$

Function Points is an ideal software size metric to estimate cost since it can be obtained in the early development phase, such as requirement, measures the software functional size from user’s view, and is programming language independent [3]. To achieve more accurate estimation, it is necessary to calibrate Function Points.

2. Function Points Calibration

The weight values of Unadjusted Function Points [1] are said to reflect the functional size of software. They have never been updated since being introduced in 1979, and are applied universally. By contrast, software development has been in a growth mode since 1979, and today's software differs drastically from what it was over two decades ago. Such an imbalance prompts the questions: Do these weight values still make sense for today's software? Are the complexity weight values obsolete?

The weight values of Unadjusted Function Points were determined by Albrecht by "debate and trial", based on his experience and knowledge. Albrecht contributed significantly to the theory of Function Points. Nevertheless, with no actual projects to justify them, the question remains as to whether the weight values were defined subjectively without convincing support data.

The weight values of Unadjusted Function Points were decided based on the study of the data processing systems at IBM. The assignment of weight values was restricted to only one organization and to only one type of software. However, this set of weight values is applied universally and is not limited to one organization or one type of software. Is it possible to trust weight values defined locally to reflect the software globally?

All these questions lead to the conclusion that Function Points need calibration. The aim of Function Points calibration is to fit specific software application, to reflect software industry trend, and to improve cost estimation.

3 Neuro-Fuzzy Approach

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 [5]. Once 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. It is rich in representing human linguistic ability with the terms such as fuzzy set, fuzzy rules [6], [7]. 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 [8], [9]. This technique is found appropriate to calibrate FP as proved by the validation results.

Yau and Tsoi [10] 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, Farias and Belchior [11] also proposed 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 again 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. [12] 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 International Software Benchmarking Standards Group (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.

4 Neuro-Fuzzy Approach to Calibrate Function Points

4.1 Calibration Approach Overview

We propose an approach to calibrate FP using Neuro-Fuzzy technique. The model overview and two parts of the model: fuzzy logic part and neural network part are described here. The empirical validation is provided in the next section. Fig 1 gives an overview of our approach.

The project data provided by ISBSG [13] is imported to extract an estimation equation and to train the neural network. The estimation equation is extracted from the data set by statistical regression analysis. Fuzzy logic is used to calibrate FP complexity weights 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 FP weights have better estimation ability than that of the original.

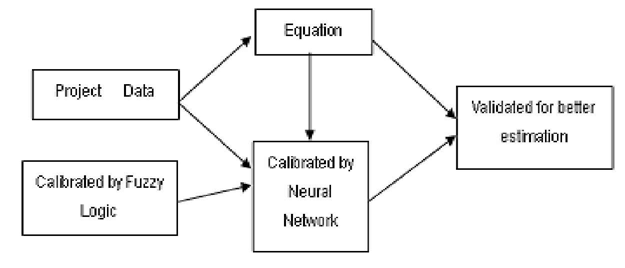
Equation
Project Data
Validated for better estimation
Calibrated by Fuzzy Logic
Calibrated by Neural Network

Fig. 1: Neuro-Fuzzy Calibration Approach Overview

4.2 Fuzzy Logic Part

The fuzzy logic part calibrates the FP complexity degree to fit the specific application. A fuzzy logic system (shown in Figure 3) is constructed based on the fuzzy set, fuzzy rules and fuzzy inference. The input fuzzy sets are to fuzzify the component associated file numbers and the output fuzzy set are to fuzzify the complexity classification. The fuzzy rules are defined in accordance with the original complexity weight matrices. The fuzzy inference process using the Mamdani approach [4] is applied based on the fuzzy sets and fuzzy rules.

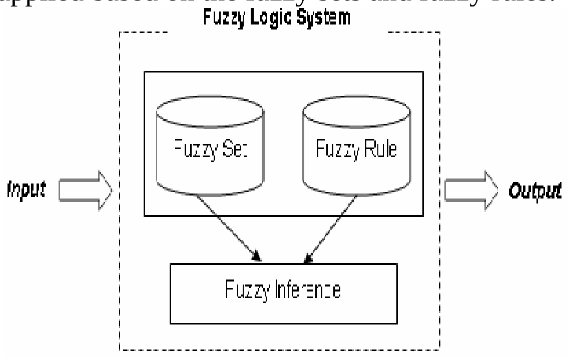

Fig 2: Fuzzy Logic System

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 Figure 3. The new fuzzy Unadjusted Function Points count is the result of the fuzzy complexity measurement system.

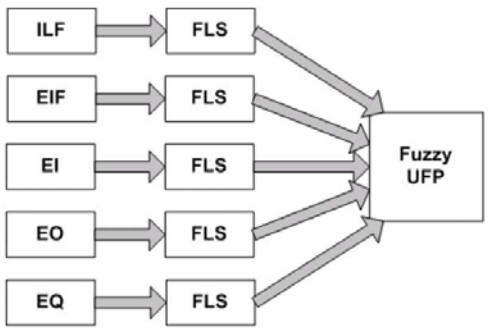

Fig. 3 Fuzzy Function Points Output

4.3 Neural Network Part

The neural network constructed to receive 15 UFP breakdowns as inputs to give the work effort as the desired output. A back-propagation learning algorithm [14] is conducted in order to minimize the prediction difference between the estimated and actual efforts. An effort estimation equation is extracted based on the data subset using statistical regression analysis. The equation in the form of:

$$Effort = A \cdot UFP^{B}$$

is achieved with the help of the statistical software SPSS v12 [15].

4.4 Empirical Validation Result

In order to reach a reasonable conclusion, the raw ISBSG data set is filtered by several criteria recommended by ISBSG [13]. A subset of 184 projects is obtained of which the quality rating is A or B, the counting method is IFPUG which excludes other counting methods such as COSMIC FFP [17] and Mark II [18], the effort resource is recorded at level one (development team), the development type is new development or re-development, the 15 Unadjusted Function Point (UFP) breakdowns and 14 General System Characteristics rating values are available.

Five experiments were conducted to validate our Neuro-Fuzzy approach. For each experiment, the original data set (184 projects) was randomly separated into 100 training data points and 84 test data points. The outliers are the abnormal project data points with large noise that may distort the training result. Thus, we used the training data set excluding the outliers for calibration, but used the rest of the data points for validation [16].

The validation results of the five experiments are assessed by Mean Magnitude Relative Error (MMRE) for estimation accuracy. MMRE is defined as:

for n projects,

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\left(| Estimated_i - Actual_i | / Actual_i\right)$$

The validation results of the five experiments are listed in Table 2 where “Improve %” is the MMRE improvement in percentage for each experiment. Based on the MMRE assessment results, an average of 22% cost estimation improvement has been achieved with the Neuro-Fuzzy calibration approach.

	Exp.1	**Exp.2**	**Exp.3**	**Exp.4**	**Exp.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
I mprove %	20%	19%	25%	26%	22%
Average Improve %	**22%**				

Table 2: MMRE Validation Result

5 Conclusion

The Neuro-Fuzzy approach to calibrate FP is validated with the empirical data repository (ISBSG Release 8). The experimental validation results show a 22% improvement in software cost estimation and demonstrate that FP need calibration and can be calibrated. The fuzzy logic part of the model calibrates the FP complexity weights 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. The combined neuro-fuzzy technique calibrates FP for better software cost estimation.

References:

[1] A. Albrecht, "Measuring Application Development Productivity," in *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, October 1979, pp. 83–92.

[2] Function Point Counting Practices Manual, 4th ed., International Function Point Users Group, January 2004.

[3] A. J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: a Software Science Validation," *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639–648, June 1983.

[4] E. H. Mamdani, "Application Of Fuzzy Logic To Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, vol. C26, no. 12, pp. 1182 – 1191, 1977.

[5] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1998.

[6] T. J. Ross, Fuzzy Logic with Engineering Applications, Wiley, 2004.

[7] L.A. Zadeh, "Fuzzy Logic", *Computer*, vol. 21, pp. 83–93, April 1988.

[8] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing A Computional Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.

[9] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 126–137,1995.

[10] C. Yau and H-L. Tsoi, "Modelling the Probabilistic Behaviour of Function Point Analysis", *Information and Software Technology*, vol. 40, pp. 59-68, 1998.

[11] O. S. Lima, P. F. M. Farias and A. D. Belchior, "Fuzzy Modeling for Function Points Analysis", *Software Quality Journal*, vol. 11, pp. 149-166, 2003.

[12] M. A. Al-Hajri, A. A. A. Ghani, M. S. Sulaiman and M. H. Selamat, "Modification of Standard Function Point Complexity Weights System", *Journal of Systems and Software*, vol. 74, pp. 195-206, 2005.

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

[14] S. Haykin, *Neural Networks*: A Comprehensive Foundation, Prentice Hall, 1998.

[15] SPSS, SPSS 12.0 Brief Guide, SPSS Inc., 2003.

[16] W. Xia, *Calibrating Software Size of Function Points Using Neuro-Fuzzy Technique*, Master Thesis, Department of Electrical and Computer Engineering, The University of Western Ontario, June 2005.

[17] *COSMICFFP Measurement Manual*, 2nd ed., Common Software Measurement International Consortium, January 2003.

[18] C. Symons, "Function Point Analysis: Difficulties and Improvement," *IEEE Transactions on Software Engineering*, vol. 14, no. 1, pp. 2–11, January 1988.