ENHANCED SOFTWARE DEVELOPMENT EFFORT AND COST ESTIMATION USING FUZZY LOGIC MODEL

***Moon Ting Su*[1], *Teck Chaw Ling*[2], *Keat Keong Phang*[3], *Chee Sun Liew*[4], *Peck Yen Man*[5]**
[1,2,3,4,5] Faculty of Computer Science and Information Technology,
University of Malaya, Malaysia.
Email: {smting, tchaw, kkphang, csliew}@um.edu.my,
wgc99007@siswazah.fsktm.um.edu.my

ABSTRACT

The development of software has always been characterized by parameters that possess certain level of fuzziness. This requires that some degree of uncertainty be introduced in the models, in order to make the models realistic. Fuzzy logic fares well in this area. Many of the problems of the existing effort estimation models can be solved by incorporating fuzzy logic. Besides, fuzzy logic had been combined with algorithmic, non-algorithmic effort estimation models as well as a combination of them to deal with the inherent uncertainty issues. This paper also described an enhanced fuzzy logic model for the estimation of software development effort. The model (FLECE) possesses similar capabilities as the previous fuzzy logic model. In addition to that, the enhancements done in FLECE improved the empirical accuracy of the previous model in terms of MMRE (Mean Magnitude of Relative Error) and threshold-oriented prediction measure or prediction quality (pred).

Keywords: Fuzzy logic, cost estimation

1.0 INTRODUCTION

Software metric refers to the measurement of software attributes which are typically related to the product, the process and the resources of software development [1]. These measurements can be used as parameters in project management models [2] which provide aids to software project managers in the daunting task of managing software projects to avoid problems such as cost overrun and behind the schedule.

One of the most extensively researched areas of software measurement is software effort prediction. Software effort prediction models fall into two main categories: algorithmic and non-algorithmic. The most popular algorithmic prediction models include Boehm's COCOMO [3], Putnam's SLIM [4] and Albrecht's Function Point [5]. These models require as inputs, accurate estimate of certain attributes such as line of code (LOC), complexity and so on which are difficult to obtain during the early stage of a software development project. The models also have difficulty in modeling the inherent complex relationships between the contributing factors, are unable to handle categorical data as well as lack of reasoning capabilities [6]. The limitations of algorithmic models led to the exploration of the non-algorithmic techniques which are soft computing based. These include artificial neural network, evolutionary computation, fuzzy logic models, case-based reasoning and so on.

Artificial neural network are used in effort estimation due to its ability to learn from previous data [7][8]. It is also able to model complex relationships between the dependent (effort) and independent variables (cost drivers) [7][8]. In addition, it has the ability to generalize from the training data set thus enabling it to produce acceptable result for previously unseen data. Most of the work in the application of neural network to effort estimation made use of feed-forward multi-layer Perceptron, Backpropagation algorithm and sigmoid function [7].

Despite of these, neural network's limitations in several aspects prevent it from being widely adopted in effort prediction [7]. It is a 'black box' approach and therefore it is difficult to understand what is going on internally within a neural network. Hence, justification of the prediction rationale is tough. Neural network is known of its ability in tackling classification problem. Contrarily, in effort estimation what is needed is generalization capability. At the same time, there is little guideline in the construction of neural network topologies. On the other hand, genetic programming was used in effort estimation by Burgess et al [9] and the use of case-based reasoning in effort estimation can be found in [8]. The work presented here focused on the application of fuzzy logic in effort prediction.

The next section presents the different ways fuzzy logic is being used in software effort estimation. Section 3 discusses the strength of fuzzy logic in effort estimation. Section 4 describes the enhanced fuzzy logic model. Section 5 shows the empirical results of the new fuzzy logic model. Analysis of result follows next and finally the conclusion.

2.0 FUZZY LOGIC

The fuzzy logic model uses the fuzzy logic concepts introduced by Lofti A. Zadeh [10]. Fuzzy reasoning consists of three main components [5]: fuzzification process, inference from fuzzy rules and defuzzification process. Fuzzification process is where the objective term is transformed into a fuzzy concept. The membership functions are applied to the actual values of variables to determine the confidence factor or membership value (MV). Fuzzification allows the input and output to be expressed in linguistic terms. Inferencing involves defuzzification of the conditions of the rules and propagation of the confidence factors of the conditions to the conclusion of the rules. A number of rules will be fired and the inference engine assigned the particular outcome with the maximum MV from all the fired rules. Defuzzification process refers to the translation of fuzzy output into objective terms.

2.1 Fuzzy logic in software effort estimation

Study showed that fuzzy logic model has a place in software effort estimation [2]. Using real project data, Gray and MacDonell compared Function Point Analysis, Regression techniques, feedforward neural network and fuzzy logic in software effort estimation. Their results showed that fuzzy logic model achieved good performance, being outperformed in terms of accuracy only by neural network model with considerably more input variables. In their fuzzy logic model, unadjusted function point and complexity adjustment factor were selected as the two most important variables. Triangular membership functions were defined for the small, medium, large intervals of size, complexity and effort. Expert knowledge was sought in defining the initial set of nine rules and later in fine-tuning them.

Later, Gray and MacDonell developed FULSOME (Fuzzy Logic for Software Metrics) which is a set of tools [11] that helps in creating fuzzy model. The functionalities provided include data entry, importing facilities, membership function construction, rule creation, inference and explanations of fuzzy logic. The automatically generated fuzzy model performs acceptably when compared to regression-based models.

A study had compared personal Fuzzy Logic Systems (FLS) with linear regression using evaluation criteria which is based upon ANOVA of MRE and MER, as well as MMRE, MMER and pred(25) [12]. Eighteen programmers developed ninety-nine programs. From these programs four FLS are generated to estimate the effort of sixty programs developed by ten developers. Results show that a FLS can be used as an alternative for estimating the development effort at personal level.

Fuzzy logic has also found its way into the effort estimation of object-oriented software [13]. FUSP (Fuzzy use case size points) metric allows gradual classifications of use case size points in the effort estimation by using fuzzy numbers. Results showed that FUSP fares better than USP. Other research on fuzzy logic in software effort estimation involves the incorporation of fuzzy logic into algorithmic models, non-algorithmic models and the combination of algorithmic and non-algorithmic models. These are discussed in the following sections.

2.2 Fuzzy logic in algorithmic models

Fuzzy logic had also been applied to algorithmic models to cater for the need of fuzziness in the input and the resulting estimates of the models. The most common algorithmic model which has been incorporated with fuzzy logic is COCOMO.

Ryder researched on the application of fuzzy logic to COCOMO and Function Points models [14]. Two approaches were suggested. For both models, the first approach incorporates fuzzy logic to the cost drivers in the calculation of the adjustment factor. The second approach introduced fuzzy logic at the preliminary estimation of nominal effort for COCOMO and the preliminary estimation of unadjusted function points for Function Points. In the second approach, an adjustment factor is also included in the models using linguistic variable.

Musflek et al worked on fuzzifying basic COCOMO model without considering the adjustment factor [15]. In their simple f-COCOMO model, the size input into the COCOMO model is represented by a fuzzy set, while *a* and *b* coefficients are crisp values. Besides the size, augmented f-COCOMO also fuzzified both the coefficients related to the development mode. Due to the fact that uncertainty at the input level yields uncertainty at the output, the resulting effort estimate is also represented by a fuzzy set.

On the other hand, Idri et al proposed fuzzy intermediate COCOMO'81 [16]. What is fuzzified here is the cost drivers. The effort multiplier for each cost driver is obtained from fuzzy set, enabling its gradual transition from one interval to a contiguous interval (such as from *high* to *very high*). Validation results showed that the fuzzy intermediate COCOMO'81 can tolerate imprecision in its input (cost drivers) and generate more gradual outputs. Thus fuzzy intermediate COCOMO'81 is less sensitive to the changes in the inputs as compared to intermediate COCOMO'81.

Saliu et al go further by fuzzifying the two different portions of the intermediate COCOMO model i.e. nominal effort estimation and the adjustment factor [6]. They proposed a fuzzy logic framework for effort prediction by integrating the fuzzified nominal effort and the fuzzified effort multipliers of the intermediate COCOMO model. This approach is able to deal with uncertainty, provides transparency on prediction rationale through rules, incorporate experts knowledge in the definition of membership functions and rules, as well as adaptable to new data by changing the parameters of membership functions.

Kumar et al [17] had applied fuzzy logic in Putnam's manpower buildup index (MBI) estimation model. MBI selection process was based upon 64 different fuzzy associative memory (FAM) rules. The work showed how fuzzy FAM's can be effectively applied to the domain of software project management and control for the estimation of the MBI.

2.3 Fuzzy logic in non-algorithmic models

Besides being applied to algorithmic models, fuzzy logic is also combined with other non-algorithmic models with the same intention i.e. to cater for uncertainty in the latter models. One such approach is fuzzy analogy where fuzzy logic is combined with estimation by analogy [8]. Estimation by analogy is one of the popularly-used techniques classified under expert-based estimation method [18]. It is in fact a type of Case-based Reasoning (CBR). For effort estimation, CBR is based on the affirmation that similar *software projects have similar costs*. Earlier on, Idri et al proposed to measure the similarity between projects using categorical data represented by fuzzy set and to compute other measures using fuzzy reasoning [19]. Categorical data is a type of linguistic values. It belongs to either nominal or ordinal scale type. Nominal type allows the classification of the data in different categories whereas ordinal type allows classification as well as the ordering of the categories. Fuzzy analogy is able to handle software projects described using either numerical or linguistic values. The validation using COCOMO'81 dataset revealed that fuzzy analogy performed better than intermediate COCOMO'81 as well as the fuzzy intermediate COCOMO'81 developed earlier [16].

Later, the use of fuzzy analogy for software cost estimation had also been applied to Web software [20]. Due to the insufficiency of Web software attributes to empirically build their fuzzy representations, this study used Fuzzy C-Means clustering technique (FCM) and a Real Coded Genetic Algorithm (RCGA) to build these fuzzy representations.

Another research combined fuzzy logic with neural network to tackle the inherent problem that neural network is a 'black box' [7]. This is done through a method to map neural network to fuzzy rule-based system to make the interpretation of cost estimation models easier. This study made use of a Backpropogation three-layer Perceptron network and COCOMO'81 dataset. Benitez's method was used to extract the fuzzy rules from the network. The results showed that the output and the premise of each rule can be explained. However, the i-or operator is not suitable to combine the effects of each cost driver/premise on the output (effort).

Stefanowski's study used rough sets and rule induction to identify the most discriminatory cost factors, extract meaningful rule representation of classification knowledge from data and construct accurate rule-based classifiers [21]. COCOMO data set was used in the study.

2.4 Hybrid of fuzzy logic with algorithmic and non-algorithmic models

Research had also been done to incorporate fuzzy logic with neural network and COCOMO 2.0 i.e. neuro-fuzzy COCOMO in the pursuit of achieving better estimation results [22]. Based on COCOMO 2.0 post architecture model, the input of neuro-fuzzy COCOMO consists of size and 22 cost drivers (5 scale factors plus 17 effort multipliers). Twenty two sub-models are constructed to translate the qualitative rating of each of the 22 cost drivers into a quantitative multiplier value and to calibrate these relations by industry project data. The outputs from these sub-models are fed into the neuro-fuzzy COCOMO together with the size. Monotonic constraints which reflect expert knowledge are applied to ensure that results are reasonable. Validation done using industry project data and COCOMO'81 dataset indicates increase of accuracy when compared with standard COCOMO model. The combination of neural network and fuzzy logic brings the benefits of learning ability, good interpretability, and generalization as well as allowing the intervention of experts.

3.0 WHY FUZZY LOGIC?

There are a number of problems with the existing effort estimation models [2, 23, 24, 14]. Firstly most prediction models required highly accurate estimates to be supplied as input into the models. The high level of accuracy of these input data is difficult to obtain especially during the early stage of software development when prediction activity is most paramount. Secondly, these estimation models usually produce crisps results which warranted over-commitment to the model outputs. The crisps results are in fact plausible since it is doubtful that uncertain estimated input can produce accurate output. Besides, linear regression is usually used to correlate cost drivers which do not have linear relationships at all. Data collection is difficult and expensive thus the available data sets are usually limited in terms of size, timeliness and consistency for constructing estimation models.

The application of fuzzy logic is able to overcome some of the problems which are inherent in existing software metrics models as outlined above. Fuzzy logic effort prediction brings numerous benefits. Undeniably the development of software is characterized by parameters that possess certain level of fuzziness [17]. This requires that some degree of uncertainty be introduced in the models, in order to make the models realistic. Moores and Edwards discovered that the accepted level of estimation accuracy was typically +/- 20% [25]. This shows that software project managers are aware of the irrefutable uncertainty in estimation and therefore expect some inaccuracy in the prediction output. Besides, the use of formal models did not improve the accuracy of estimation [26].

Fuzzy logic enables linguistic representation of the input and output of a model to tolerate imprecision [2][14][6]. It is particularly suitable for software estimation as many software attributes are measured on nominal or ordinal scale type which is a particular case of linguistic values [8]. The results may come in less precision but it is able to avoid over-commitment to the model output due to the fuzziness of the output value instead of crisp output value [2][24][14]. The use of fuzzy set satisfies the first criterion of soft computing which is the tolerance of imprecision, as defined by Zadeh [27]. More importantly, the model should support the fact that human reasons in fuzziness.

The use of fuzzy set supports continuous belongingness (membership) of elements to a given concept (such as small software project) [22] thus alleviating a dichotomy problem (yes/no) [15] that caused similar projects having different estimated efforts.

Apart from that, fuzzy logic approach is less dependent on historical data [28] [11] [14]. Fuzzy logic models can be constructed without any data or with little data [28][11]. This makes fuzzy logic superior over data-driven model building approaches such as neural network, regression and case-based reasoning. In addition, fuzzy logic models can adapt to new environment when data become available [6].

Another advantage of fuzzy logic model is that it has the ability to represent different levels of uncertainty for the inputs and outputs whilst inferring based on the same model (rules and membership functions). Consequently it is able to cater for the needs of different level of precision for the different stages of development life cycle [28][11]. Table 1 shows the suggested levels of precision across the software development life cycle when estimating development effort. For example input variables can be expressed as fuzzy label (*large, medium, small*), fuzzy number (*about* 100) or crisp value (*100*). The use of fuzzy logic variables for the metrics and models also allow the development effort to be estimated much earlier in the development process [2].

Table 1: Suggested levels of precision across the life-cycle
when estimating development effort [28]

Phase	**Inputs**	**Outputs**
Analysis	fuzzy label	fuzzy label
Design	fuzzy number	fuzzy number
Coding	crisp value or fuzzy number	fuzzy number
Testing	crisp value	fuzzy number
Maintenance (small project)	crisp value	fuzzy number

Molokken and Jorgensen analyzed a number of surveys done on software effort estimation [18]. Expert judgment appeared to be the most frequently used effort estimation. Earlier work done by Bergeron and St-Arnaud found that the most accurate estimates were based on analogy and expert opinion [29]. Expert-based estimation was also found to be better than all regression-based models [18]. Henceforth the use of fuzzy logic in effort estimation is desirable since expert knowledge can be incorporated into the fuzzy effort estimation models [22][24][6]. Incorporation of expert-knowledge can also reduce the number of input variables [22][14][6].

Fuzzy logic also improves the interpretability of the model allowing the user to view, evaluate, criticize and adapt the model. Prediction can be explained through rules [2][6]. Following the interpretability of fuzzy logic model, experts can check the model to avoid adversary effects by unusual data [28] [11]. This increases the robustness of the model. Apart from that, fuzzy logic models can be easily understood when compared to regression models and neural network. This makes it an effective communication tool for management [28][11]. Case-based reasoning has the same transparency but requires high volume of data.

4.0 ENHANCED MODEL

FLECE (Fuzzy Logic Model for Software Development Effort and Cost Estimation), the model in this study take into consideration the problems of effort estimation models outlined in the previous section. FLECE is based on the fuzzy logic model for effort estimation proposed by Gray and MacDonell [2]. Several enhancements had been done to the earlier model [30]. First of all, FLECE provides two different types of patterns for membership function namely triangular and trapezium. In the earlier model only triangular pattern is provided. This poses some limitation since triangular pattern may not cater for certain situation. Taking for example, in a triangular membership function, system size with 50 unadjusted function points (UFP) is considered 0.8 small and 0.2 medium, but size with 55 UFP is considered 0.7 small and 0.3 medium. However, there may be cases where 50 and 55 UFP should be mapped to the same values due to the small differences between these two UFPs.

The fuzzy logic model proposed in [2][23] only considered two factors namely the program size (unadjusted function point) and program complexity. A third factor is added in FLECE which is developers' experience [30]. Due to the fact that software production is a labor-intensive activity, there is a close relationship between the ability of the developers and the effort. There are two aspects of ability which need to be considered: the general competence of the individual [31], and the familiarity of the individual with the particular application area. For the latter aspect, the experience of the developers is considered. The more experienced developer will be able to derive better solution to problem and spend less time. Hence an experienced developer requires less effort compared to a less experienced developer.

In the previous study [2], the language level has no influence on the model. However, in FLECE different level of programming languages will have different impact [30]. It is used in the selection of membership function. For example a program written in low-level programming language such as assembly language with 200 UFP is considered large in size, complicated, required experienced developers and consumed a lot of development effort. However a program written in 4GL language with 200 UFP is considered medium in size, less complicated, required less experienced developers and less effort. To cater for the impact of the different category of programming language to the model, different membership functions are used for different category of programming language.

Three categories of levels of programming language are taken into consideration in this model [32]. These essentially correspond to:

i. The use of more traditional languages in a mainframe environment (e.g. COBOL language with an IMS-type database or the equivalent) is refers to as category 1.
ii. The use of higher-level language like ADSO, CAPEX, etc is refers to as category 2.
iii. The use of fourth-generation language or higher is refers to as category 3.

The FLECE model is shown in Fig. 1. It allows the input to be either crisp values or linguistic values. If crisp value is inputted to the model, it has to go through the fuzzification process. Otherwise if linguistic values are used as input they can be fed directly into the rule-based inference process. The output (development effort) can be either in linguistic values or crisp value after the defuzzification process.

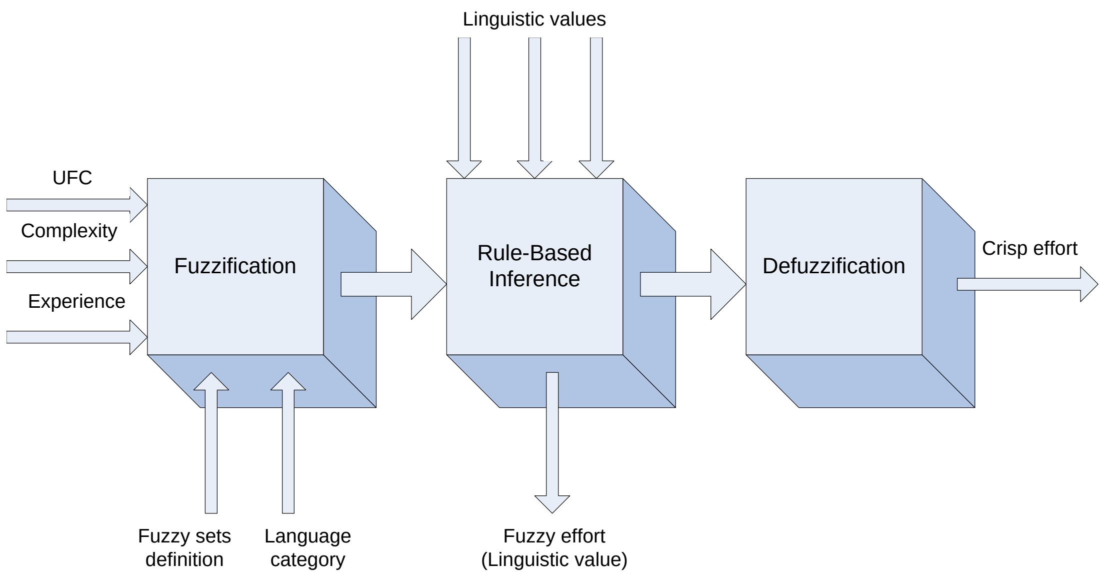

Fig. 1: FLECE model

5.0 EMPIRICAL STUDY

The empirical study carried out here is based on the empirical study done by Gray and MacDonell [2] with their results shown in Table 2 (row 1 to row 7). They used the sets of system development projects data from Desharnais [32]. In their study, they had randomly selected 54 sets of data for the training of the techniques and the remaining one-third of the data was used for the validation of each technique. To be comparable with their results, the same sets of data were used in the setting of the membership functions and fuzzy inference rules and the same remaining sets of data were used for validation in FLECE [30]. The result of FLECE is shown by the last row in Table 2.

Table 2: Overall Empirical Results [2], [30]

	Method	MMRE	**pred(10) %`**	**pred(25) %**
1	FPA estimation (mean-based)	0.7	4	22
2	FPA estimation (median-based)	0.89	19	41
3	LS regression	0.86	15	41
4	LS regression (no outlier)	0.88	30	56
5	LMS regression	0.85	7	41
6	Neural network	0.44 (1)	26 (2)	63 (1)
7	Fuzzy logic	0.54 (3)	7 (6)	30 (7)
8	FLECE	0.47 (2)	19.1 (3)	30.6 (6)

Note: The figure in bracket indicates the ranking.

The techniques used in the metric analysis to indicate the adequacy of the predictive model were mean magnitude of relative error (MMRE) and threshold-oriented prediction measure (pred) [1] [2]. MRE (magnitude of relative error) measures the difference between the actual values of the dependent variable and the predictions of the model. MMRE is the mean of MRE. By definition, the smaller the value of MMRE, the better is the model. The smaller of MMRE indicates that the model produces on average a good set of predictions. The pred measure indicates the overall fit for a set of data points, based on the MRE values for each data point. For example if pred(m) = n, then n% of the predicted values fall within m% of their actual values [1]. The larger the value of pred, the better the model is. The larger value indicates more data points fall within the range of error.

As FLECE uses different membership functions for different categories of programming language, there are 3 sets of data for each language category. The averages of these values are used to compare with Gray and MacDonell's results. The averages MMRE, pred(10), pred(25) are derived using this formula:

$$\text{Average} = (E1 * N1 + E2 * N2 + E3 * N3) / \text{Total number of data}$$

For example, to calculate the average of MMRE, E1 refers to the MMRE of FLECE for programming language in category 1, E2 refers to the MMRE for programming language in category 2 and E3 refers to the MMRE for programming language in category 3, respectively. N1 is the number of data for projects developed using programming language in category 1, N2 is for programming language in category 2, and N3 is for programming language in category 3, respectively. Total number of data is the sum of N1, N2 and N3.

6.0 ANALYSIS OF RESULT

From the result shown in Table 6, in terms of MMRE, neural network model constructed by Gray and McDonell showed the best result. Consequently if enough historical data are available and the users are willing to compromise with its inherent limitations, neural network is able to produce more accurate estimates. On the other hand, FLECE outperforms the previous fuzzy logic model in all the measurements. The results show that the average MMRE of FLECE is 0.47. This is smaller than the corresponding value obtained from the previous fuzzy logic model (0.54) and thus is better. Besides, the pred(10) and pred(25) for FLECE are 19.1% and 30.6% respectively which are greater than the corresponding values (7% and 30%) obtained by the previous fuzzy logic model. This indicates that more percentage of the predicted values fall within 10% and 25% of their actual values for FLECE as compared to the previous fuzzy logic model.

7.0 CONCLUSION

The analysis of the results shows that FLECE is able to obtain more accurate results in the estimation of software development effort when compared to the previous fuzzy logic model. Hence, the enhancements to FLECE are truly useful and had given better performance to the model. At the same time, this study also shows the applicability and the strength of using fuzzy logic in software development effort estimation.

REFERENCES

[1] N. E. Fenton, S. L. Pfleeger, *Software Metrics, A Rigorous and Practical Approach*, 2nd Edition, PWS Publishing Company, Thomson Publishing, Boston, 1997.

[2] A. R. Gray, S. G. MacDonell, "Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation". *Fuzzy Information Processing Society 1997 NAFIPS' 97, Annual Meeting of the North American*, 21 – 24 September 1997, pp. 394 – 399.

[3] B. W. Boehm, *Software Engineering Economics*, Englewood Cliffs, NJ, Prentice Hall, 1981.

[4] L. H. Putnam, "A general empirical solution to the macrosoftware sizing and estimating problem". *IEEE Transactions on Software Engineering*, SE-4(4), 1978, pp 345-361.

[5] Attar Software. 2002. "Fuzzy Logic in Knowledge Builder", White Paper. http://www.intellicrafters.com/fuzzy.htm

[6] M.O. Saliu, M. Ahmed and J. AlGhamdi. "owards adaptive soft computing based software effort prediction" *Fuzzy Information, 2004. Processing NAFIPS '04. IEEE Annual Meeting of the North American Fuzzy Information Processing Society*, 27-30 June 2004, Vol.1, pp. 16-21.

[7] A. Idri, T. M. Khoshgoftaar, A. Abran. "Can neural networks be easily interpreted in software cost estimation?", *IEEE Trans. Software Engineering*, Vol. 2, 2002, pp. 1162 – 1167.

[8] A. Idri,, A. Abran,, T.M. Khoshgoftaar. "Estimating software project effort by analogy based on linguistic values" in.*Proceedings of the Eighth IEEE Symposium on Software Metrics,* 4-7 June 2002, pp. 21 – 30.

[9] C. J. Burgess, M. Lefley. "Can Genetic Programming improve software effort estimation? A comparative evaluation", *Information and Software Technology,* Vol. 43, No.14, 2001, pp. 863-873.

[10] L. A. Zadeh. "Fuzzy Sets". *Information And Control*, Vol. 8, 1965, pp. 338-353.

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

[12] L.M. Cuauhtemoc, Y.M. Cornelio, G.T. Agustin. "Fuzzy Logic Systems for Software Development Effort Estimation Based Upon Clustering of Programs Segmented by Personal Practices", *Electronics, Robotics and Automotive Mechanics Conference (CERMA'06),* 2006, pp. 367-372.

[13] M.R. Braz, S.R. Vergilio, "Using fuzzy theory for effort estimation of object-oriented software". *16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004*, 15-17 Nov 2004, pp. 196 – 201.

[14] J. Ryder, "Fuzzy Modeling of Software Effort Prediction" in *Proceeding. of IEEE Information Technology Conference,* Syracuse, NY, 1-3 Sept 1998, pp: 53-56.

[15] P. Musflek, W. Pedrycz, G. Succi, M. Reformat, "Software Cost Estimation with Fuzzy Models", *Applied Computing Review*, Vol. 8, No.2, 2000, pp. 24-29.

[16] A. Idri, A. Abran, L. Kjiri. „COCOMO cost model using Fuzzy Logic", *7th International Conference on Fuzzy Theory & Techniques*, 27 Feb – 3 March 2000, Atlantic City, New Jersey.

[17] S. Kumar, B.A. Krishna, and P.S. Satsangi, „Fuzzy systems and neural networks in software engineering project management". *Journal of Applied Intelligence,* Vol. 4, 1994, pp. 31-52.

[18] K. Molokken, M. Jorgensen, "A review of software surveys on software effort estimation"in *Proceedings of IEEE International Symposium on Empirical Software Engineering, ISESE 2003*, 30 Sept.-1 Oct. 2003, pp: 223 – 230.

[19] A. Idri, A. Abran, "A fuzzy logic based set of measures for software project similarity: validation and possible improvements" in *Proceedings of Seventh International Software Metrics Symposium, METRICS 2001*, 4-6 April 2001, pp.85 – 96.

[20] A Idri, A. Zahi, A. Abran, "Software Cost Estimation by Fuzzy Analogy for Web Hypermedia Applications" in *Proceedings of the International Conference on Software Process and Product Measurement,* 6 – 8 Nov, 2006, Cadiz, Spain, pp. 53 – 62.

[21] J. Stefanowski, "An empirical study of using rule induction and rough sets to software cost estimation", *Fundamenta Informaticae*, *Special issue on theory and applications of soft computing (TASC04*), Vol. 71, No. 1, May 2006, pp. 63 – 82.

[22] X. Huang, L. F. Capretz, J. Ren, D. Ho, "A Neuro-Fuzzy Model for Software Cost Estimation". *IEEE Proceedings of the Third International Conference On Quality Software QSIC'03,* 2003.

[23] A. R. Gray, and S. G. MacDonell, "A Comparison of Techniques for Developing Predictive Models of Software Metrics", *Information and Software Technology 39,* 1997, pp. 425 – 437.

[24] S. G. MacDonell, A. R. Gray, "A Comparison of Modeling Techniques for Software Development Effort Prediction" in *Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems,* Springer-Verlag, 1997, pp. 869 – 872.

[25] T.T. Moores, and J.S. Edwards, "Could Large UK Corporations and Computing Companies Use Software Cost Estimating Tools? - A Survey". *European Journal of Information Systems,* Vol.1, No.5, 1992, pp.311-319.

[26] F.J. Heemstra, R.J. Kusters. "Controlling Software Development Costs: A Field Study". *International Conference on Organisation andInformation Systems*, Bled, Yugoslavia, 1989.

[27] L.A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing", *Communication of ACM,* Vol.37, No.3, March 1994, pp.77-84.

[28] A. R. Gray, S. G. MacDonell, “Fuzzy Logic for Software Metric Models throughout the Development Life-Cycle” in *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society – NAFIPS,* 10 – 12 June 1999, New York, USA, pp. 258 – 262.

[29] F. Bergeron, J. F. St-Arnaud, “Estimation of Information Systems Development Efforts: A Pilot Study”. *Information & Management*, Vol. 22, 1992, pp.239-254.

[30] P. Y. Man. “Enhanced Software Development Cost and Effort Estimation Using Fuzzy Logic Model”. Master’s Thesis, University of Malaya, 2000.

[31] H. Sackman, et al., “Exploratory Experimental Studies Comparing Online and Offline Programming Performance”, *Comm. ACM,* Vol.11, No.1, January 1968.

[32] J. M. Desharnais, “Analysis *Statistique de la productivitie des projects de development en informatique apartir de la technique des points des fontion*“. Master’s Thesis, Universite du Montreal, 1989.

BIOGRAPHY

Su Moon Ting is a lecturer at Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. Her current research interests include software architecture, e-Learning and web services.

Ling Teck Chaw obtained his M. Comp. Sc. and PhD in 1996 and 2005 from University of Malaya, Malaysia. He is a senior lecturer at Faculty of Computer Science & Information Technology, University of Malaya. His research areas include core network research, inter-domain Quality of Service (QoS), Voice over IP (VoIP), grid computing and network security.

Phang Keat Keong is a senior lecturer at Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. He obtained his PhD in 2004 from University of Malaya. His current research interests include high speed computer network, network QoS, grid computing, traffic conditioning and traffic engineering, and fuzzy logics.

Liew Chee Sun is a lecturer at the Faculty of Computer Science & Information Technology, University of Malaya, Malaysia. He obtained his M. Comp. Sc. in 2002 from Universiti Sains Malaysia. Prior to working at University of Malaya, he was the core researcher of the Grid Computing Research Lab in USM. His current research interests include Grid computing, Peer-to-Peer computing and VoIP.